

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

**BOOKIE: COOKIE THEFT THROUGH XSS ATTACKS AND ITS
RELATIONSHIP WITH PERSONAL DATA LEAKAGE: A
FRAMEWORK TO DEMONSTRATE IT TO UNIVERSITY
STUDENTS**

**THESIS SUBMITTED AS PART OF THE
REQUIREMENTS FOR THE AWARD OF THE DEGREE
OF DOCTOR OF PHILOSOPHY IN INFORMATICS**

GERMÁN EDUARDO RODRÍGUEZ GALÁN

german.rodriquez@epn.edu.ec

SUPERVISOR: JENNY GABRIELA TORRES OLMEDO

jenny.torres@epn.edu.ec

Quito, Julio 2024



ESCUELA
POLITÉCNICA
NACIONAL

TESIS

Para la obtención del título de

DOCTOR EN INFORMÁTICA

Resolución RPC-SO-43-No.501-2014

del Consejo de Educación Superior

Presentada por

**GERMÁN EDUARDO
RODRÍGUEZ GALÁN**

Tesis dirigida por

JENNY GABRIELA TORRES OLMEDO, PhD,

Profesor de la Escuela Politécnica Nacional (Ecuador)

**BOOKIE: COOKIE THEFT THROUGH XSS AT-
TACKS AND ITS RELATIONSHIP WITH PER-
SONAL DATA LEAKAGE: A FRAMEWORK TO
DEMONSTRATE IT TO UNIVERSITY STUDENTS**

Examen oral en **FECHA,**

por la siguiente comisión:

Luis Patricio Tello Oquendo, Ph.D.

Universidad Nacional de Chimborazo, External Member, Coordinator.

Luis Enrique Sánchez Crespo, Ph.D.

Universidad de Castilla-La Mancha, External Member.

Gabriela Lorena Suntaxi Oña, Ph.D.

Escuela Politécnica Nacional, Internal Member - Oponent.

Luis Felipe Urquiza Aguiar, Ph.D.

Escuela Politécnica Nacional, Internal Member.

Ángel Leonardo Valdivieso Caraguay, Ph.D.

Escuela Politécnica Nacional, Internal Member

DECLARACIÓN

I hereby declare under oath that I am the author of this work, which has not previously been presented for obtaining any academic degree or professional qualification. I also declare that I have consulted the bibliographic references included in this document.

Through this declaration, I transfer my intellectual property rights corresponding to this thesis, to the Escuela Politécnica Nacional, as established by the Intellectual Property Law of Ecuador, its Regulations and the current institutional norms.

I declare that this work is based on the following articles of my authorship (as main author or co-author) related to the title of this thesis:

Conference Proceedings in Print (Paper Presented at a Conference):

- Rodríguez, G.E., Benavides, D.E., Torres, J., Flores, P., Fuertes, W. (2018). Cookie Scout: An Analytic Model for Prevention of Cross-Site Scripting (XSS) Using a Cookie Classifier. In: Rocha, Á., Guarda, T. (eds) Proceedings of the International Conference on Information Technology Systems (ICITS 2018). ICITS 2018. Advances in Intelligent Systems and Computing, vol 721. Springer, Cham.
- Rodríguez, G.E., Torres, J.G., Benavides-Astudillo, E. (2023). DataCookie: Sorting Cookies Using Data Mining for Prevention of Cross-Site Scripting (XSS). In: Daimi, K., Alsadoon, A., Peoples, C., El Madhoun, N. (eds) Emerging Trends in Cybersecurity Applications. Springer, Cham.
- Rodríguez, G., Torres, J., Flores, P., Benavides, E., Proaño, P. (2020). Trusted Phishing: A Model to Teach Computer Security Through the Theft of Cookies. In: Bottotobar, M., León-Acurio, J., Díaz Cadena, A., Montiel Díaz, P. (eds) Advances in Emerging Trends and Technologies. ICAETT 2019. Advances in Intelligent Systems and Computing, vol 1067. Springer, Cham.
- G. Rodriguez, J. Torres, P. Flores, E. Benavides and D. Nuñez-Agurto, "XSStudent: Proposal to Avoid Cross-Site Scripting (XSS) Attacks in Universities," 2019 3rd Cyber

Security in Networking Conference (CSNet), Quito, Ecuador, 2019, pp. 142-149

- Rodríguez, G.E., Torres, J., Benavides, E. (2022). XSS2DENT, Detecting Cross-Site Scripting Attacks (XSS) Vulnerabilities: A Case Study. In: Botto-Tobar, M., Montes León, S., Torres-Carrión, P., Zambrano Vizueté, M., Durakovic, B. (eds) Applied Technologies. ICAT 2021. Communications in Computer and Information Science, vol 1535. Springer, Cham.
- Germán Rodríguez-Galán, Jenny Torres-Olmedo, Luis Chica-Moncayo. (2024). Hack-MySelf: Decrypting Cookies to Show the Theft of Personal Data in University Students. Aproved for oral presentation in ICR'24. INTERNATIONAL CONFERENCE ON INNOVATIONS IN COMPUTING RESEARCH (August 2024).

Journals:

- Germán E. Rodríguez, Jenny G. Torres, Pamela Flores, Diego E. Benavides, Cross-site scripting (XSS) attacks and mitigation: A survey, Computer Networks, Volume 166, 2020, 106960, ISSN 1389-1286.
- Rodríguez-Galán, G., Torres, J. Personal data filtering: a systematic literature review comparing the effectiveness of XSS attacks in web applications vs cookie stealing. Ann. Telecommun. (2024).

I also declare that I have acknowledged the collaboration of third parties, and the contribution made by other published or unpublished material.

(GERMÁN EDUARDO RODRÍGUEZ GALÁN)

CERTIFICACIÓN

I certify that GERMAN EDUARDO RODRIGUEZ GALÁN has carried out his/her research under my supervision. To the best of my knowledge, the contributions of this work are novel.

(Ph.D. JENNY GABRIELA TORRES OLMEDO)
ADVISOR

DEDICATORIA

A mi esposa *Verónica*, por su amor incondicional, apoyo constante y sacrificio invaluable a lo largo de esta travesía académica.

A mis hijos *Sebastián y Mateo*, si algún día llegan a leer este legado, quiero que sepan que cada palabra que escribí fué pensando en Ustedes, quiero que lleguen mas lejos de lo que yo he llegado y quiero que este logro sea un ejemplo para los dos.

A mis amigos y colegas, por su compañía, estímulo y comprensión en los momentos de alegría y de desafíos. Su amistad ha sido un pilar fundamental en este camino.

A mis profesores y mentores, en especial a mi Tutora y Directora, Dra. Jenny Torres, por su orientación sabia, enseñanzas profundas y confianza en mis capacidades. Ha sido un guía esencial en mi formación académica y profesional.

A mis padres *Germán y Luz María*, mis hermanos *Andrés y Lisseth*, mis sobrinos *Joel y José*, porque siempre creyeron en mi, por su paciencia y comprensión que han sido mi refugio en los momentos más difíciles y mi motivación en los momentos de éxito.

A todas las personas que, de una manera u otra, contribuyeron a este logro, les expreso mi más sincero agradecimiento. Este trabajo es también el fruto de su apoyo y amistad.

AGRADECIMIENTOS

Quisiera expresar mi profunda gratitud a la Dra. Jenny Torres, por su guía experta, paciencia y dedicación durante todo el proceso de investigación y redacción de esta tesis. Sus consejos y conocimientos fueron fundamentales para alcanzar los resultados presentados. Le agradezco sinceramente por su valiosa colaboración, comentarios constructivos y apoyo inquebrantable a lo largo de este viaje académico. Su compromiso y visión fueron cruciales para el desarrollo de este trabajo.

Deseo reconocer el respaldo financiero otorgado por la Universidad de las Fuerzas Armadas-ESPE, por el apoyo económico para el pago de algunas de las contribuciones científicas que se publicaron. Su apoyo fue fundamental para la realización de este trabajo.

A mis compañeros y colegas, por sus discusiones estimulantes, aportes valiosos y camaradería que enriquecieron este proyecto. Su colaboración fue vital en la consecución de los resultados obtenidos.

Agradezco a mi familia por su amor, comprensión y ánimo incondicional durante este arduo proceso. Su apoyo moral fue un motor que me impulsó a seguir adelante en los momentos más desafiantes.

Un agradecimiento especial a todos mis estudiantes que han pasado por la asignatura de Seguridad Informática, su apoyo fue fundamental en la recolección y análisis de datos para la realización de este trabajo.

Finalmente, quiero agradecer a todas las personas que de alguna manera contribuyeron a este trabajo, así como a las instituciones que facilitaron los recursos necesarios para su realización. Su contribución ha sido fundamental en la culminación de esta etapa académica.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Research Context | 1 |
| 1.2 | Justification | 4 |
| 1.3 | Research questions and objectives | 6 |
| 1.3.1 | General objective | 6 |
| 1.3.2 | Specific objectives | 6 |
| 1.3.3 | Research Questions(RQ) | 7 |
| 1.4 | Research methodology and approach | 8 |
| 1.4.1 | Phase A: Characterization | 9 |
| 1.4.2 | Phase B: Replication and Collect: | 11 |
| 1.4.3 | Phase C: Training and Predict | 11 |
| 1.4.4 | Phase D: Visualization | 12 |
| 1.5 | Motivation | 12 |
| 1.6 | Scientific Contribution | 14 |
| 2 | Background | 16 |
| 2.1 | COOKIES | 16 |
| 2.1.1 | ¿What are cookies? | 16 |
| 2.1.2 | Cookie attributes: | 17 |
| 2.1.3 | Cookie vulnerabilities | 19 |
| 2.1.4 | Most common types of cookies | 20 |
| 2.1.5 | Policies for the use of cookies: | 21 |
| 2.2 | CROSS-SITE SCRIPTING (XSS) | 22 |
| 2.2.1 | Types of XSS attacks and exploitation examples | 24 |
| 2.2.2 | Common scenarios for executing XSS attacks | 27 |
| 2.3 | COOKIES THEFT TROUGH XSS ATTACKS | 30 |

| | | |
|----------|---|-----------|
| 2.3.1 | Background of XSS attacks and theft of cookies | 30 |
| 2.3.2 | Statistics to remedy cookie theft: | 30 |
| 3 | Literature Review | 32 |
| 3.1 | Cross-site scripting (XSS) attacks and mitigation: A survey | 32 |
| 3.1.1 | Analysis of Methods and Tools | 32 |
| 3.1.2 | Analysis of results | 45 |
| 3.2 | A Systematic Literature Review Comparing the Effectiveness of XSS attacks in Web Applications vs Cookie Stealing | 47 |
| 3.2.1 | Protocol for systematic document review | 47 |
| 3.2.2 | Data collection process | 53 |
| 3.2.3 | Methodology used to analyze the content of documents | 54 |
| 3.2.4 | Analysis of Results | 56 |
| 4 | The roadmap to the BOOKIE framework | 67 |
| 4.1 | PHASE A: CHARACTERIZATION | 68 |
| 4.1.1 | Sub Phase 1: XSS Attacks | 68 |
| 4.1.2 | Sub Phase 2: Cookie Behaviour | 77 |
| 4.1.3 | Sub Phase 3: Cookie Theft | 88 |
| 4.1.4 | Sub Phase 4A. Personal Data Filtering | 96 |
| 4.1.5 | Sub Phase 4B: XSS2DENT | 104 |
| 4.2 | PHASE B: REPLICATE AND COLLECT | 113 |
| 4.2.1 | Cookie Collector Phase 1 | 113 |
| 4.2.2 | Cookie Collector Phase 2 | 116 |
| 4.3 | PHASE C: TRAINING AND PREDICT | 124 |
| 4.3.1 | Training | 124 |
| 4.3.2 | Prediction | 126 |
| 4.4 | PHASE D: VISUALIZATION | 134 |
| 4.4.1 | Methodological proposal through a pen testing challenge | 134 |
| 4.4.2 | Using statistical methodology to analyze computer security challenge results. | 136 |
| 4.4.3 | Pen testing results | 137 |
| 4.4.4 | Knowledge assessment results | 139 |

| | | |
|----------|--|------------|
| 5 | How to use the BOOKIE Framework | 142 |
| 5.1 | Initial considerations | 142 |
| 5.2 | Steps to use BOOKIE Framework | 143 |
| 5.2.1 | Introduction | 143 |
| 5.2.2 | Framework Advantages | 143 |
| 5.2.3 | How to implement the framework through the graphical interface . . . | 143 |
| 6 | Conclusions and perspectives | 148 |
| 6.0.1 | Conclusions | 148 |
| 6.0.2 | Future Work | 151 |
| 7 | References | 153 |

List of Figures

| | | |
|------|---|----|
| 1.1 | Timeline and areas of application of our research proposal. | 8 |
| 1.2 | Methodological proposal to show university students how stealing cookies is related to personal data filtering. | 9 |
| 2.1 | Alert on the policy of use of cookies in browsers | 22 |
| 2.2 | Welcome message and notice of the use of cookies | 22 |
| 2.3 | Top Ten Software Errors Ranking | 23 |
| 2.4 | Scenario for a reflected or indirect XSS attack | 27 |
| 2.5 | Sample XSS code reflected injected in the pizza.com domain | 27 |
| 2.6 | Preview of Pizza.com domain before the XSS attack | 28 |
| 2.7 | Preview of Pizza.com domain after the XSS attack | 28 |
| 2.8 | Chrome browser response to XSS attack | 28 |
| 2.9 | Scenario for executing stored XSS attack | 29 |
| 2.10 | Scenario for DOM XSS attack | 29 |
| 2.11 | Current remediation rate according to the type of vulnerabilities | 31 |
| 2.12 | Remediation rate according to the XSS vulnerabilities type | 31 |
| 3.1 | Classification of tools/methods proposed and used to mitigate XSS attacks . | 45 |
| 3.2 | Classification of proposals that use some method of Artificial Intelligence to mitigate XSS attacks | 45 |
| 3.3 | Trends in the proposal of the traditional methods/tools to mitigate XSS attacks | 46 |
| 3.4 | Trends in the proposal of methods/tools using Artificial Intelligence to mitigate XSS attacks | 47 |
| 3.5 | Search strategy using the snowballing technique | 49 |
| 3.6 | Overview of the research domains included to build the search strings | 50 |
| 3.7 | Overview of the process of analysis and selection of seed papers | 51 |

| | | |
|------|---|----|
| 3.8 | Overview of the authors whose proposals relate the research domains (cookies + XSS) | 54 |
| 3.9 | Overview of the authors whose proposals relate the research domains (web + XSS) | 55 |
| 3.10 | A suggested approach for analyzing content and organizing information. . . . | 56 |
| 3.11 | Trend of the analysed works, assessment of the attribute Tools/Methods from 2018 to 2023 | 58 |
| 3.12 | Most used tools within the Software Tools category | 58 |
| 3.13 | Most used tools within the Machine Learning-AI category | 60 |
| 3.14 | Most used tools within the Web Browsers Analysis category | 60 |
| 3.15 | Most used tools within the Security Tools category | 61 |
| 3.16 | Most used tools within the Web Scrapers category | 61 |
| 3.17 | Trend of the analyzed works, evaluation of the attribute Type of XSS Attack . | 62 |
| 3.18 | Analyzed works supported by domains registered in Alexa.com | 62 |
| 3.19 | Comparison between XSS attacks aimed at web applications and XSS attacks aimed at cookie theft | 66 |
| 4.1 | BOOKIE: A framework to teach university students about filtering personal data by stealing cookies via xss attacks | 68 |
| 4.2 | CookieScout: An analytic model for prevention of Cross-Site Scripting | 69 |
| 4.3 | Flowchart to rate the reputation of cookies according to their attributes | 73 |
| 4.4 | Statistics of the expiration date of the cookies created | 76 |
| 4.5 | Cookies created that allow the execution of commands | 77 |
| 4.6 | DataCookie: Proposed Model for Obtaining Rules from Cookie Mining | 78 |
| 4.7 | Resume of records obtained from the Web Security Appliance (DataSet01) . | 78 |
| 4.8 | Resume of records obtained to structure the DataSet03 | 79 |
| 4.9 | Preview of DataSet.arff file formatted for analysis using Weka Software | 80 |
| 4.10 | ROC Area for J48 model - YES and NO Class | 85 |
| 4.11 | ROC Area for RandomTree model - YES and NO Class | 85 |
| 4.12 | ROC Area for RepTree model - YES and NO Class | 85 |
| 4.13 | Structure of the decision tree obtained with the J48 algorithm | 86 |
| 4.14 | View of Python algorithm to obtain the domain of cookies classified as suspicious according to the rules obtained with Weka | 87 |
| 4.15 | Suspect Domains Report, Output after executing the script developed in Python | 87 |

| | |
|---|-----|
| 4.16 Domain which contains JavaScript files extension susceptible to XSS attacks | 88 |
| 4.17 Advertisement category with fake Adobe Flash Player update message for phishing. | 88 |
| 4.18 Cookie generated for the domain codeonclick.com and redirected to the domain http://installupgradenow.bigtrafficsystemsforupgrades.date | 88 |
| 4.19 Proposed model to teach computer security through XSS attacks | 89 |
| 4.20 a) QR Code generated as a WhatsApp status, b) Status of WhatsApp with link to the compromised blog | 90 |
| 4.21 Data collected with the hidden visitor counter | 91 |
| 4.22 Post on Facebook with a false note to redirect to the compromised blog | 91 |
| 4.23 Proposed scheme to implement the compromised blog | 92 |
| 4.24 Compromised blog design using JavaScript code | 93 |
| 4.25 Preview of the message displayed by the blog with the stolen cookie information | 93 |
| 4.26 Statistics by Operating System obtained from the visitor counter | 95 |
| 4.27 Statistics by Browsers obtained from the visitor counter | 95 |
| 4.28 XXStudent: Model used to execute the XSS attack | 96 |
| 4.29 Preview of the flyer that were placed in the hallways of the University | 98 |
| 4.30 Website No.1 compromised with the Hook.js script | 98 |
| 4.31 Website No.2 compromised with the Hook.js script | 100 |
| 4.32 XSSStudent: proposed model to prevent XSS attacks | 100 |
| 4.33 JavaScript code that our web scraper searches within the compromised websites | 102 |
| 4.34 Results of the vulnerability check of our website /demos/basic.html | 103 |
| 4.35 Results of the vulnerability check of our website /demos/butcher/index.html | 103 |
| 4.36 Results obtained using the XSSCon software. | 104 |
| 4.37 Mobile operating systems recorded during data collection | 106 |
| 4.38 Type of users that accessed to the vulnerable website | 106 |
| 4.39 Gender of users who accessed the vulnerable website | 106 |
| 4.40 Average age of users who accessed the vulnerable website | 106 |
| 4.41 Information collected related to the brand of mobile devices | 106 |
| 4.42 Information collected related to the antivirus installed on mobile devices | 106 |
| 4.43 Record of the most used attack vector | 107 |
| 4.44 Total users hacked with our controlled XSS attack | 107 |

| | |
|--|-----|
| 4.45 XSS2DENT: Proposed model to analyze XSS vulnerabilities in the educational establishments of Santo Domingo de los Tsachilas City | 107 |
| 4.46 Mobile operating system recorded during victim data collection | 111 |
| 4.47 Manufacturer registered during victim data collection | 111 |
| 4.48 Percentage of devices with antivirus software or similar | 111 |
| 4.49 Network connection used to access the challenge | 111 |
| 4.50 Attack vector with the highest records during access to the challenge | 111 |
| 4.51 Universities and number of users who accessed the challenge from their local network | 111 |
| 4.52 System Log that shows a zombie connected from Peru | 112 |
| 4.53 World map showing the number of connections of the different countries registered in the Challenge System Log | 113 |
| 4.54 CookieCollector Phase 1: The system can anonymize and clean data from URLs in web search history. | 114 |
| 4.55 CokieCollector Phase 2:Configured test infrastructure to search the URL and extract the cookies-related, generated network traffic and third party domains. | 116 |
| 4.56 BotSoul, graphical description of how our cookie collector operates by automating bots in python. | 117 |
| 4.57 Entity relationship diagram of the datasets obtained in the collection phase | 126 |
| 4.58 Preview of BOOKIE, a browser for search and analyse cookies | 127 |
| 4.59 Preview of BOOKIE, a browser to predict the level of vulnerability of students to XSS attacks | 127 |
| 4.60 Option to display statistical results for each browsed web page | 128 |
| 4.61 Preview of K-MEANS Hypermarameters Analysis | 129 |
| 4.62 Preview of K-MEANS with 2 clusters | 130 |
| 4.63 Preview of K-MEANS with 5 clusters | 130 |
| 4.64 Preview of K-MEANS with 10 clusters | 131 |
| 4.65 Preview of clusters identified with DBSCAN | 132 |
| 4.66 Preview of Dendograms identified with Hierarchical Clustering | 133 |
| 4.67 Preview of Clusters identified with Hierarchical Clustering | 133 |
| 4.68 NIST methodology for penetration testing and ISO 27000 standard for safeguarding information | 134 |
| 4.69 Resume of proposed methodologies to teach computer security. | 135 |
| 4.70 General structure to present the resolution of the challenge posed. | 135 |

| | |
|--|-----|
| 4.71 HackMyself - Proposed methodology to motivate learning about cookie theft and XSS attacks. | 136 |
| 4.72 Summary of the grades obtained in each partial and the average grades of the challenge solved | 137 |
| 4.73 Presentation of the solved challenge in dashboard format, average rating 19/20. | 138 |
| 4.74 Presentation of the solved challenge in a dashboard format, including a list of cookies found by the browser. | 138 |
| 5.1 Preview of our graphical interface to apply the BOOKIE framework in real scenarios | 144 |
| 5.2 Options menu to join browsing history and cookie databases | 144 |
| 5.3 Menú de Opciones para mostrar las estadísticas de las bases de historiales de navegación y cookies | 145 |
| 5.4 Options Menu to display statistics from browsing history and cookie databases | 145 |
| 5.5 Preview of the statistics generated by the cookie databases | 146 |
| 5.6 Preview of web browsing domain statistics before-after being normalized . . . | 146 |
| 5.7 Preview of cookie domain statistics before-after being normalized | 147 |
| 5.8 BOOKIE browser preview to demonstrate data filtering through cookies . . . | 147 |

List of Tables

| | | |
|------|---|----|
| 1.1 | Published Scientific Contributions | 15 |
| 2.1 | Rating of Insecure Interaction Category | 24 |
| 3.1 | Analysis and discussion of the tools and methods found. (1/10) | 35 |
| 3.2 | Analysis and discussion of the tools and methods found (2/10) | 36 |
| 3.3 | Analysis and discussion of the tools and methods found (3/10) | 37 |
| 3.4 | Analysis and discussion of the tools and methods found (4/10) | 38 |
| 3.5 | Analysis and discussion of the tools and methods found (5/10) | 39 |
| 3.6 | Analysis and discussion of the tools and methods found (6/10) | 40 |
| 3.7 | Analysis and discussion of the tools and methods found (7/10) | 41 |
| 3.8 | Analysis and discussion of the tools and methods found (8/10) | 42 |
| 3.9 | Analysis and discussion of the tools and methods found (9/10) | 43 |
| 3.10 | Analysis and discussion of the tools and methods found (10/10) | 44 |
| 3.11 | Search Strings proposed to find and filter the articles | 50 |
| 3.12 | Results after executing the search strings | 51 |
| 3.13 | Results after executing search using Snowballing | 52 |
| 3.14 | Research questions to establish the problem to be solved. | 55 |
| 3.15 | Top 5 most used methods and tools for detecting and mitigating XSS attacks from 2018 to 2023 | 59 |
| 3.16 | Summary of the analysis to find information related to cookies | 64 |
| 4.1 | Cookie information created when a user accesses the vulnerable web page . | 70 |
| 4.2 | Summary of packages sent/received in the attack (A) towards each victim (V) | 71 |
| 4.3 | Summary of ports used in the attack (A) on each victim (V) | 71 |
| 4.4 | Summary of packages sent/received in the attack (A) towards each victim (V) | 81 |
| 4.5 | Final DataSet.arff Attribute Description | 82 |

| | | |
|------|--|-----|
| 4.6 | Summary of results of classification tree algorithms | 83 |
| 4.7 | WEKA Decision Tree Algorithms Output (YES Class) | 84 |
| 4.8 | WEKA Decision Tree Algorithms Output (NO Class) | 84 |
| 4.9 | Content of the cookie that will be created on user's computer when visiting the compromised blog. | 93 |
| 4.10 | Summary of contacts who visited the blog | 95 |
| 4.11 | Questions posed in the survey to analyze the level of security of the devices. | 99 |
| 4.12 | Configuring the BarCodeScanner and WebViewer Components properties for QR Reader app | 101 |
| 4.13 | Summary of results of the online pentest tools used | 104 |
| 4.14 | Summary of results of the online XSS scan testing tools used | 105 |
| 4.15 | Questions included in the proposed survey | 108 |
| 4.16 | Challenge proposed or the students | 109 |
| 4.17 | Answers to the questions proposed in the Cybersecurity Challenge | 109 |
| 4.18 | Location of database containing web search history for Chrome, Firefox, and Breve browsers | 115 |
| 4.19 | Evaluation summary of programs used to capture and analyze network traffic | 118 |
| 4.20 | Evaluation summary of RPA programs used to test the programming of bots . | 119 |
| 4.21 | Evaluation summary of programs used to capture/analyze cookies | 120 |
| 4.22 | Location of the database that contains the cookies for each web browsers used | 125 |
| 4.23 | Hyperparameter analysis for the KMEANS algorithms | 129 |
| 4.24 | Performance tests with different eps values and min_samples | 131 |
| 4.25 | Performance test for Hierarchical Clustering | 133 |
| 4.26 | Rubric used to evaluate the challenge results | 138 |
| 4.27 | Linear regression coefficients | 139 |
| 4.28 | Result of the analysis of the hypotheses using the t-Student method | 140 |

RESUMEN

Esta tesis doctoral explora varias áreas clave relacionadas con la seguridad web, centrándose en el filtrado de datos personales mediante el robo de cookies via ataques XSS. Un análisis exhaustivo realizado entre 2013 y 2023 reveló tendencias notables en la lucha contra los ataques XSS, incluido el uso de herramientas específicas para navegadores web y el análisis de contenido de las páginas web para identificar vulnerabilidades. Además, se exploraron métodos de IA para clasificar páginas web y mitigar ataques XSS. El estudio profundizó en determinar la vulnerabilidad de las cookies frente a ataques XSS, dando lugar al desarrollo de un framework al que hemos denominado BOOKIE. Los resultados indicaron que un porcentaje significativo de las cookies creadas tienen propiedades que permiten la ejecución de comandos, lo que resalta posibles riesgos de seguridad. Además, el análisis reveló deficiencias en las propiedades de las cookies. Otro aspecto crucial de esta investigación fue el establecimiento de una relación entre el historial de navegación y los atributos de las cookies para desarrollar un clasificador de cookies basado en arboles de decisión. Este modelo ayuda a clasificar nuevas cookies en función de atributos sospechosos, mejorando las medidas de seguridad. Además, se evaluó la viabilidad de utilizar herramientas de web scraping para recopilar cookies de terceros. El estudio concluyó que las técnicas de simulación que imitan el comportamiento del usuario durante la navegación web permitieron la generación automática de las cookies necesarias, optimizando la captura de datos y la funcionalidad del sistema. Por último, el estudio evaluó el impacto de los ataques XSS en la fuga de datos personales y la competencia de los estudiantes en el descifrado de cookies. Los hallazgos enfatizaron la necesidad de una educación continua en ciberseguridad y una aplicación práctica para abordar las amenazas en evolución de manera efectiva. En conclusión, esta tesis contribuye significativamente a la comprensión y mitigación de las amenazas a la seguridad web, construyendo un camino para mejorar las medidas de ciberseguridad y los marcos educativos para abordar los riesgos emergentes.

Palabras Claves - Cookies, XSS, CookieScout, DataCookie, XSStudent, Bookie

ABSTRACT

This doctoral thesis explores several critical areas related to web security, focusing on leaking personal data through cookie theft via XSS attacks. A comprehensive analysis conducted between 2013 and 2023 revealed notable trends in the fight against XSS attacks, including web browser-specific tools and content analysis of web pages to identify vulnerabilities. Additionally, AI methods were explored to classify web pages and mitigate XSS attacks. The study delved into determining cookies' vulnerability to XSS attacks, leading to the development of a framework we have called BOOKIE. The results indicated that a significant percentage of the cookies created have properties that allow the execution of commands, highlighting potential security risks. Furthermore, the analysis revealed deficiencies in the cookies' properties. Another crucial aspect of this research was establishing a relationship between browsing history and cookie attributes to develop a cookie classifier based on decision trees. This model helps classify new cookies based on suspicious attributes, improving security measures. The feasibility of using web scraping tools to collect third-party cookies was also evaluated. The study concluded that simulation techniques that imitate user behavior during web browsing allowed the automatic generation of necessary cookies, optimizing data capture and system functionality. Finally, the study evaluated the impact of XSS attacks on personal data leakage and students' proficiency in cookie decryption. The findings emphasized the need for continued cybersecurity education and practical application to address evolving threats effectively. In conclusion, this thesis contributes significantly to understanding and mitigating web security threats, building the way for improving cybersecurity measures and educational frameworks to address emerging risks.

Keywords - Cookies, XSS, CookieScout, DataCookie, XSSStudent, XSS2dent, Bookie

PROLOGUE

"Yes, I am a criminal. My crime is curiosity. My crime is being smarter than you, something you will never forgive me for. I'm a hacker, and this is my manifesto. You can stop me, but you can't stop them all... After all, we are all equal."

The Mentor (Loyd Blankenship).

This work is titled "*BOOKIE: Cookie theft through XSS attacks and its relationship with the leakage of personal data: a framework to demonstrate it to university students*". The basis of this research is computer security, which was developed at the Universidad de las Fuerzas Armadas-ESPE sede Santo Domingo. This work has been written as part of the graduation requirements for the Doctorate in Computer Science program at the National Polytechnic School. The research and writing period for this work has lasted since 2016.

It is supported by a well-founded theoretical/experimental base and is structured in four large blocks. The first is a Background that describes the concept of what cookies are, their attributes, their vulnerabilities, the most common type of cookies, and the policies for using these files generated on our devices every time we browse the Internet. The concepts of XSS attacks are also explained, and some scenarios have been proposed to demonstrate their "modus operandi". Finally, it explains the relationship between the theft of cookies through XSS attacks and what has been done to remedy this vulnerability.

The second block has been divided into two parts. The first part explains, through a survey of scientific articles analyzed from 2013 to 2018, the tools and methods most used to detect/mitigate XSS attacks. In the second part, a systematic literature review covering scientific articles from 2018 to 2023 answered some research questions that served to put our research in context and find and confirm the research gap related to the cookies and XSS attacks.

The third block brings together all the scientific contributions published over these years and has been organized as a road map to give a vision of how the idea of the proposed

framework was born, which we have named Bookie. It is a personalized web browser that was trained with history data and cookies obtained from the computer laboratories of the Universidad de las Fuerzas Armadas-ESPE to predict students' vulnerability level towards cookie theft through XSS attacks.

Finally, the fourth section details the experimentation and results obtained after developing and sharing the framework with University students.

The entire research process was complex, but conducting an exhaustive study has allowed me to answer one of the most critical research questions: "Is it possible to compromise all the privacy of university students through the theft of their cookies through XSS attacks?". This answer was given thanks to the support of my tutor and thesis director, Dr. Jenny Torres Olmedo, who was always available and willing to help me with all my questions.

The development of this work has been an enriching and challenging journey. From an exhaustive literature review to data collection and analysis, each stage has involved deep and meaningful learning. I hope that the findings of this thesis contribute to the advancement of computer security and offer valuable tools for professionals in the area.

With this thesis, a crucial stage of my academic life concludes. I look to the future with the hope of continuing to contribute to knowledge in this field and applying what I have learned to benefit society and students.

Chapter 1

Introduction

Índice

| | | |
|-------|---|----|
| 1.1 | Research Context | 1 |
| 1.2 | Justification | 4 |
| 1.3 | Research questions and objectives | 6 |
| 1.3.1 | General objective | 6 |
| 1.3.2 | Specific objectives | 6 |
| 1.3.3 | Research Questions(RQ) | 7 |
| 1.4 | Research methodology and approach | 8 |
| 1.4.1 | Phase A: Characterization | 9 |
| 1.4.2 | Phase B: Replication and Collect: | 11 |
| 1.4.3 | Phase C: Training and Predict | 11 |
| 1.4.4 | Phase D: Visualization | 12 |
| 1.5 | Motivation | 12 |
| 1.6 | Scientific Contribution | 14 |

1.1. Research Context

When we browse the Internet, we often notice a message when visiting a website for the first time. The message typically states: *"We use our own and third-party cookies to improve our services by analyzing your browsing habits. If you continue to browse, we will consider this as your acceptance of our use of cookies"*.

A cookie is a file created by a website that contains small amounts of data, is stored on the devices of users who access the website, and can later be retrieved remotely. Being a

small file, it can store information about the web pages and advertisements that we have visited, actions that we perform on the website, sub-pages that we visit, how long we stay on them, what type of products we purchase, browsing habits, geographic location, viewing language, time zone, whether we have provided our e-mail address, the type of browser used, the type of device from which we have accessed, etc.

Generally, cookies are linked to a domain. In theory, a cookie can only be read or modified by a website coming from a similar domain (first party cookies), i.e. a cookie placed by the advertising provider a.mysite.com can be read by the domain b.mysite.com but not by the advertising provider c.anothersyte.com. However, there is the so-called remarketing [1], which breaks this theory. Imagine a web page containing advertising, the provider of this advertising places a tracking cookie on the user's terminal the first time the user visits an Internet site displaying an advertisement from its advertising network.

From this action is derived the concept of behavioral advertising [2], which is based on monitoring browsing history using tracking cookies to segment users based on their specific interests and send them advertising that is presumed of interest. Because these tracking cookies are placed by a third party other than the web server that displays the site's main content (i.e., the publisher), they are known as third-party cookies.

The issue arises due to the fact that, in addition to the "legitimate uses" that websites employ cookies for, they are at times exploited by malicious actors to carry out specific illicit activities such as:

- Stealing session cookies (credentials) through software failures or vulnerabilities in the protocols used by browsers.
- Redirection to fraudulent stores, since third-party cookies record all the searches we perform, fraud occurs when, using this data stored in the cookies, the user is redirected to fraudulent stores, through misleading advertising (Retrieval of information from cookies).
- Presentation of fake news, aimed at promoting biased or sensationalist articles or videos.
- Session hijacking, when a modified cookie is placed in the browser of a user who has previously accessed a website controlled by cyber criminals.

In the above cases, the victim does not know that personal data is transferred from a trusted website to a different site controlled by a malicious user [3]. Thus, third-party

cookies [4] are deliberately used for web tracking, user analysis, and online advertising, with the consequent loss of privacy for end users.

Web applications must use their cookies and one or more third-party cookies to improve and enhance web services by analyzing browsing habits [5]. They also use cookies to maintain the user's session state. Subsequently, no additional verification is required for connection, which makes cookies more vulnerable to attackers.

There are many attacks to execute a cookie theft, but the most powerful is one called Cross-Site Scripting (XSS). It occurs when malicious code is sent or executed on a website, usually scripts. This execution can leak personal information and steal the user's cookies to hijack the identity in a fraudulent session, thus allowing attackers to steal sensitive data or even take control of specific devices.

According to 2021 OWASP Top 10 Security Risks Vulnerabilities [6], this attack is ranked 3rd. The main security issues resulting from an XSS attack are:

- User session hijacking
- Web site defacement
- Redirecting the user to malicious sites
- Keystroke capture
- Denial of service attacks
- Corporate network scans
- Phishing attacks or geolocation of a user.

Based on this study [7], it is claimed that a cookie can be stolen, and the user's entire privacy can be violated. Although cookies are not the only mechanism that servers can use to track users through HTTP requests, cookies facilitate tracking because most cookies are persistent across user agent sessions (typically web browsers, which request one or more resources typically through HTTP or HTTPS, such as HTML documents, CSS style-sheets, and scripts) and can be shared between hosts.

These third-party servers may use cookies to track the user even if the user has never visited the server directly. For example, if a user visits a site that contains content from a third party and then visits another site that contains content from the same third party, the third party may track the user between the two sites.

While cookie hijacking is not a new attack vector, it can still affect even the most popular websites (social networks) and expose users to significant threats, including full account takeover. It is possible to think that many third parties use tracking cookies, placing them on different web pages, but the truth is that most of these types of cookies come from the same companies. As shown by a study conducted by Princeton University [8]; Google, Facebook and Twitter are present in more than 10% of the million websites analyzed.

There are 19586 out of 157703 records (12.29%) for vulnerabilities related to XSS and Cookies in Common Vulnerabilities and Exposures (CVE) [9]. This type of attack directly affects all users, who are unaware of what is happening when, for example, they access a web page that requires authentication (email, social networks, banking, shopping, online transactions, etc.). A stolen and modified cookie can be passed off as legitimate to obtain the user's credentials.

From a negative ethical point of view, our online world can have similar implications to having a persistent observer who haunts us in the physical world, continually monitoring various aspects of our daily lives, such as our mode of transportation, consumption behavior, eating habits, residential location, education and occupations, and cultural inclinations, among others. This process implies the application of a monitoring mechanism that captures and segments the users' activity, making it possible to predict their preferences and interests.

Unfortunately, when we browse the Internet, we ignore our privacy and allow it to be significantly abused [10] without raising any objection. This makes it easier for advertisers to track users' activities on the website using third-party cookies.

1.2. Justification

This section presents the justification of the present study, highlighting its relevance and need to address the proposed research problem [11]. It will explain the theoretical, practical, and methodological reasons underlying the importance of this work.

Theoretical Relevance

The proposed research contributes to computer security by offering new insights into leaking personal data by stealing cookies using XSS attacks. Despite existing advances, more literature still needs to be on cookie theft and XSS attacks. This study seeks to fill these gaps by:

- Providing a detailed analysis of the methods or tools that have been proposed to mitigate this type of attack.
- Developing a new theoretical/methodological framework to highlight the research gap related to research.
- Comparing and contrasting existing methods and tools with new empirical evidence, this study aims to provide practical insights that can be used to effectively mitigate cookie theft through XSS attacks.

Practical Importance

In addition to the theoretical contribution, this research has significant practical relevance. The findings of this study may influence:

- Implementing policies in the education sector to improve the protection of data from university students.
- Improving professional practices in the area of knowledge related to computer security.
- Designing programs to reduce the level of vulnerability that students have against this type of attacks.
- Applying the results to demonstrate how the personal data of university students can be leaked by stealing their cookies.

Social context

The research also addresses a critical educational need. Protecting personal data among university students is a significant concern in today's digital age. The increasing use of technology in education has made it imperative for institutions to implement robust protective measures to safeguard student data privacy, as the unauthorized disclosure of sensitive data can lead to identity theft and other cyber-crimes.

This study will help us better understand this social problem and propose a solution for teaching students about personal data leaking.

Originality and Innovation

The present study is distinguished by its originality in several aspects. Firstly, it uses a method to detect XSS vulnerabilities in web pages by analyzing the attributes of cookies;

this method has yet to be widely explored in the existing literature. Secondly, it focuses on university students and the care of their data, an issue that needs to be considered in previous research. This will allow:

- Generate new knowledge about filtering personal data by stealing cookies through XSS attacks. Furthermore, we put forth theoretical and methodological innovations that have the potential to revolutionize future research in the field of data security and privacy.
- These innovations, if adopted, can inspire a new wave of exploration and understanding in our academic community.

Methodological Justification

The methodology used in this study is carefully designed to address the research questions rigorously and validly. A qualitative, quantitative, and experimental methodology will be used, which will allow:

- Obtain robust and reliable data on cookies, XSS attacks, and cookie theft using XSS attacks.
- Conduct a comprehensive and detailed analysis that supports the conclusions of the study.

1.3. Research questions and objectives

1.3.1. General objective

- Demonstrate to university students, through a framework, the relationship between the theft of cookies through XSS attacks and personal data leakage.

1.3.2. Specific objectives

- Explore and determine the current state of the methods and tools that have been proposed to mitigate XSS attacks.
- Determine the level of vulnerability that cookies have against Cross-Site Scripting (XSS) attacks.

- Establish the relationship between browsing history and cookie attributes in order to develop a decision tree-based cookie classifier.
- Determine the level of difficulty experienced by university students to identify cookies that transmit their personal information to third parties.
- Determine the feasibility of using web scraping tools to gather third-party cookies.
- Evaluate the feasibility of using clustering techniques and a multi-agent bot system to collect cookies and determine browsing preferences, with the goal of discovering cookie content.
- Illustrate the impact of cookie theft using XSS attacks on the risk of personal data leakage by designing an educational framework to effectively demonstrate and address this threat for university students.

1.3.3. Research Questions(RQ)

- **RQ1:** ¿ What is the level of vulnerability of cookies against Cross-Site Scripting attacks?
- **RQ2:** ¿What is the relationship between browsing history and cookie attributes to model a cookie classifier using decision trees?
- **RQ3:** ¿Is it possible to compromise the privacy of university students through the theft of their cookies?
- **RQ4:** ¿ What is the level of difficulty that university students have to identify the cookies that send their personal information to third-parties?
- **RQ5:** ¿Is it possible to collect third-party cookies using web scraping tools?
- **RQ6:** ¿ Is it possible to decrypt the content of cookies using clustering techniques and a multi-agent bot system to collect cookies and define browsing preferences?
- **RQ7:**¿How does the exploitation of XSS attacks in the context of cookie theft contribute to the risk of personal data leakage, and what educational framework can be designed to demonstrate and address this threat for university students?

1.4. Research methodology and approach

Figure 1.1 shows our research timeline, which was born with the idea of analyzing user cookies when the author of this thesis worked at the National Institute of Statistics and Censuses (2016 to 2018). As a result of this curiosity, two scientific articles were published that served as a basis for publishing the other scientific contributions during the author's work time at the University of the Armed Forces ESPE, Santo Domingo (2018-2024).



Figure 1.1: Timeline and areas of application of our research proposal.

The research focuses on a relevant challenge within the field of computer security related to filtering personal data through the theft of cookies via Cross-Site Scripting attacks. On the other hand, the digital security of university students is an essential issue because they are exposed to various online threats.

It is well known that cookies are files created on users' computers who browse the Internet. However, the challenge is to analyze characteristics such as their behavior because they are stored for a long time and the type of information they handle and share with third parties.

Figure 1.2 shows the structure of our methodological proposal, divided into four essential phases:

- Phase A. Characterization (4 sub phases)
- Phase B. Replication and Collect
- Phase C. Training and Prediction (2 sub phases)
- Phase D. Visualization

Furthermore, Figure 1.2 illustrates the alignment of each phase with the scientific contributions stemming from the entire research process. To reduce the need for writing the full title, each contribution has been assigned an alias. In summary, for Phase A of our framework, the contribution is aligned with the following articles: "CookieScout" [12], "DataCookie"

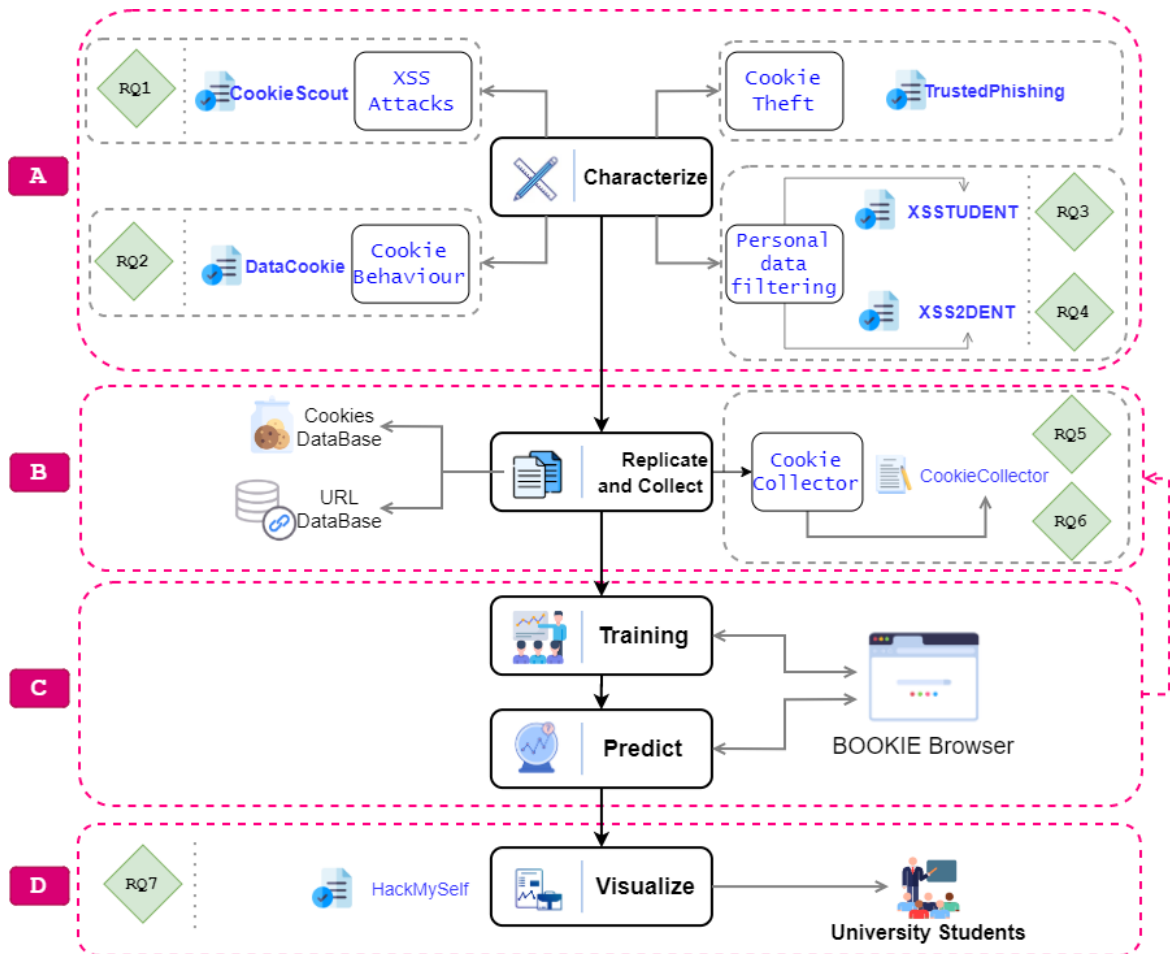


Figure 1.2: Methodological proposal to show university students how stealing cookies is related to personal data filtering.

[13], "TrustedPhishing" [14], "XSSTUDENT" [15], and "XSS2DENT" [16]. On the other hand, Phase D is aligned with an ongoing approved article titled "HackMySelf" [17].

1.4.1. Phase A: Characterization

In the first phase of the research, the critical components of our proposal were identified: XSS Attacks, Cookie Behavior, Cookie Theft, and Filtering of personal data.

Sub phase 1 - XSS Attacks

An experimental approach was taken to demonstrate how XSS attacks work and evaluate an effective mitigation strategy. A secure test environment was set up to simulate a reflected and stored XSS attack. The results of this experiment were recorded, allowing a comprehensive analysis of the attributes of the cookies created when a victim visits a

website with XSS vulnerabilities.

The contribution of this phase is reflected in a published scientific article entitled "*CookieScout*" [12], which made it possible to answer the first research question posed (RQ1). The contribution is an algorithm that allows the analysis of cookies' attributes to identify those susceptible to XSS attacks.

Sub phase 2 - Cookie Behaviour

An experimental approach was also adopted to apply the highly effective proposed Phase 1 algorithm and analyze a base of web browsing histories identified by category (according to the logs of the security equipment used to extract the information). A dataset was structured to train tree classification algorithms to predict new domains vulnerable to XSS attacks. In this phase, a Python script was also programmed to eliminate vulnerable cookie records and thus reduce the risk of a user's vulnerability to XSS attacks.

The contribution of this phase is reflected in a published scientific paper entitled "*Data-Cookie*" [13]. It includes a tree algorithm that predicts if a given domain is vulnerable to XSS attacks and, if so, deletes the corresponding cookie(s).

Sub phase 3 - Cookie Theft

Through a qualitative and quantitative approach, we have relied on data collection and analysis to identify vulnerability patterns of our contacts from 2 networks: instant messaging (WhatsApp) and social (Facebook). Through a phishing attack using QR codes and posted messages, we played on the contacts' curiosity to access a blog that had been compromised with an XSS script that allowed us to steal a cookie created by the blog itself. This attack aimed to demonstrate the theft of cookies through XSS attacks using vulnerable pages.

The contribution of this phase is reflected in a published scientific paper entitled "*TrustedPhishing*" [14], which served as the basis for answering the research questions for the following phases. The contribution is an algorithm that detects whether QR codes redirecting to a URL are malicious and have XSS vulnerabilities.

Sub phase 4 - Personal Data Filtering

A qualitative and quantitative approach was also applied to demonstrate that students at the Universidad de las Fuerzas Armadas are vulnerable to XSS attacks. An improved version of this proposal was executed city-wide in Santo Domingo de los Tsachilas to discover if other universities or educational institutions are vulnerable to this type of attack.

In this phase, we published two articles, "*XSSTUDENT*" [15] and "*XSS2DENT*" [16], which addressed research questions RQ3 and RQ4.

1.4.2. Phase B: Replication and Collect:

A complete system for collecting cookies was assembled with the characterization of each component of our research. In this part of the research, we have proposed a system that allows the automatic collection of cookies through a set of bots that simulate a user's browsing behavior. Five bots were programmed, one for each browser most used by university students (Chrome, Firefox, Edge, Opera, and Brave).

Through an experimental approach, we set up a scenario using virtual machines to isolate web browsing from bots. Each bot was fed with a base of web search histories extracted from the Armed Forces University computer labs. With this initial base, the bots replicated the search and recorded the cookies generated by each URL.

This section's contribution is based on a system of bots that allows them to collect cookies. By analyzing their attributes, they can classify them as suspicious or not for XSS attacks (based on the algorithm presented in phase one). From this analysis, the system extract the domains to which these cookies belong and add them to their knowledge base to analyze new domains.

1.4.3. Phase C: Training and Predict

The Training and Prediction phase aims to develop and evaluate machine learning models to classify web domains as suspicious and predict new domains' vulnerability to XSS attacks. Using a dataset of web browsing history records associated with generated cookies, this phase focuses on training classification tree models and then applying clustering techniques to identify user vulnerability patterns.

The dataset used in this phase was collected by the Phase 3 bot system, which has collected detailed information from cookies and web browsing domains. Each record in the dataset includes attributes such as:

- Domains of visited URLs (first-party)
- Cookies generated by visited domains
- Network traffic generated

- Third-party or advertising-generated domains (third-party)
- Classification of domains according to the attributes of the associated cookies (suspicious/not suspicious).

The first stage of training uses the same decision tree algorithms used in sub phase 2 of phase 1 of our model. Subsequently, clustering algorithms were applied to group users with similar browsing characteristics and assessed their vulnerability to XSS attacks. Once the clusters were defined, the clustering models were used to predict the vulnerability of new users. By analyzing their browsing histories and cookie attributes, each user was assigned a level of risk when vulnerable to XSS attacks.

In this phase, we are combining decision tree algorithms and clustering techniques to develop a predictive system. This system will not only classify suspicious domains, but also identify and predict the vulnerability of new users to cross-site scripting (XSS) attacks. We created a framework called **"BOOKIE" (Own web browser to explore cookies)**, which allowed us to assess the level of vulnerability of students and strengthen web security measures through teaching.

1.4.4. Phase D: Visualization

This last phase aims to demonstrate to university students the level of vulnerability to XSS attacks to which they are exposed. Through a computer security challenge, they were asked to replicate the working model of the BOOKIE browser but on their personal computers. Two groups of students from different academic periods had an entire semester to develop their framework to display, through a dashboard, the personal cookie information that each student had stored on their computers. This challenge, designed with respect for personal privacy, ensured that the personal information obtained by each student was treated with the highest level of confidentiality. The scientific contribution of this phase was reflected in a scientific article called *"HackMySelf"* [17] and also allowed us to answer the last research question posed (RQ7).

1.5. Motivation

The motivation for conducting this study stems from curiosity about the types of data that cookies store on our computers every time we browse the Internet. This research addresses a personal and professional interest in computer security and the need to address the issue

of filtering personal information to prevent cookie theft through XSS attacks in the educational context of university students.

Data privacy on the Internet is a significant issue due to the amount of personal information shared online. With the increasing use of network-connected devices and mass data collection [18], there is growing concern about how personal data is handled and protected. The right to privacy of our data means that we must have the ability to decide how and to what extent we can share or disclose our personal information to third parties.

Academic Motivation

The academic motivation behind this research lies in the opportunity to contribute to the existing knowledge of computer security. Despite the numerous studies conducted, unanswered questions and areas need to be explored further. Our objective are:

- Provide not just new perspectives, but groundbreaking insights on the filtering of personal data through the theft of cookies via XSS attacks, a topic that has not been extensively explored.
- Develop a methodological framework that future researchers can use.
- Publish results that can be applied in the educational context and relate to university students.

Social Motivation

Another crucial motivational factor is the potential impact that this research can have on society. Specifically with the community of university students by offering a framework that demonstrates how their personal information can be filtered through the theft of cookies. Through this research, it is expected:

- It is crucial to raise immediate awareness among university students about the filtering of personal data through the theft of cookies via XSS attacks.
- Propose viable solutions that can be implemented to improve the experience of university students when browsing the Internet, solving the problem of data filtering.
- Contribute to the well-being of university students who do not know computer security concepts and personal data filtering.

Intellectual Curiosity

Intellectual curiosity also plays a fundamental role in our motivation. The complexity and multi-dimensionality of filtering personal data through the theft of cookies via XSS attacks has presented a significant challenge that has been addressed throughout this work's development. Researching how cookies behave will satisfy our desire for knowledge and enrich the field with new findings and innovative approaches.

1.6. Scientific Contribution

Our research has generated several significant contributions to computer security, specifically in identifying and mitigating Cross-Site Scripting (XSS) attacks by analyzing the attributes of cookies. Table 1.1 shows the information from our published scientific articles.

Table 1.1: Published Scientific Contributions

| Ref. | Title | Impact | Type |
|----------|---|--------|----------------------------------|
| [12] | Cookie scout: An analytic model for prevention of cross-site scripting (XSS) using a cookie classifier | Q4 | Conference Paper |
| [13] | DataCookie: Sorting Cookies Using Data Mining for Prevention of Cross-Site Scripting (XSS) | - | Book Chapter |
| [14] | Trusted Phishing: A Model to Teach Computer Security Through the Theft of Cookies | Q4 | Conference Paper |
| [15] | XSSStudent: Proposal to Avoid Cross-Site Scripting (XSS) Attacks in Universities | Q4 | Conference Paper |
| [16] | XSS2DENT, Detecting Cross-Site Scripting Attacks (XSS) Vulnerabilities: A Case Study | Q4 | Conference Paper |
| Accepted | HackMySelf: HackMySelf: Decrypting Cookies to Show the Theft of Personal Data in University Students | Q4 | Conference Paper |
| [19] | Cross-site scripting (XSS) attacks and mitigation: A survey | Q1 | Journal |
| [20] | Personal data filtering: a systematic literature review comparing the effectiveness of XSS attacks in web applications vs cookie stealing | Q2 | Journal |
| [21] | Diseño e Implementación de BOTS para Automatizar Tareas de Búsqueda y Análisis de Vulnerabilidades en Sistemas Web. | - | Undergraduate Thesis (Co-Author) |
| [22] | Vulnerability of CAPTCHA Systems Using Bots with Computer Vision Abilities. | Q4 | Conference Paper (Co-Author) |

Chapter 2

Background

Índice

| | | |
|-------|--|----|
| 2.1 | COOKIES | 16 |
| 2.1.1 | ¿What are cookies? | 16 |
| 2.1.2 | Cookie attributes: | 17 |
| 2.1.3 | Cookie vulnerabilities | 19 |
| 2.1.4 | Most common types of cookies | 20 |
| 2.1.5 | Policies for the use of cookies: | 21 |
| 2.2 | CROSS-SITE SCRIPTING (XSS) | 22 |
| 2.2.1 | Types of XSS attacks and exploitation examples | 24 |
| 2.2.2 | Common scenarios for executing XSS attacks | 27 |
| 2.3 | COOKIES THEFT THROUGH XSS ATTACKS | 30 |
| 2.3.1 | Background of XSS attacks and theft of cookies | 30 |
| 2.3.2 | Statistics to remedy cookie theft: | 30 |

2.1. COOKIES

2.1.1. ¿What are cookies?

Cookies are a tool commonly used in web applications and browsers. They allow the capture and storage of information transmitted during sequential communication, providing continuity and state across hypertext transfer protocol (HTTP) connections [23]. A cookie is a file sent by a web server containing end-user information. Through their use, cookies can overcome the statelessness of the HTTP protocol by storing information on the client side.

A session code is created by a server and transmitted to a client to facilitate the current interaction. This code is then saved in a cookie [24], which is a tiny text file that identifies a specific client. However, since cookies are transmitted over HTTP, they become susceptible to attacks like session hijacking.

Ensuring secure cookies is a crucial aspect of web development. One of the most widespread techniques is utilizing hypertext transfer protocol secure (HTTPS). Nevertheless, implementing full support for HTTPS can pose significant challenges, especially for distributed applications. These issues primarily relate to performance and financial considerations, making it a complex task that requires careful planning and execution.

2.1.2. Cookie attributes:

Secure Attribute:

The Secure attribute instructs the browser to send the cookie only if the request is sent over a secure channel like HTTPS [6], protecting it from being passed in unencrypted requests. If the application can be accessed over HTTP and HTTPS, an attacker could redirect the user to send their cookie as part of non-protected requests.

- *true or 1*: This value is essential to protect the cookie information during transmission, preventing it from being intercepted by third parties through man-in-the-middle attacks.
- *false or 0*: The cookie can be sent over secure (HTTPS) and unsecured (HTTP) connections. Although it allows for greater flexibility, it could be more insecure since the cookie information could be intercepted if transmitted over an unsecured connection.

HttpOnly Attribute:

The HttpOnly attribute prevents the cookie from being accessed via client-side scripts like JavaScript, reducing the reach of XSS attack vectors.

- *true or 1*: The cookie is marked HttpOnly, making it inaccessible via JavaScript in the browser. It can only be sent in HTTP(S) requests from the client to the server, improving its security and preventing malicious scripts from accessing its contents in the event of an XSS attack.
- *false or 0*: The cookie is not marked HttpOnly, which means it is accessible via JavaScript in the browser. According to the literature, it could be more secure and

may be necessary for some functionality that requires access to cookies from client-side scripts.

Domain Attribute:

The Domain attribute of a cookie is compared to the server's domain. If the domain matches or is a subdomain, the path attribute is checked next.

It's crucial to understand that only hosts belonging to a specified domain can set a cookie for that domain [6]. This is a security measure to prevent servers from setting arbitrary cookies for another domain. If the domain attribute is not set, then the hostname of the server that generated the cookie is used as the default value of the domain.

Path Attribute:

The Path attribute plays a significant role in setting the scope of the cookies in conjunction with the domain. In addition to the domain, the URL path for which the cookie is valid can be specified. If the domain and path match, the cookie will be sent in the request.

As with the domain attribute, if the path attribute is set too loosely, it could leave the application vulnerable to attacks by other applications on the same server. For example, suppose the path attribute was set to the web server root /. In that case, the application cookies will be sent to every application within the same domain (if multiple applications reside under the same server).

Expires Attribute

The Expires attribute is used to:

- set persistent cookies
- limit lifespan if a session lives for too long
- remove a cookie forcefully by setting it to a past date

The browser uses persistent cookies until they expire, and deletes the cookie once the expiration date has passed.

SameSite Attribute

El atributo SameSite de las cookies en algunos navegadores web, se utiliza para mejorar la seguridad y la privacidad de las cookies al restringir cuándo se envían junto con las

solicitudes de origen cruzado. Los valores que puede tener el atributo SameSite y sus significados son los siguientes:

- *SameSite=Strict* (*value = 1*): This value is the most restrictive and provides the highest level of security, preventing the cookie from being sent in cross-origin requests.
- *SameSite=Lax* (*value = 0*): The cookie is sent in requests originating from the same site and in some cross-origin requests, such as those initiated by the user's navigation (for example, clicking on a link to the site).
- *SameSite=None* (*value = -1*): This value is the least restrictive and allows the cookie to be sent on all requests, which may be necessary for cases such as cross-site resource usage.
- *Without SameSite attribute*: The absence of the attribute is considered less secure and may result in inconsistent behavior between browsers.

2.1.3. Cookie vulnerabilities

Using cookies to profile individuals has raised significant concerns regarding privacy protection [25]. Cookies can be used to track a user's online behavior, such as browsing history and search queries, and to collect personal information, such as name, email, and phone number. Such information can be used for targeted advertising, behavioral analysis, or other use. Consequently, governments and businesses have instituted privacy protection laws to establish a legal framework that mitigates these concerns and protects individual privacy [26].

Cookies are vulnerable to unauthorized access, modification, and phishing attacks due to their easy storage and transfer in text form [27]. Web-based applications frequently utilize cookies to maintain the status of an ongoing web session. However, cookies may result in security threats as they may contain sensitive information. In addition, a hacker who gains access to a cookie can impersonate an authorized user [28], further exacerbating potential security issues.

To protect privacy, users block HTTP cookies. However, indiscriminate cookie blocking can affect critical web services and reduce the effectiveness of online advertising [29]. Blocking essential cookies can harm the reputation of websites and frustrate users. Employing cookie-blocking tools that allow necessary cookies to function is recommended.

Cookies can also be used to track a user's online activity, which poses a serious threat to privacy. By monitoring a user's behavior, cookies can collect personal information and other sensitive data, such as browsing history and login credentials [30]. This information can be used for targeted advertising or even sold to third-party companies without the user's consent or knowledge. Users need to be aware of the potential risks associated with cookies and take steps to protect their privacy online.

2.1.4. Most common types of cookies

First party cookie:

First-party cookies are small data websites store on a user's device, typically within the user's web browser. The website sets these cookies the user directly interacts with, hence the term "first-party." The primary purpose of first-party cookies is to enhance the user experience by enabling websites to remember user preferences, login information, and other relevant data.

Unlike third-party cookies, which are set by domains other than the one the user is actively visiting, first-party cookies are associated with the domain of the website directly accessed by the user. This distinction often places first-party cookies in a more favorable light from a privacy perspective, as they are typically considered essential for the proper functioning of websites and are subject to less scrutiny than third-party cookies.

While first-party cookies contribute significantly to a seamless and personalized online experience, concerns about user privacy and data security have prompted ongoing discussions and regulatory developments. Understanding the characteristics and implications of first-party cookies is crucial for enhancing user experiences and addressing privacy considerations in the digital landscape.

These are known as first-person/first-party cookies because they do not send data to other sites, offering some privacy, since only the host where the cookie was created can access the information it contains. To verify that it is a first-person cookie, the parameter that registers the domain of the host with the browser bar will be compared. If they are the same, it is a cookie of the first person. However, this does not guarantee that our information will be shared with third-parties through an attack that retrieves or hijacks our cookies.

The browser automatically sends third-party cookies to third parties in HTTP requests, while first-party cookies do not [31]. However, tracking through first-party cookies is still possible because any included third-party code runs in the context of the parent page and

can fully set or read existing first-party cookies. This action can result in the leakage of first-party cookies to the same or other third parties.

Third party cookie:

Internet advertisers reach millions of customers through practices that accurately track users' online activities. The tracking is conducted by third-party ad services engaged by websites to facilitate marketing campaigns. [32]

When browsing the Internet, cookies are stored on users' computers. Some of these cookies are from "third parties" that can track users across different websites [33].

The public and policymakers are increasingly aware that tracking cookies support behavioral advertising, but it is unclear to what extent tracking occurs [34].

First, there are essential cookies that are necessary to guarantee the website's functionality. Second, there are third-party ad-tracking cookies. A website's host still has a monetary incentive to nudge users to allow for the ad-tracking cookies. [35]

As users browse the internet, their activity is constantly monitored by entities that profit from collecting and analyzing data. This has significant implications for users' privacy. [36]

Session Cookies:

They are stored in the browser's memory and destroyed when closed; this is their most significant advantage. Their disadvantage is that they store information such as the login credentials of a session (for example, the e-mail), which could be stolen to obtain this sensitive data. A particular type of stateless session cookies allows web applications to modify their behavior based on the user's preferences and associated access rights, avoiding maintaining the server's status for each session [37].

2.1.5. Policies for the use of cookies:

Alert messages like: " This website uses cookies to ensure you have the best experience on our website (Figure 2.1)" or "Alert on the policy of use of cookies in browsers (Figure 2.2)".

The float messages appear on most visited web pages as a notification banners popups. However, it is worrying how this message is accepted with the following warning: *If you continue browsing, we consider that you accept its use.*

Política de cookies

Ricoh emplea herramientas de recopilación de datos, como cookies, para ofrecerle la mejor experiencia cuando use este sitio. Descubra cómo puede cambiar esta configuración y obtenga [más información sobre las cookies](#).

Figure 2.1: Alert on the policy of use of cookies in browsers

¡Bienvenido a [NIVEA.com.ec](#)! Utilizamos cookies para ofrecerte un sitio web lo más atractivo posible. Al seguir utilizando [NIVEA.com.ec](#) declaras que estás de acuerdo con el uso de cookies. [Cookies](#)

Figure 2.2: Welcome message and notice of the use of cookies

Although the user does not click the OK button, only navigating the site will install the cookies warned in the informative messages. This new mode of navigation makes web pages install cookies compulsorily on the user's computer visit. As mentioned in the previous sections, a successful XSS attack could steal the user's cookies and show an active session to pass as a legitimate user.

2.2. CROSS-SITE SCRIPTING (XSS)

Web applications are insecure by default. Their developers do not establish secure development protocols, which contributes to the theft of personal and crucial information from users [38]. This lack of good practices is considered a vulnerability. If the website is not developed correctly, a hacker uses this flaw to execute malicious code on the systems. In addition, it could scale through the entire organization's network. Nowadays, the most popular attack vector is a web browser due to the growth of Internet access. That is, most web applications have vulnerabilities in their source code, which increases the possibility of exploring their flaws and exploiting them.

A successful malicious attack on a victim can result in session manipulation on social networks, theft of cookies to steal identity, and control of the victim's browser. According to the latest security statistics report on web applications [39], the victim is the user, not the applications.

XSS ranks second as one of the most severe vulnerabilities, with approximately 38% critical. However, what is worrying is that this type of attack has a shallow solution and remediation rate. On the other hand, according to the ten main security risks of the Open Web Applications Security Project (OWASP) [40], the third place (2021) was occupied by the XSS attacks, compared to 2013, which occupied the seventh place.

In summary, XSS is evaluated using parameters such as exploitability, business impacts,

technical impacts, weakness detectability, and prevalence. Automated programs detect and exploit three types of attacks: persistent, non-persistent, and Document Object Model (DOM). Frameworks are easily accessible to exploit this vulnerability.

The prevalence and detectability of XSS vulnerabilities are analyzed because they are the third most frequent problem in the Top Ten OWASP reports (injection category). Programs can automatically find XSS vulnerabilities in PHP, J2EE/JSP, and ASP/.NET technologies.

According to the CWE/SANS [41] (See Figure 2.3), XSS is named Improper Neutralization of Input During Web Page Generation, identified as CWE-79, and is in second place among the Top 25 Software Errors. In addition, it has a classification according to the following parameters: weakness prevalence, remediation cost, attack frequency, consequences, ease of detection, and Attacker Awareness (see Table 2.1).

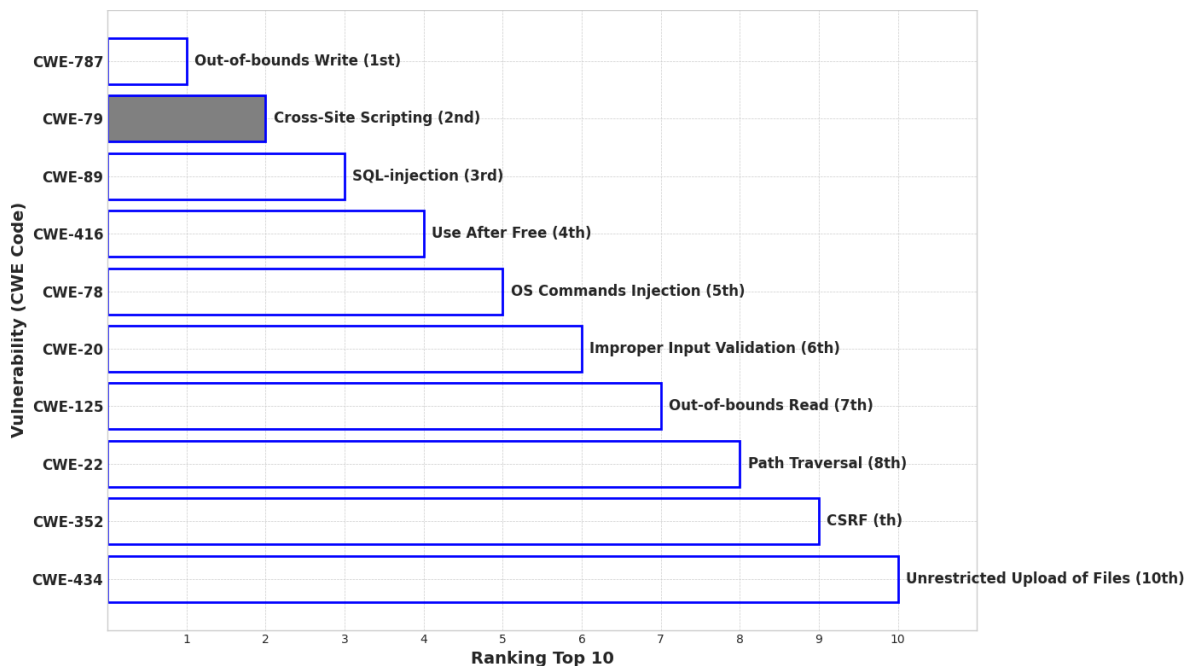


Figure 2.3: Top Ten Software Errors Ranking

XSS attacks often occurs when [42]:

- Untrusted data can be entered into a web application.
- The same web application dynamically generates this data that is not trusted.
- A victim visits a website through a web browser infected with a malicious XSS script.
- A script of type XSS sent by a server is executed in a web page, it means, in the same context of the domain of the web server.

Table 2.1: Rating of Insecure Interaction Category

| Parameters | Qualification |
|---------------------|---------------------------------|
| Weakness Prevalence | High |
| Remediation Cost | Low |
| Attack Frequency | Often |
| Consequences | Code execution, Security bypass |
| Ease of Detection | Easy |
| Attacker Awareness | High |

This attack does not exploit the bugs of any specific browser and affects the web servers where the web applications are hosted. According to [43], we could list the impact of cross-site scripts in the following way:

- The web application’s content could be modified by injecting scripts that display false advertisements, influence the reputation of commercial websites, or mislead the user.
- The theft of session cookies will be executed in open sessions to extract information while these sessions remain online.
- The potential for identity theft is a significant concern due to cross-site scripting attacks that can hijack the identity of legitimate users and lead to the theft of confidential personal information.
- The user’s HTTP sessions may be compromised if the attacker misuses this stolen information.

2.2.1. Types of XSS attacks and exploitation examples

According to the literature, there are three types of attacks [44]: Persistent XSS, Non-persistent XSS, and Document Object Model (DOM) XSS. The vulnerabilities are found in the software, the hardware, and the users (developers) that are part of any computing environment. The lack of mechanisms to filter and validate the input fields present in the web forms is exploited by XSS attacks, allowing the sending and execution of malicious scripts, which are stored in text files as instructions that are interpreted line by line in real-time for execution.

XSS non-persistent, reflected or indirect

The attacker can execute a malicious script to steal the victim's session cookies, allowing them to impersonate the victim and perform actions using the victim's permissions without needing a password.

In search engines, code can be injected through forms, URLs, cookies, flash programs, or videos. This attack exploits vulnerabilities in web applications that use user-provided information to generate an output response, hence the name "reflected."

To successfully execute an attack, a third mechanism is needed to transport the malicious code, such as through a method called spoofing. Using this technique, an attacker could trick a user into clicking on a website to run JavaScript code, redirecting all traffic from the victim to the attacker. If the victim's accessed application had an XSS vulnerability, the code would run within the trusted environment of the hosting website.

Persistent or direct XSS

In this type, malicious code is injected directly into the web page or vulnerable site. Scripts such as JavaScript or programming tags are required to execute this attack. The power of the attack allows these codes to be permanently on the web for all users after the first attack is executed.

When a user enters a website section with an injected XSS code, it can execute actions programmed in the web browser. This is more dangerous because the attacker's scripts are permanently stored on the server and displayed to website visitors.

The injection of malicious scripts into websites' content can elevate privileges, especially when users have their default Administrator account activated. To prevent this, consider disabling JavaScript execution in browsers, although user interaction may still be required in some cases.

DOM XSS

The type is considered more complicated but less known. It is also known as Type 0 or XSS based on DOM. The DOM (Document Object Model) is the structure of an HTML document, made up of HTML tags and their characteristics. The malicious code is injected through a URL but not loaded into the website's source code. Detection is more difficult because the malicious load does not reach the server; it is considered a local XSS, as the client-side scripts cause the damage.

The user opens an infected web page, and the malicious code exploits a vulnerability to install itself in a web browser file, running without any prior verification. Unlike the attacks that were previously analyzed, the server would not be involved in this attack. It resembles the attack of the reflected type since both require the user to click on some link. However, it is more effective to steal session cookies.

Difference between the 3 types of attacks

The difference between the three attack types is where they are executed. Direct and indirect XSS occur on the server side, while DOM XSS occurs on the client side (web browser) without server intervention. It's the application developer's responsibility to protect against these attacks, regardless of the type of XSS failure.

Another difference between attacks is in the time and place of execution. With the first two, the malicious codes are injected while processing the requests that originate through the entries programmed through HTML. With DOM, the malicious code is injected directly into the application during the execution time on the client.

By Script Origin:

- Not Persistent: Reflected in the server response.
- Persistent: Stored on the server.
- DOM: Manipulated and executed on the client side.

By Attack range:

- Non-Persistent: Affects individual users who interact with the malicious link or form.
- Persistent: Affects all users who access content stored on the server.
- DOM: Affects individual users who visit the page with the manipulated DOM.client.

By Attack Duration:

- Non-Persistent: Temporary, depends on user interaction.
- Persistent: Potentially long-term, until the script is removed from the server.
- DOM: Temporary, depends on the manipulation of the DOM in the user's browser.

2.2.2. Common scenarios for executing XSS attacks

Scenario for the reflected XSS attack:

The simplest scenario is when a web page is required to inject malicious code through its search engine. Figure 2.4 and Figure 2.5 show the result of an actual test performed within the pizza.com domain, where a malicious script was injected to show an alert in the web browser. This script that goes after the domain pizza.com is the easiest to execute (in this case, because sites vulnerable to XSS attacks were consulted).

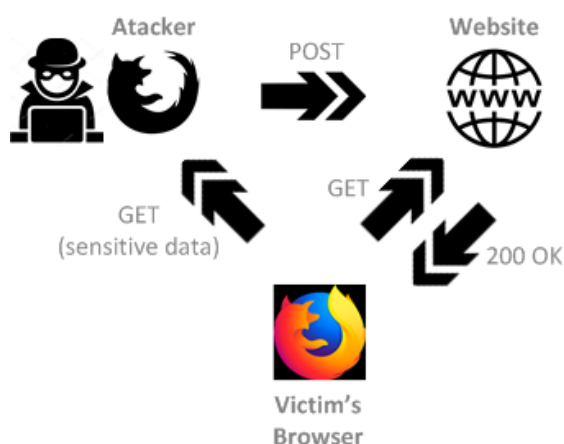


Figure 2.4: Scenario for a reflected or indirect XSS attack

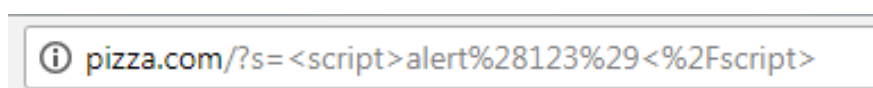


Figure 2.5: Sample XSS code reflected injected in the pizza.com domain

As a result of a first test, Figure 2.6 shows the view of the website pizza.com after the execution of the malicious code. As shown in Figure 2.5, the bottom of the website is altered after this exploitation; it changes to black, as shown in Figure 2.7. In a second test, the Google Chrome browser blocked the execution attempts of the malicious code, so no more results were obtained; this can be seen in Figure 2.8; the message indicates that an unusual code has been detected on the page, and that has been blocked.

Scenario for the stored XSS attack:

As shown in a previous study [45] to execute an XSS attack, it requires using any social engineering technique so that the user, through a convincing link, accesses a contaminated page with a JavaScript type file (*.js) Figure 2.9, which infects the victim and establishes a

Find the Pizza You're Looking for: Delivery, Local Pizzerias, and Recipes all in One Place

by Pizza.com

Recent Articles

The Pizza Vending Machine

Most college students would deem a pizza vending machine too good to be true; however, technology has graced the world with a vending machine that ...

Figure 2.6: Preview of Pizza.com domain before the XSS attack

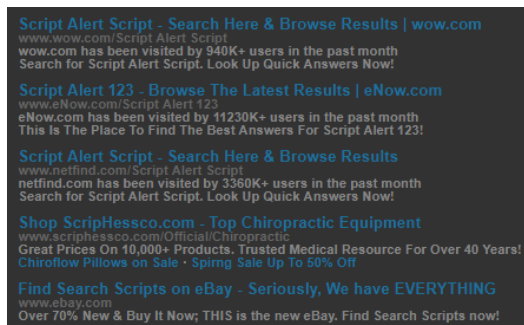


Figure 2.7: Preview of Pizza.com domain after the XSS attack

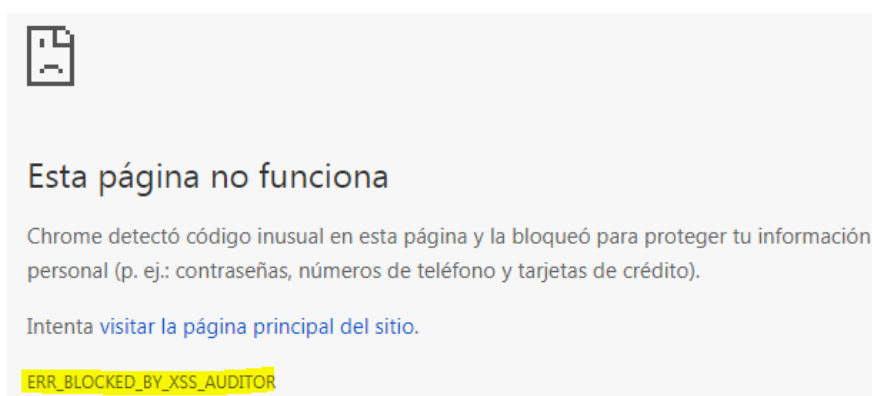


Figure 2.8: Chrome browser response to XSS attack

link with a remote controller. This attack uses the ignorance of the user when accessing a reliable page. In this test, Beef software was used, which stores a vulnerability in a web page that turns machines into zombie computers. This link is established even after closing the victim's browser.

As a framework to perform security tests for web applications (BeeF), a large number of attacks can be executed, such as exploiting extensions of programs such as Adobe, Flash, stealing cookies, displaying alert messages, making the user believe that they closed their Facebook' session and, therefore, steal their credentials.

Scenario for the DOM XSS attack:

DOM XSS uses scripts similar to vulnerable JavaScript code and runs directly in the browser, as shown in Figure 2.10. A vulnerable script could be used to send a URL link via email to be clicked or insert a URL into a website for when it is visited and clicked.

In both scenarios, the URL will be linked to the trusted site and will contain additional data that will be used to execute the XSS attack. Secure Socket Layer (SSL) type connectivity

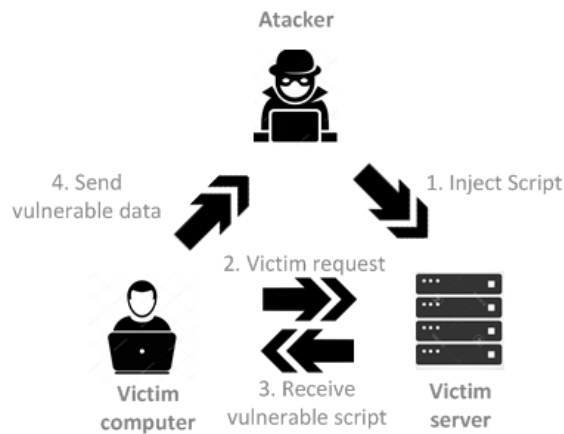


Figure 2.9: Scenario for executing stored XSS attack

offers no protection for this problem.

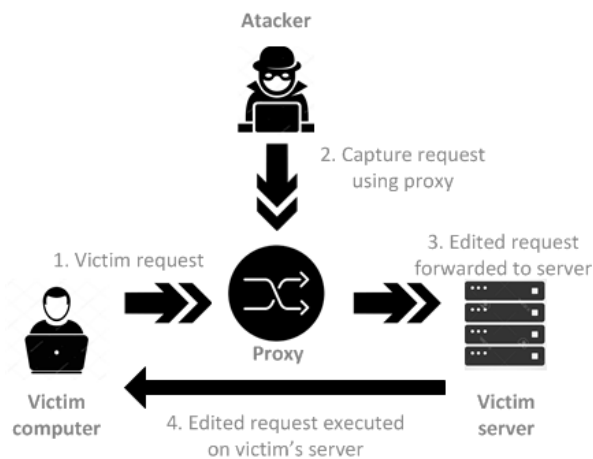


Figure 2.10: Scenario for DOM XSS attack

The different consequences associated with XSS attacks are detailed below.

- **Disclose information stored in users' cookies:** An advanced user could create a script on the client side, which, once executed, performs a malicious activity. The consequence of this script is that it will be uploaded and executed each time a user visits the website.
- **Handling:** Some vulnerabilities can manipulate or steal cookies, create requests to confuse or impersonate a valid system user, steal secret information from users' systems, or execute harmful code.
- **Other harmful attacks include:** Disclosure of secret files, installation of Trojan programs, redirecting fake pages to the user, execute "Active X" controls (Edge) from sites

that are perceived as reliable, and modification of the content of these sites.

2.3. COOKIES THEFT THROUGH XSS ATTACKS

2.3.1. Background of XSS attacks and theft of cookies

As cookies are transferred in plain text there are high possibilities of that cookies can be manipulated [23].

Cross-site scripting (XSS) is a type of attack that can result in an adversary executing an arbitrary script and accessing personal information on a victim's computer. Using an XSS attack, an adversary can easily get hold of the victim's cookies. However, if the adversary cannot use the stolen cookies to impersonate the victim, then stealing them is pointless [46].

Cross-site scripting (XSS) is a kind of online attack where cyber-criminals insert harmful code into web pages. This allows them to gain access to sensitive information stored in a user's browser, such as login details, personal data, and session cookies. XSS attacks can be dangerous and jeopardize website security, which is why it's necessary to take precautions to avoid them. To prevent XSS attacks, it's essential to keep web pages free of harmful code [47].

2.3.2. Statistics to remedy cookie theft:

Currently, most web applications use cookies to maintain the user's session status; the cookies are sent after the user has authenticated (session cookie). No additional authentication will be needed for a later connection because the validated cookies will only be verified to allow the new request. This authentication functionality makes cookies a potential target for attackers because websites create them and contain small amounts of data that can be sent between a sender and a receiver.

According to their browsing habits, cookies can identify users by storing their activity history on a website to offer more specific content according to their preferences. For example, when a user visits a web page for the first time, a cookie is saved on his computer. If the user visits the same page later, the website server requests the same cookie to update it with new configurations of the site; this is how the user's visit becomes so personalized.

As seen in Figure 2.11, according to the class of vulnerabilities for insecure session cookies and session cookies of non-secure type, the remediation rate is less than 20% [48]. In addition, in Figure 2.12, developers are always oriented to fix or remedy the most

accessible or effortless problems. For example, incorrect settings of applications have a 74% remediation and unpatched libraries (62%). However, the most complex and challenging solutions are still those of type XSS (38%) and SQL injection (32%) [48].

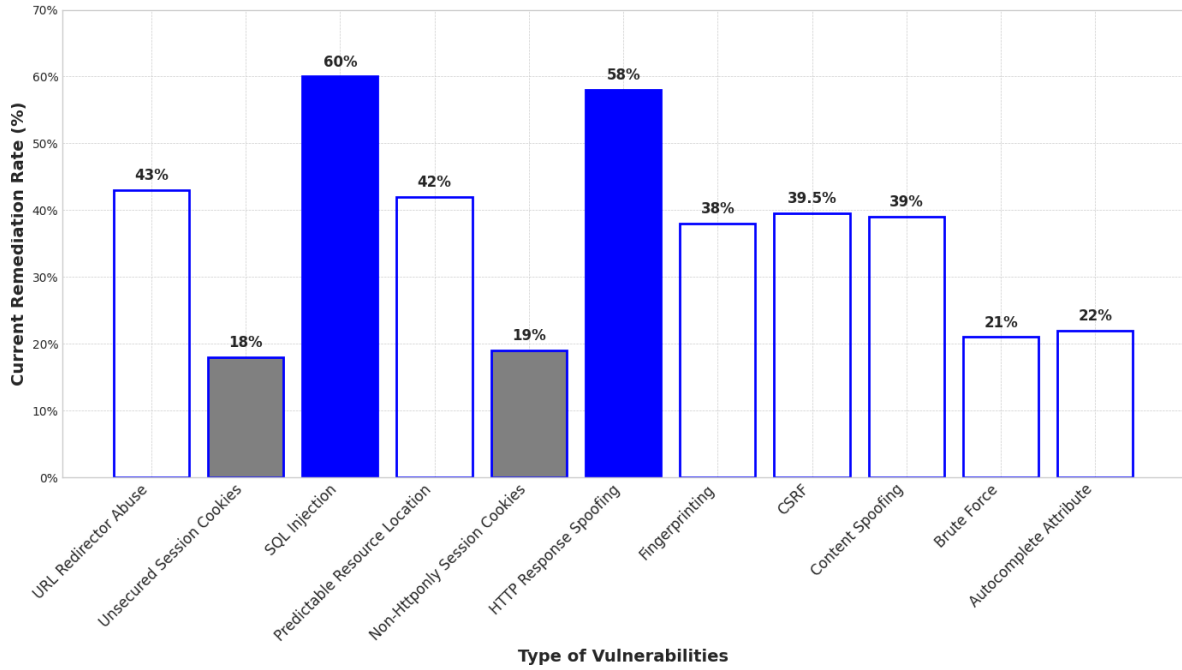


Figure 2.11: Current remediation rate according to the type of vulnerabilities

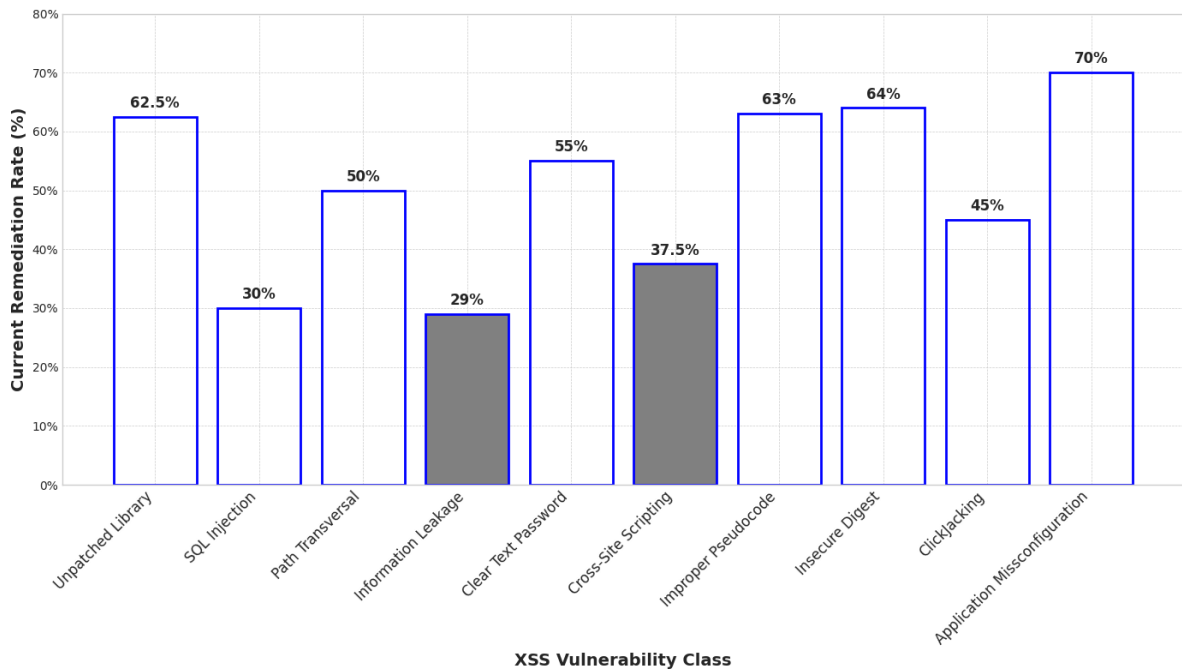


Figure 2.12: Remediation rate according to the XSS vulnerabilities type

Chapter 3

Literature Review

Índice

| | | |
|-------|---|----|
| 3.1 | Cross-site scripting (XSS) attacks and mitigation: A survey | 32 |
| 3.1.1 | Analysis of Methods and Tools | 32 |
| 3.1.2 | Analysis of results | 45 |
| 3.2 | A Systematic Literature Review Comparing the Effectiveness of XSS at- tacks in Web Applications vs Cookie Stealing | 47 |
| 3.2.1 | Protocol for systematic document review | 47 |
| 3.2.2 | Data collection process | 53 |
| 3.2.3 | Methodology used to analyze the content of documents | 54 |
| 3.2.4 | Analysis of Results | 56 |

3.1. Cross-site scripting (XSS) attacks and mitigation: A survey

3.1.1. Analysis of Methods and Tools

This section has been structured in three subsections, the first details the methodology used to select the documents, the second explains the methodology used to analyze the documents, and the third explains the technical characteristics of each selected document.

Methodology used for document selection

The search for information has been structured as follows:

1. We have analyzed some surveys [49]–[53] related to the topic of XSS attacks, and all the contents that each one details has been mapped in order to propose our contents.

Through a deeper analysis, specific parameters were found that served to structure the content of our survey. Also, its content was a limitation, aimed at describing the operation and type of XSS attacks. Our interest was to analyze methods and tools proposed to mitigate these types of attacks.

2. The search was delimited between the years 2013 and 2019.
3. It was included all those proposals that include in the title (XSS) or (Cross-Site Scripting) or (XSS and SQL Injection) or (Cross Site Scripting).
4. All the works that did not present some type of methodology, proposal or tool to mitigate this type of attacks were excluded. In addition, works that only present a short literary review or concepts on the subject analyzed, were also excluded.
5. The proposed hypothesis will allow us to find the tendency of proposals of the classic technology versus the use of artificial intelligence to mitigate XSS attacks.
6. Within the trends, we also want to find those that use cookie analysis to mitigate XSS attacks; this is a complement to guide our current research.

Methodology used to analyze the content of documents

Our goal was to find all the methods, proposals, classic tools, and those that use some artificial intelligence technique to compare the proposed period (2013 to 2019) and find the current trend to mitigate the XSS type of attacks.

To select the parameters that were analyzed in each document, we rely on specific common characteristics found in all documents; for example, according to the classification proposed in [54], there are two analysis approaches:

- Static Analysis (In Tables 3.1 to 3.10 abbreviated as **S**)
- Dynamic Analysis (In Tables 3.1 to 3.10 abbreviated as **D**)

In the same way in [55] a combination between the static and dynamic approach called *hybrid* (In Tables 3.1 to 3.10 abbreviated as **H**) has been proposed.

On the other hand, we have found 3 ways of prevention analysis of these types of attacks:

- From the client's side (In Tables 3.1 to 3.10 abbreviated as **C**)
- From the server side (In Tables 3.1 to 3.10 abbreviated as **S**)

- Hybrid client/server combination (In Tables 3.1 to 3.10 abbreviated as **H**)

Most proposals aimed at the DOM Cross Site Scripting detection method (DOM-XSS) have been divided into three types: black box fuzzing, static analysis, and dynamic analysis [56].

Other defense methods found in [57] are based on the analysis of code extraction:

- Through input validation
- Through escape characters
- Through filters
- Through the set of characters that is specified

Our objective was to analyze and extract the following variables from all the documents analyzed:

- Type of tool of method proposed
- Action performed by this method / tool
- Objective of the proposal
- Where the proposal is oriented or applied
- To whom it is developed: client (C), server (S), hybrid (H)(client and server)
- What are its advantages?
- What are the limitations?
- What type of analysis is: static (S), dynamic (D), hybrid (H)(static and dynamic)
- Was any kind of artificial intelligence technique used (AI)?

Analysis of scientific documents

In this section, we describe all the information analyzed in the documents selected for our study, considering the above mentioned variables.

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | S/D/H | AI |
|------|--|---|--|-------------------------|-------|---|--|---------|--------------------|
| [58] | Qualys Guard | Escaner Web | Detect intrusions, anomalies based on learning and attacks against applications and the web server | Web Application Servers | H | It is based on a system that qualifies URLs as safe or not, allowing their execution or not in the system | The functionality is incomplete as the client has to manually scan the URLs they want to access. | S/D | No |
| [59] | - | Entry validation | Automatically extract the encoding functions used in a web application to disinfect untrusted entries | Web Application | C | Slit guides developers to detect and correct problems without relying on security experts | It only targets XSS attacks | 0 Day S | No |
| [60] | Acunetix WVS, Rational AppScan Enterprise, ZAP | Black box web scanner | Develop a custom test bench to compare the capabilities of 3 black box scanners | XSS Stored | C | Allows the use of scan profiles to specify to which category of vulnerabilities the attack should be directed | Relatively poor performance in detecting only stored XSS | - | No |
| [61] | Ghost.py | Browser without header | Interpret JavaScript and load Ajax content simulating browser behavior to obtain hidden injection points | Web Application | C | More accessible and portable system for secondary development | Use a framework to attack pages with pre-established vulnerabilities | D | No |
| [62] | - | Optimized repertoire of attack vectors | Automate attacks and dynamically detect XSS vulnerabilities in applications | Web Application | C | Generate XSS attack vectors automatically | It has only been tested on 24 websites | D | Machine Learning |
| [63] | Kameleon Fuzz | Combination of evolutionary fuzzing with inference models | Generate entries through genetic algorithms through a formal learned model | XSS type 1 | C | Generates test cases through the automated search of XSS type 1 | It has not been experienced in real-world applications | S | Genetic algorithms |
| [64] | - | Technique based on concolic tests | Use a contextual tool to translate web applications by identifying input and output variables. | JSP Application | C | It is able to identify the entries that occur in the form of malicious strings | It is only oriented to pages of type JSP | - | No |
| [65] | - | JavaMail Development Kit | Build a repository of XSS test vectors to implement a dynamic tool | Webmail systems | C | 14 XSS attack vectors related to HTML5 have been identified | The HTML5 tag (embedded) is not given sufficient consideration by the filtering mechanism. | D | No |

Table 3.1: Analysis and discussion of the tools and methods found. (1/10)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | S/D/H | AI |
|------|---|---|--|--|-------|--|---|-------|--------------------------------------|
| [66] | Web Input Vector Extractor Teaser (WIVET) | Distributed scanning tool | Provide a database of attack vectors to search and correct vulnerabilities within websites | Modern Web Applications, DOM (XSS, Open Redirection, Clobbering) | S | It is functional to analyze data on a large scale | Only 1000 of the best websites in the Alexa ranking were analyzed | - | No |
| [67] | Dom XSS Micro XSS | Benchmark | Extract representative vulnerabilities to create templates | XSS DOM | S | Acts as a basis for further development and discussion in the community | Currently the tool is not complete | D | No |
| [68] | Repositorio GIT | Text mining | Detect code files vulnerable to XSS | Source code of PHP web applications | S | It does not depend on the programming language version and works with object-oriented code | Source code tokens are not enough as a feature to develop machine learning models | D | NB, Bagging, Random Tree, J48, JRip. |
| [69] | XSSDM | Analysis of patterns by C | Act as a prototype to evaluate a set of public data | PHP | C | Minimizes false positive and false negative type results | It has only focused on web applications developed in PHP | S | No |
| [70] | RIPS, PIXY | Detailed analysis through defensive programming | Precisely detect XSS vulnerabilities based on sensitive context vs. HTML context | PHP | - | Minimize false positive and false negative type results | Only one empirical evaluation has been presented | S | No |
| [71] | Fiddler | Development at the proxy level based on Kullback-Leibler Divergence (KLD) | Analyze the distance between the probability distribution of the legitimate JavaScript code and the JavaScript code present in a response page | PHP | C | Can detect most known XSS attack signatures and display a very low false positive warning | They only apply the approach in three vulnerable PHP applications | - | No |

Table 3.2: Analysis and discussion of the tools and methods found (2/10)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | S/D/H | AI |
|------|---|---|--|--|-------|--|---|-------|--------|
| [72] | JAVA, SOA, DAO, MVC, Facade, and Security | Validate the prototype at the design, coding and security level | Utilize architectural patterns like DAO, Facade, and WS-Security framework (MVC) based on design specifications. | J2EE | - | Demonstrate that the use of standards, techniques and recommendations are necessary to avoid XSS vulnerabilities | The SOAP web service description document (WSDL) is extensive | S | No |
| [73] | - | Analysis of cookies | One-time password and authentication to identify if a person is a valid owner of the cookie | Avoid theft of cookies | C | It is a secure protocol that avoids the abuse of cookies stolen by XSS | It does not work if the attack is made before the cookies expire, it also has latency | - | No |
| [74] | - | Validate the input strings | Check that the websites are vulnerable, giving the guidelines to the developers | Software and regular expressions | C | Regular expressions protect multiple domains and prevent XSS attacks | Only 50 popular domains have been analyzed | - | No |
| [75] | Pearl based IDS | Capture the executable content | Mix the executable code with the content and obtain a Hash for a later visit | Web Application | C | Useful for web forums and other sites where the user controls the content | The tests are carried out with web pages created by the authors | S | No |
| [76] | - | Filtering terms | Filter user entries to protect any web application | Web Application | C | An adequate level of protection has been achieved | Better scalability and improved security measures are needed to guard against future sophisticated attacks. | S | No |
| [77] | - | Attack model | Implement attack tests using XSS patterns to generate case studies | UML | C | Test methodologies are integrated into the software development cycle | Only use 2 web applications for type 1 and type 2 attacks | S | No |
| [78] | XSSERC | Filter based on rules | Estimate patterns to evaluate whether intercepted requests have malicious attempts or not | Browser extensions, COR-JACKING, HTML5 | C | Intercept web requests and evaluate them to estimate patterns | They only test the performance of the system for the 50 most popular popular websites in the world | S | No |
| [79] | DMOZ, XSSed DB | Classifiers improved | Sort web pages by identifying their characteristics with an improved program model | Online Social Networks (OSN) | C | They can simulate and obtain more precise experimental data | The combination of program model and classifiers affected the detection result. | S | AdTree |

Table 3.3: Analysis and discussion of the tools and methods found (3/10)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | S/D/H | AI |
|------|------------------------------------|---|---|-----------------------------------|-------|---|---|-----------|---------------------------------|
| [80] | DMOZ, XSSed database, n-grams WEKA | Classifiers and improved database, n-grams | Improve what is proposed in Paper28 | Online social Networks (OSN). | C | An initial test database has been used | - | S | AdTree, AdaBoost |
| [81] | Short | Coincidence of regular expression patterns | Show how to investigate the pattern or behavior of XSS attacks | Persistent and non-persistent XSS | H | It is enough as a first defense barrier | Many false positives, use of resources, does not perform prevention or deterrence | S | No |
| [82] | - | Validation of the absence and weakness of entries | Statically check the vulnerabilities and face the formalization of policies based on the World Wide Web Consortium (W3C) and the source code of the Firefox browser | Javascript interpreter | C | Its source is the flow of contaminated information combined with string analysis (semantics of input validation routines) | Only oriented to Firefox, large number of lines programmed in O'neil | Practical | No |
| [83] | WebKit XSS Auditor | PHP Array Injection, PHP Array-like Injection | Take advantage of the incorrect administration of variables and matrices in the PHP code to bypass the XSS Auditor | Web applications based on PHP | C | Focus on the mistakes that PHP-based web application developers make | They only present concrete examples of poorly written PHP code | S | No |
| [84] | ETSS Detector | Web Scanner | Automatically analyze web applications to find XSS vulnerabilities | Web application | C/S | Ensures the correct entry of the entry fields | Can only detect persistent and non-persistent XSS | - | No |
| [85] | Git Repository | Feature extraction algorithms | Build several machine learning models to predict context sensitive XSS security vulnerabilities | PHP | C | Sort and separate a vulnerable source file from a benign source code | Unable to extract vulnerability prediction characteristics | S | SVM, NB, Bagging, J48, and JRip |

Table 3.4: Analysis and discussion of the tools and methods found (4/10)

| Ref. Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | E/D/H | IA |
|----------------------------------|---|--|-----------------|-------|---|---|-------|------------------|
| [86] – | Buffer-based cache test | Store a cache that contains a validated instance of trust from the last page that was submitted | Mobile Browsers | C/S | It has reduced the rendering time of browsers, as well as the number of queries within browsers | Oriented only for mobile browsers | S | No |
| [87] XSSDE, SOOT | Program analyzer, defence feature miner (DF miner) and TIFG | Propose a code audit approach to recover the defense model implemented in source code | HTML | S | A variant of flowcharts has been proposed | It is not oriented to DOM XSS | – | Pattern Matching |
| [88] Xbuster | Firefox extension | Split and store each HTTP request parameter separately (convert them into HTML and JavaScript contexts) | Firefox browser | C | Protects against all attack vectors XSS, protects against clickjacking | There is a delicate balance between the false positive rate and the false negative rate | – | No |
| [89] XSS-ME | Firefox extension | Block the execution of malicious scripts specifically XSS vulnerabilities reflected | XSS reflected | C | Condensing the false negative rate | Only oriented to mozilla firefox | S | No |
| [90] JSON dictionary, JavaScript | HoneyPot con LikeJacking | Emulate vulnerabilities through a low interaction honeypot to be exploited using XSS | HTTP requests | – | It not only records the attacker's activity but also tries to expose his identity | Some attackers use techniques to hide their identity, so they can not be tracked | – | No |
| [91] SpiderNet | Machine learning techniques, MATLAB .NET, | Predicting the type of malicious attack used in machine learning approaches, the prediction time has been considered | DOM XSS | C | Structured dataset of a set of pages in real time | – | – | SVM, ELM |

Table 3.5: Analysis and discussion of the tools and methods found (5/10)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | E/D/H | IA |
|------|--|--|--|---|-------|--|---|-------|-------------|
| [92] | XSS-me (Firefox), XSS Auditor (Chrome) | Browsers add-ons to simulate attacks | Evaluate the most commonly used web browsers to demonstrate that there is no adequate defense against XSS attacks | Reflected XSS, Explorer11, Chrome32 and Firefox | C | It is more appropriate if the plug-in is integrated within the browser, that is, it is applied as an extension | Compared to three browsers: Chrome, Firefox, and Explorer. | - | No |
| [93] | - | Laboratory to simulate attacks | Educate and motivate the understanding of the meaning of XSS vulnerabilities | End users | - | Students can better understand XSS vulnerabilities, how they occur, how to exploit them and how to solve them | The proposal skips technical details and only covers source codes for XSS attacks. | - | No |
| [94] | PURITY | Malicious activity emulation | Detect if sites are vulnerable to SQL and XSS injection threats | Web Application | C | It is executed automatically and also manually. It differs from other tools because it is based on planning | The program is mostly automated with minimal manual intervention. Only a single case study has been assessed. | - | No |
| [95] | CSRFQue | Session kens | Filter and intercept requests, verify if a token exists or not and match the reserved token of the current session | Java EE | E | Dynamic addition of Token through event triggering can prevent the use of XSS to bypass CSRFGuard. | Scripts need manual insertion. CSRF can be skipped if attackers exploit XSS. | D | No |
| [96] | DIMVC | Mining of code attributes to predict vulnerabilities | Classify SW as "safe" or "unsafe" | Software | H | The cases that can not be proven are predicted through extracted signatures | Mining is applied only to the relevant code that is affected by the data entry (input validation) | H | Data mining |

Table 3.6: Analysis and discussion of the tools and methods found (6/10)

| Ref. Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | E/D/H | IA |
|--------------------|--|---|--------------------------------------|-------|---|--|-------|----|
| [97] Grease Monkey | Create proof of concept docs with text filtering, syntax analysis, and scripts. | Python filter, JS analyzer, DOM symbolic execution, regex automaton in Java, and a navigation simulation library. | XSS, DOM, extensions of web browsers | H | The waiting time is relatively low | Scripts may be omitted due to compatibility issues with DOM manipulation. | H | No |
| [98] – | Content security policies or CSP | Enhance online communication security. | Secure web services, IoT | H | Recommended for communication in interactive environments (people and devices). | No XSS attacks have been spread on IoT devices | – | No |
| [99] XSS Chaser | Analyzing chains for vulnerable patterns using a Non-Deterministic Turing Machine. | Create vulnerable patterns to avoid XSS. These patterns are generated using text string analysis | Web Application | C | XSS Chaser finds attack patterns in less time than existing algorithms | Analyze only with samples of attack patterns | S | No |
| [100] IDS | Containers through a mapping model of query requests | Client containers limit damage to themselves. | Web Application | S | Use containers to limit XSS attacks | For accurate results, save URLs in a text file and consider network speed. | S/D | No |
| [101] .Net | Attacks on systems in the form of web requests | Provide individualized web services by querying the client's specific data from the database. | Web Application | S | Reduce false positives and enhance input disinfection. | It's just a proposed study | S/D | No |
| [102] – | Coding of N and binary alphabets | Dynamic access control method to detect and prevent XSS attacks. | Web Application | C | An improved web proxy has been proposed to avoid XSS attacks based on advanced coding | Not optimal for multidomain XSS attacks | D | No |

Table 3.7: Analysis and discussion of the tools and methods found (7/10)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | S/D/H | AI |
|-------|--|--|--|------------------------------------|-------|---|--|-------|--|
| [103] | Zend Framework, HTML Purifier | Combination of two models | This model combines multiple tools into a sequence of levels. | Web application | C | Hide web page names and design language to protect the URL. | - | S/D | No |
| [104] | VB Script | Malicious execution sequences identification | Identify legitimate and malicious sequences based on official lists and training data. | Input validation | C/S | The proposal detects all types of XSS attacks through a content-based approach. | The WAEP automatic update using the log report analysis is a challenging task | S | No |
| [105] | - | Automatic modeling algorithm | Store the behavior model of the website in the form of an XML file | Web application of e-commerce type | C | The behavior of an operation is judged if it complies with the requirements of the legal behavior of the website, in order to avoid XSS attacks from the point of operation | It is only oriented to e-commerce pages | S/D | No |
| [106] | Snort | Signatures to detect XSS attacks | Use rules to identify XSS attacks by monitoring incoming and outgoing packets that match or not match predefined rules | Analysis of packages | C/S | The experiments have been carried out in a real network environment | Many resources are required for large-scale package analysis | S | No |
| [107] | Wordpress, BuddyPress, OTC browser, Fennec | Session Authentication | Use unique cookies as a more robust alternative to prevent attacks such as session hijacking | Web application (PC and mobile) | C | The proposal is an experimental evaluation to characterize and compare the performance of authentication cookies | Some cookies use symmetric encryption to protect session information of sensitive users. | S | No |
| [108] | Weka | Authentication Tokens | Evaluate a set of cookies collected from 70 popular websites obtained from the Alexa ranking | Web application | C | The proposal allows storing the user name and password information using tokens | Only protect authentication tokens | 2 - | Supervised learning, binary classifier |

Table 3.8: Analysis and discussion of the tools and methods found (8/10)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | S/D/H | AI |
|-------|---|--|--|----------------------------|-------|---|---|-------|--------------------|
| [109] | Newton | Checklists | Help programmers verify safe implementation of authentication cookies. | Developers | S | It is oriented to the developer user | Some sites taking deployment were not tested in the lab, causing high server load. | - | No |
| [110] | Ground truth of cookies | Collect cookies from a group of websites and mark each cookie with a binary identifier | Assess cookie identification and detection techniques for their effectiveness and level of protection. | Web authentication schemes | C | The proposal of supervised learning is very precise, achieving a good balance between security and usability | The proposal is limited since the client's authentication is based on complex uses, often difficult to predict. | - | Binary Classifiers |
| [111] | Using windows tools (Anti-XSS) | Recommendation for use | Secure development life cycle practices and techniques | Software development | C | Makes recommendations for using windows own tools | Only focus on type 1 or non-persistent XSS | S | No |
| [112] | Web Proxy | Sample website with some basic user interactions | Relieve XSS attacks | Web applications | C | Analyze the 3 types of vulnerabilities (persistent, not persistent, reflected) | Customer system performance decreases. Poor web browsing | S | No |
| [113] | PHP functions | Evaluate the efficiency of web pages | Provide sufficient knowledge while developing and using websites | Developers, users | C | Regular expressions that can be used to find threats. | It only targets PHP expressions | S | No |
| [114] | Browser extension | Google Chromium browser extension | Alert users whenever a possibility of XSS attack is discovered | Web applications | C | The extension starts to work automatically when the browser is loaded and also works when a new tab is opened | It is only chrome oriented and does not analyze other browsers | H | No |
| [115] | PHP framework Laravel version 4.2, PHP 7, MySQL | XML processing and its practical tests | Propose the set of rules to protect the graphic content of websites | Web browser graphics files | C | Overview of the problems associated with (XSS) in the graphic content of web applications | Must include combination of black and white lists | H | No |

Table 3.9: Analysis and discussion of the tools and methods found (9/10)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | S/D/H | AI |
|-------|--------------------------------------|--|--|---------------------------------|-------|---|---|-------|-----------------------|
| [116] | Moving Target Defense (MTD) - XSSMTD | Frequently changing system settings | Defeat XSS attacks | Web applications | C | Add a random attribute to each insecure element in the web application, little impact on system performance | The tests performed only use Windows 7 and virtual machines | H | No |
| [117] | Python | Tracker, builder, simulator, detector and generator. | Study the principle of vulnerability generation and the mechanism of XSS attacks | Web applications | C | Refine dynamic black box model for XSS detection. | Only one forum site was used as a test objective for comparison | D | No |
| [118] | JavaScript | This framework is a Google Chrome extension. | Suggest an offline and online model based on a disinfection method | OSN | C | Disinfection with nested context recognition | Oriented only to one type of browser | S | No |
| [119] | Statics Analysis Tools | Choose vulnerable apps from a repository. | Measure for all possible combinations of (Static analysis Tools) SAT | Wordpress plugin | C | Analyze the performance of diverse Static Analysis Tool configurations. | Potential increase of False Positive. | S | Binary classification |
| [120] | Dynamic feature extraction technique | Robust multilayer perceptron scheme | Combination of the mechanism of extraction of dynamic characteristics and the model of neural networks | Web applications | H | It acts as a layer to extract and provide training and test data sets | Currently, there is no open dataset available for XSS based attacks | D | ANN |
| [121] | word2vec, python3.6.5, klearn0.19.1 | Vectorization method improved. | Find the frequency of occurrence and coincidence of words in XSS scripts | Web applications | C | A large data set has been used to decrease data deviation | - | S | Machine learning |
| [122] | JS-SAN (JavaScript SANitizer) | Deepest possible web page tracking to find possible injection points | Generate an attack vector template | Web applications | C | Grouping in the payloads of the JS attack vector extracted | Oriented only to one type of browser | H | No |
| [123] | PHP-Sensor | Defensive model | Discover vulnerabilities of workflow violation attacks and XSS attacks simultaneously | Real-world PHP web applications | H | Discover the automatic propagation characteristics of XSS worms | Only oriented to PHP type pages | S | No |
| [124] | Cloud-based framework | Detects malicious HTML5 code injected into XSS DOM nodes. | Collect key web application modules, extract suspicious HTML5 strings | HTML5 web applications | C | Frustrates XSS vulnerabilities based on DOM | Oriented only to one type of browser | H | No |

Table 3.10: Analysis and discussion of the tools and methods found (10/10)

3.1.2. Analysis of results

A mental map has been proposed in Figure 3.1. Here, the classification of methods or tools that use traditional technology to mitigate XSS attacks has been structured.

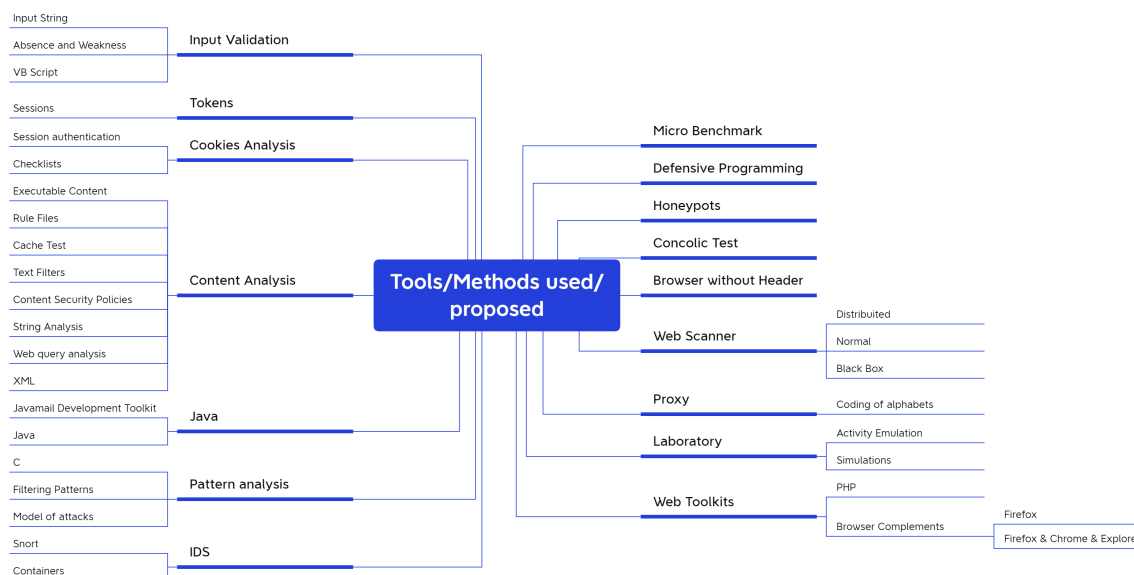


Figure 3.1: Classification of tools/methods proposed and used to mitigate XSS attacks

In Figure 3.2, a diagram illustrates the use of artificial intelligence methods and tools to detect or mitigate XSS attacks.

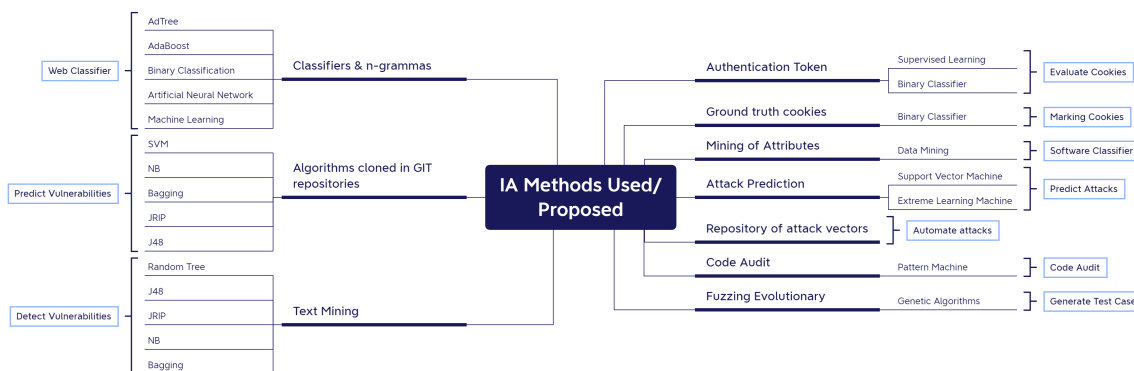


Figure 3.2: Classification of proposals that use some method of Artificial Intelligence to mitigate XSS attacks

The following methods have been proposed, using artificial intelligence techniques to detect XSS attacks. They have been structured as follows:

- Documents that evaluate and mark cookies
- Documents that predict and automate attacks
- Documents that execute code audit
- Documents that predict and detect vulnerabilities
- Documents that make Web and Software classification
- Documents that use the generation of test cases

After conducting our analysis, we found the following information: there is a high trend in the use of traditional tools to detect and mitigate XSS attacks. As shown in Figure 3.3, the most significant tendency is to "Analyze the Content of the Web Pages" (13.20%) and use "Web Toolkits" (16.98%) to detect this type of attacks. Additionally, web scanners, pattern analysis, the use of Java, and input validation are also proposed in high numbers.

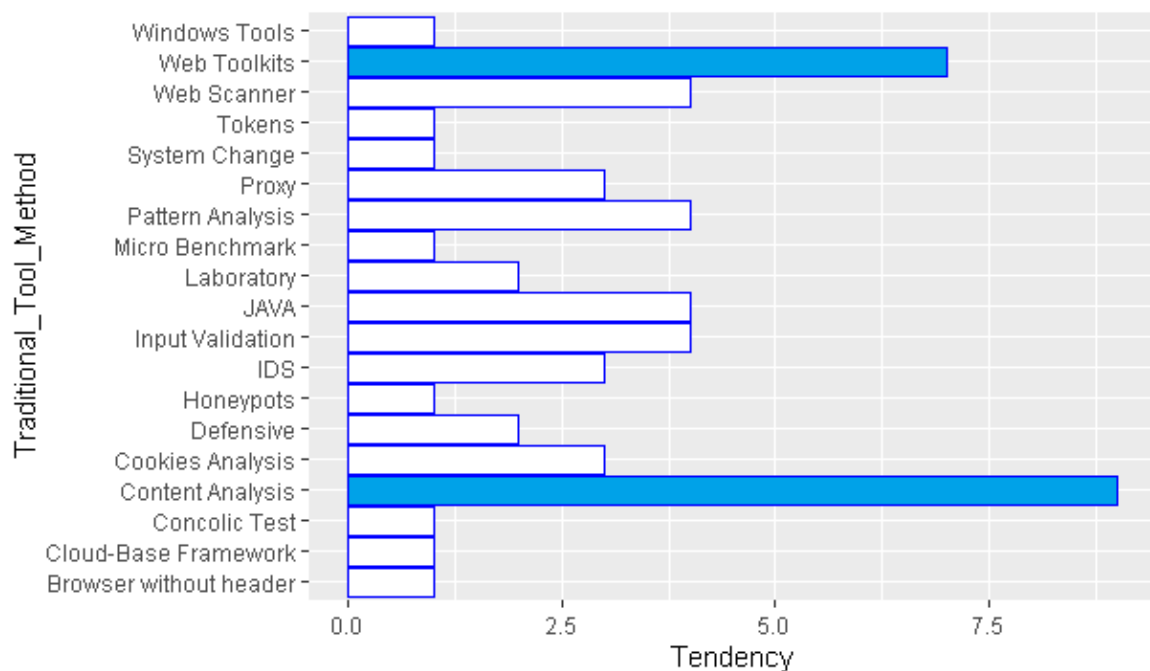


Figure 3.3: Trends in the proposal of the traditional methods/tools to mitigate XSS attacks

Regarding the use of artificial intelligence techniques to detect or mitigate this type of attack, it has been found that the application of **Web Classifiers (9.43%)** is one of the most used proposals in all scientific documents, as can be seen in the Figure 3.4. The blue arrow in the figure means that there are very few AI tools that analyze the properties of cookies by applying a technique called "cookie marking."

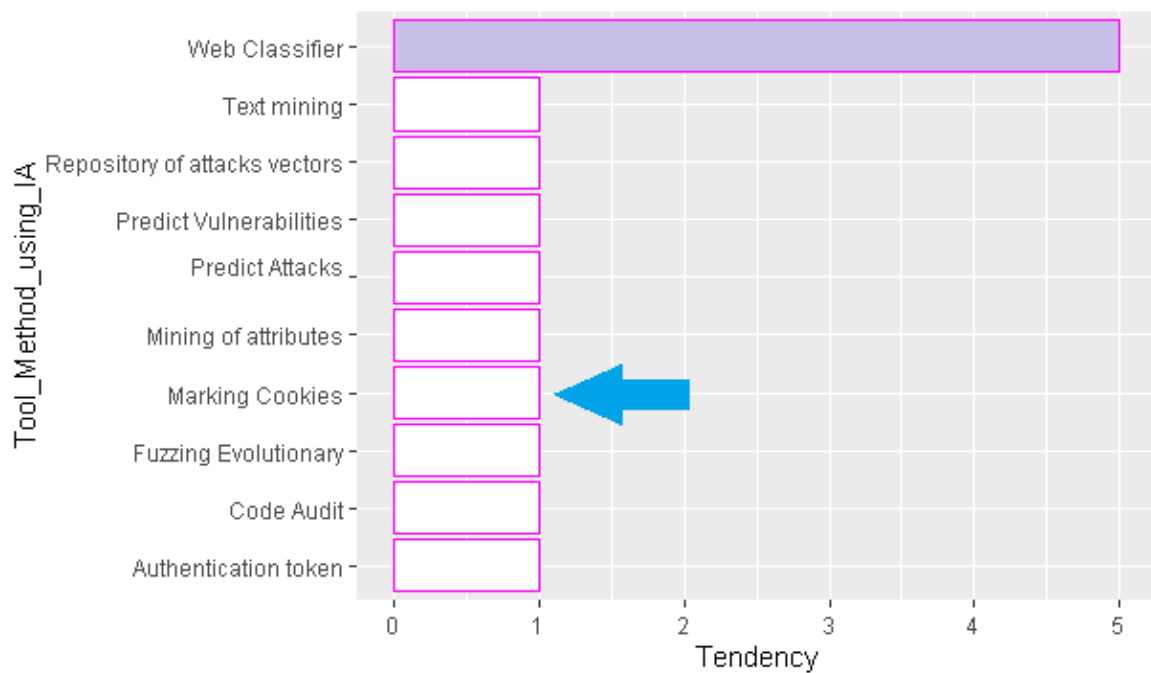


Figure 3.4: Trends in the proposal of methods/tools using Artificial Intelligence to mitigate XSS attacks

3.2. A Systematic Literature Review Comparing the Effectiveness of XSS attacks in Web Applications vs Cookie Stealing

3.2.1. Protocol for systematic document review

The research findings were systematically documented and presented using the checklist of requirements from the PRISMA methodology. This checklist provided a structured framework for conducting the study, ensuring transparency and adherence to established guidelines. By employing the PRISMA methodology, the research process followed a standardized approach, enhancing the reliability and reproducibility of the results.

Eligibility criteria

The inclusion and exclusion criteria for the review and the methodology applied to select the articles are detailed below.

Inclusion Criteria

- The research was focused on the period from 2018 to 2023, building upon our previous study [125]. By delimiting the timeframe, we aimed to capture the most recent and relevant information within a specific time range
- Articles in English were analyzed
- Only research article-type documents were selected as primary studies
- Research like tutorials, SLR, Systematic Mapping Study (SMS), and surveys were selected as secondary studies (for related work)
- Articles within the field of computer science were selected for inclusion in this study. The inclusion of computer science articles ensures that the study is grounded in the most up-to-date and specialized knowledge within the field

Exclusion criteria

- Duplicate proposals
- Proposals without results
- Work without a clear or concrete methodological proposal
- Theoretical proposals
- All scientific proposals that did not present any methodology, proposal or tool to exclusively mitigate Cross-Site Scripting attacks
- Papers that only present a brief bibliographic review or general concepts on the analyzed topic
- All articles with download difficulties were also discarded, however, the content of their abstract was analyzed to complement our study
- Papers in whose title or abstract only part of the research component appears as 'Cross-Site or 'Script' or 'Site' or 'Cross'
- Proposals that do not belong to the scientific libraries are mentioned in the Information Sources section

Elimination criteria

- Articles with less than 4 pages

Information sources

The study utilized multiple scientific libraries as sources, including the ACM Digital Library, IEEE Xplore, Springer Link, Web of Science, and Mendeley. These reputable platforms were chosen to access a wide range of scholarly articles and research papers in order to gather comprehensive and diverse information on the topic.

Search Strategy

We have applied the snowball search method to our systematic bibliographic study. As mentioned in [126], the snowball technique uses an article's reference list and citations to identify additional articles. Analysis of references is called backward snowballing, and analysis of citations is called forward snowballing.

Figure 3.5 shows the general structure of our search strategy. We have used the Research Rabbit tool [127], an innovative citation-based literature mapping tool, to automate this process.

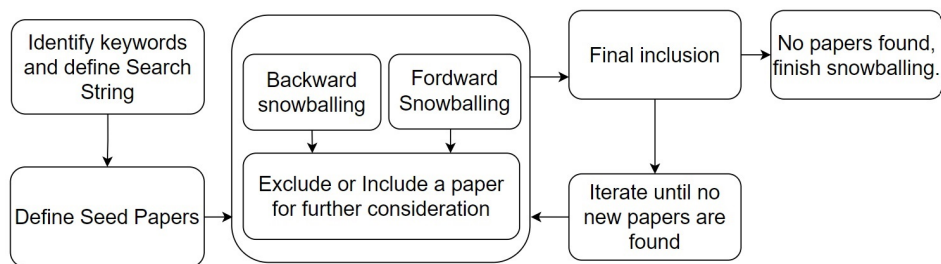


Figure 3.5: Search strategy using the snowballing technique

Identify keywords and define search string

Figure 3.6 shows the general structure of the three research domains (keywords) used to assemble the search strings. Our systematic review aims to investigate how personal information is leaked through XSS attacks. We have related the first domain (cookies) and the second domain (web pages) to the third domain corresponding to cross-site scripting (XSS)

attacks. We have included the main types of cookies related to exchanging information with third parties (third parties and trackers) to improve the search string.

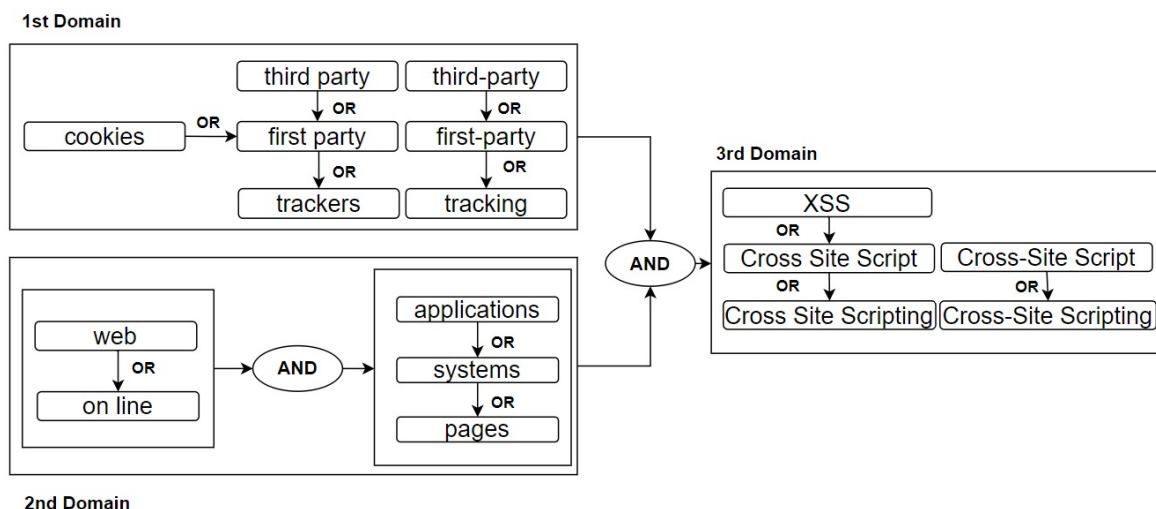


Figure 3.6: Overview of the research domains included to build the search strings

Table 3.11 provides an overview of the main keywords identified during the research process; the combination of research domains has facilitated the analysis and selection of the primary scientific papers (seed papers).

Table 3.11: Search Strings proposed to find and filter the articles

| No. | Search String |
|-----------------------|---|
| Search String 1 (SS1) | (cookies OR first-party OR third-party OR trackers OR tracking) AND (XSS OR Cross-Site Script OR Cross-Site Scripting) |
| Search String 2 (SS2) | ((web OR online) AND (systems OR applications OR pages OR site)) AND (XSS OR Cross- Site Script OR Cross-Site Scripting) |

Define Seed Papers

The process of selecting seed papers went through five phases of comprehensive analysis. A breakdown of each step, including the criteria used and resulting analysis, can be found in Figure 3.7 for promoting transparency in the selection process.

In addition, the following review filters were used to select the primary studies:

First filter

- **Title:** The titles of the publications found in the databases were reviewed.

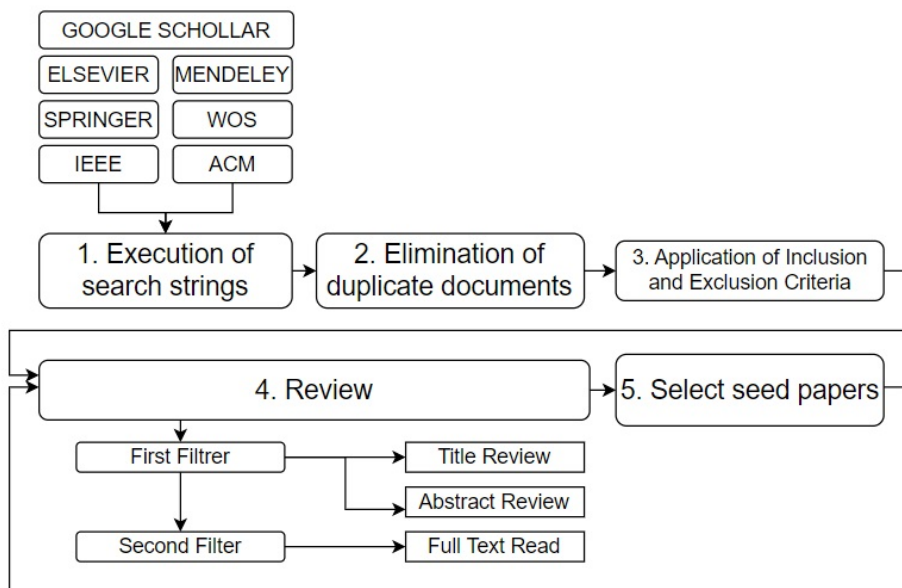


Figure 3.7: Overview of the process of analysis and selection of seed papers

- **Summary or Abstract:** Following the selected titles, the abstract was reviewed and read.

Second Filter

- **Full Text:** Finally, the publications that passed the first filter were submitted for reading and complete analysis.

Table 3.12 summarizes the results for each of the executed phases of Figure 3.7. As a final result, we have selected 73 articles as initial seed papers.

Table 3.12: Results after executing the search strings

| No. | Description | Recovered Documents |
|---------|--|---------------------|
| Phase 1 | Execution of search strings | 316 |
| Phase 2 | Elimination of duplicate documents | 220 |
| Phase 3 | Application of inclusion and exclusion criteria | 124 |
| Phase 4 | Full text review include first and second filter | 73 |
| Phase 5 | Select seed papers (Search String 1) | 25 |
| Phase 5 | Select seed papers (Search String 2) | 48 |

We have used 73 seed articles that were the basis for finding more related articles using

the Research Rabbit tool.

Backward, Forward Snowballing and Iterations

With the Research Rabbit tool, the backward snowballing (references) and forward snowballing (citations) techniques have been applied to find more articles related to the initial research domains (cookies + XSS and web applications + XSS).

The iterations were run for each of the 73 seed articles, repeating the search process and sorting the results to select all related articles from 2018 to 2023.

Table 3.13 shows the related works found, taking the following criteria as analysis attributes: similar work, earlier work, and later work. Alternatively, the citations are relatively few for the works proposed in 2022 and 2023 because these works still need to be cited.

Table 3.13: Results after executing search using Snowballing

| Seed Papers | Similar work | Earlier work | Later work |
|-----------------------|---------------------|---------------------|---------------------|
| 48 (SS2) | 511 | 60 | 31 |
| 25 (SS1) | 554 | 93 | 209 |
| TOTAL FOUND | 1065 | 153 | 240 |
| TOTAL SELECTED | - | 15 (forward) | 51(backward) |

Final inclusion

In total, the seed articles (73), the articles found with backward snowball (51) and the articles found by applying forward snowball (15) were added, for an initial total of 139 articles.

The filter was applied to select all articles related to the research domains (SS1 and SS2), and those that were oriented to the analysis of XSS attacks in cloud technologies [128] [129], mobile, IoT [130] [131] and online social networking (OSN) [132] [133] were eliminated from the list.

Furthermore, as mentioned in the search strategy, only articles from the ACM Digital Library, IEEE Xplore, Springer Link, Web of Science, and Mendeley libraries were included; from the set of articles found in the snowball, those that did not belong to these scientific libraries were analyzed for but not included in our research, a total of 37 articles.

With this analysis, the final total number of articles selected for our systematic review was 96.

Study risk of bias assessment

In this context of computer security, the GRADE methodology has been applied to address the research question initially posed:

Research question: *¿What is the relationship between web application XSS attacks and the theft of cookies through XSS attacks?*

The GRADE methodology was applied to assess the quality of the selected evidence, the following assessment domains were considered:

Study design and execution limitations: We assessed the quality of studies included in this SLR regarding the tools used or suggested to detect or prevent XSS attacks in web applications (SS2). This assessment helped us to determine how effective and reliable these proposals are in addressing the XSS vulnerability.

Inconsistency: The evaluation included assessing the presentation of results across all included studies. Inconsistencies were identified for proposals that lacked result comparisons with other studies. This assessment aimed to ensure the completeness and coherence of the findings reported, enhancing the overall quality and reliability of the research outcomes.

Indirectness: The evaluation considered the population's relevance and the comparison with the research question. Although initially focused on the theft of information through cookie theft (SS1), to minimize bias, studies that applied alternative methods or techniques to steal information from other sources (web pages, websites, online infrastructures) were also considered relevant.

Publication bias: We incorporated studies found with the Research Rabbit Tool to mitigate publication bias. This inclusive approach broadened the search horizon, avoiding the exclusion of potentially valuable contributions.

3.2.2. Data collection process

We used the Rayyan Intelligent Systematic Literature software to analyze the full abstracts of articles and make annotations. These annotations helped us classify the content and gather attributes for further analysis.

To improve the quality of the information collected, we have analyzed the text and synthesized it using word clouds. This helped us to establish connections between important components of our research question such as web, cookies, XSS, and cross-site script, and

cross-site script. By doing so, we were able to determine which documents were relevant and pertinent to our research.

Using Atlas-TI software, we processed each study and generated a word list that visually represented the most commonly used terms in the paper. For example, we deduced that a study included techniques or methods that used JavaScript programming or solutions based on the frequency of occurrence of the words "XSS" and "JavaScript."

To rate the level of relationship of the selected studies to the main research question, the values detailed in Figures 3.8 and 3.9 were used. Through this analysis, our objective was to better understand the evolution of the approaches and technologies used that directly relate the three research domains we have proposed.

In Figure 3.8, we can see which works are closely tied to the research areas of XSS and Cookies. Meanwhile, Figure 3.9 displays the authors whose research is most relevant to XSS and Web analysis. Our novel proposal helps us determine if the chosen papers are related to our research question, ensuring the quality of our systematic analysis.

As a result of this analysis, it is observed that the proposals of authors such as Dembla et al., Mishra et al., Putthacharoen et al., and Takashi et al. are directly related to the three domains of our research (XSS + Cookies + Web).

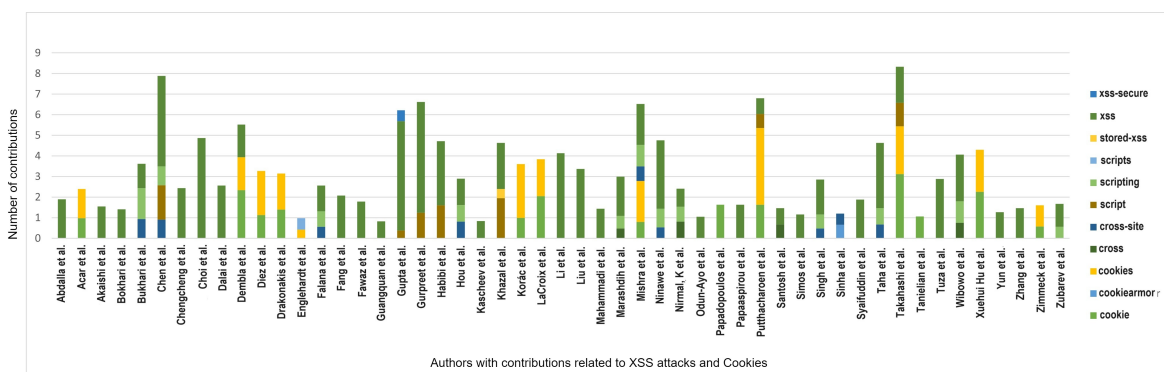


Figure 3.8: Overview of the authors whose proposals relate the research domains (cookies + XSS)

3.2.3. Methodology used to analyze the content of documents

This literature review aims to systematically answer a central research question to analyze the information collected in the selected articles. To improve the quality of the results, three complementary research questions RQA, RQB, and RQC, were formulated, which have been detailed in Table 3.14. The first question (RQA) analyzed all proposed methods or

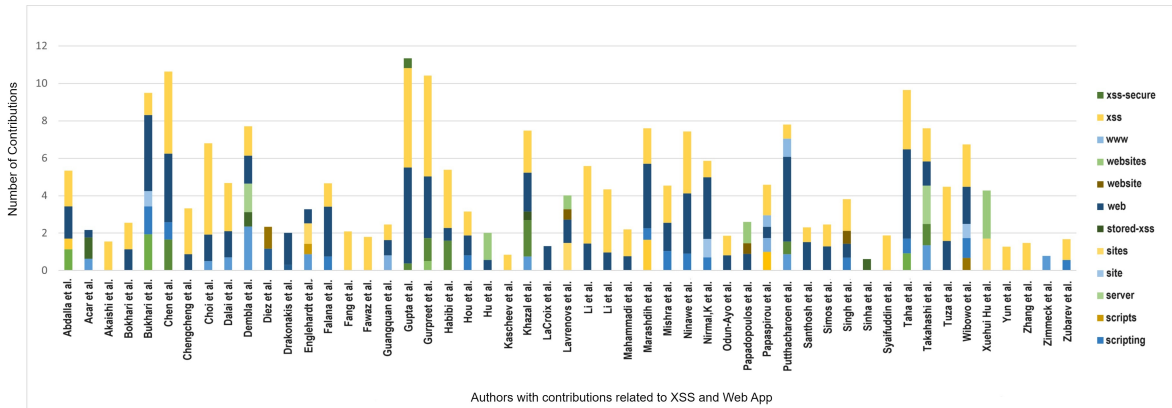


Figure 3.9: Overview of the authors whose proposals relate the research domains (web + XSS)

tools to detect or mitigate XSS attacks. With the second question (RQB), we investigated the different techniques presented to steal cookies through XSS attacks, and with the third question (RQC), we answered the question of how personal data is leaked through the theft of cookies executed through XSS attacks.

Table 3.14: Research questions to establish the problem to be solved.

| ID | Research Question |
|-----------------------------|---|
| Principal Research Question | ¿What is the relationship between web application XSS attacks and the theft of cookies through XSS attacks? |
| RQA | ¿What is the trend of methods and techniques proposed for detecting or mitigating XSS attacks? |
| RQB | ¿What techniques have been proposed for stealing cookies through XSS attacks? |
| RQC | ¿How our personal data is leaked through cookie theft executed by XSS attacks? |

The following attributes were configured to analyze the articles:

- **Type of proposal:** Variable to classify the type of study proposed (example: framework, experiment, theoretical study, practical study, laboratories, real scenarios, simulations, etc.)
- **C/S/H/D:** Variable to identify the type of XSS attack that the authors have tried to mitigate (Client - Reflect (C), Server - Stored (S), Hybrid(Client/Server) (H), DOM (D))

- **Goal:** General objective of the proposal
- **Orientation:** Analyze the orientation of the proposal (example: sites/programs/web applications, mission critical infrastructure, IoT systems, SCADA systems, cloud systems, etc.)
- **Methods/Tools:** Methods and tools used by the authors within their proposal
- **Limitations:** Variable to summarize the limitations that the author has encountered during his research process.
- **Relationship with the theft of cookies through XSS attacks:** Serves to determine whether the proposal is related to the leaking of personal information through cookie theft executed by XSS attacks.

3.2.4. Analysis of Results

Answering the Research Question A

RQA ¿ What is the trend of methods and techniques proposed for detecting or mitigating XSS attacks?

To support the information to answer this first research question, we analysed studies that proposed or used some tools or methods to detect or mitigate XSS attacks. We analysed the papers to look for trends from 2018 to 2023. We have applied the following methodology of analysis proposed in Figure 3.10 , in order to inspect, clean and transform the data and highlight the useful information that will serve to obtain the conclusions of our proposal.

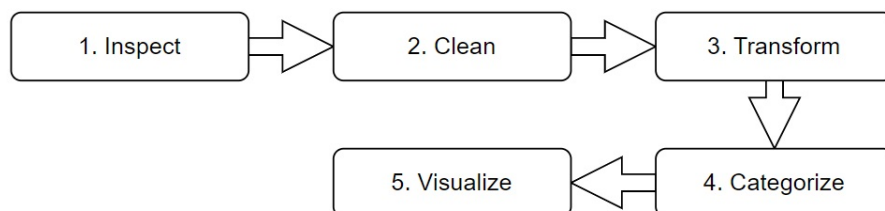


Figure 3.10: A suggested approach for analyzing content and organizing information.

- **Inspect:** in-depth reading of each scientific article, identification of orientation, identification of methods, tools, techniques to detect vulnerabilities or mitigate XSS attacks, identification of the type of XSS attacks. Determine if the proposal is related to cookie analysis or if it talks about cookie theft through XSS attacks, analysis of the limitations of all works.
- **Clean:** Identification of common tools/methods/techniques and goals for all scientific articles.
- **Transform:** Homologation of words (first capital letter, elimination of accents, elimination of singular and plural words).
- **Categorize:** Assignment of a category for all found values, e.g. JAVA and Python are categorised as Software Tools, IDS/IPS or antivirus are categorised as Security Tools.
- **Visualize:** Summary of results over time (2018 - 2023) using time diagrams.

Figure 3.11 shows an overview of the results categorized by years; in Table 3.15, the Top 5 most used tools are presented, among which Python and some of its libraries, Java and Javascript, and finally, PHP and some of its extensions stand out. On the other hand, in the Machine Learning-AI category, algorithms such as Multi-Layer Perceptron (MLP), Logistic Regression (LR), the Support Vector Machine (SVM), and tools such as word2vec, which is an algorithm used for natural language processing, stand out. In third place, we have categorized the analysis of web browsers, where the most used browsers are Google Chrome and Firefox, and proposals that use a browser called PhantomJS that allows testing browsing without headers. Fourth, we have categorized security tools, where tools such as XSSed, Ad-Blockers, Web Application Filter, and security equipment such as IDS/IPS stand out. In the last place of this top 5, we have placed the category Web Scrapers that allow web scraping to obtain data from websites vulnerable to XSS attacks; the most used tools are Selenium and BeautifulSoup.

In Figure 3.12, an analysis of all the tools within the Software Tools category was performed, establishing a word cloud to determine which tools have been most used to detect or mitigate XSS attacks from 2018 to 2023. It corroborates the information in Table 3.15 where it has been summarised that the most used tools were Python in conjunction with some libraries, Java, Javascript, PHP, and some of its extensions. 2019 was the year when there were more proposals for Python-based tools to detect or mitigate XSS attacks. On the

Table 3.15: Top 5 most used methods and tools for detecting and mitigating XSS attacks from 2018 to 2023

| Category | Tools/Method | Reference |
|-----------------------------|------------------------------|--|
| Machine learning algorithms | Multi-layer perceptron (MLP) | [134],[135],[136] |
| | Logistic Regression (LR) | [137], [138], [139],[140] |
| | Support Vector Machine (SVM) | [135],[141], [139],[140],[136] |
| | word2vec algorithm | [142], [137], [138], [139], [143] |
| Software tools | Python, Python libraries | [144], [145], [146], [147], [137], [148], [149], [150], [151], [152], [153], [143] |
| | JAVA, Javascript | [154], [155], [134], [156], [157], [158],[159], [160], [161], [162], [163], [164] |
| | PHP, PHP extensions | [165], [166], [167], [161], [168] |
| Web Browsers Analysis | Firefox, Chromium, tomJS | [144], [169], [145], [166], [157], [159], [167], [170], [171][172],[163], [164] |
| | AD-blockers | [172], [173] |
| Security tools | Web Application Filter (WAF) | [135],[140] |
| | IDS/IPS | [174], [140] |
| | XSSed tool | [175] |
| Web Scrapers | Selenium | [176],[150], [172],[136] |
| | BeautifulSoup | [151], [177] |

when there were more proposals for tools based on machine learning.

Figure 3.14 analyses all tools within the Web Browsers category, establishing a word cloud to determine the most used browser to analyze XSS attacks from 2018 to 2023. It corroborates the information in Table 3.15 where it has been summarised that the most used browsers to detect XSS attacks or vulnerabilities are Firefox in first place (more proposals are registered in the year 2018) and Chrome in second place (more proposals are registered

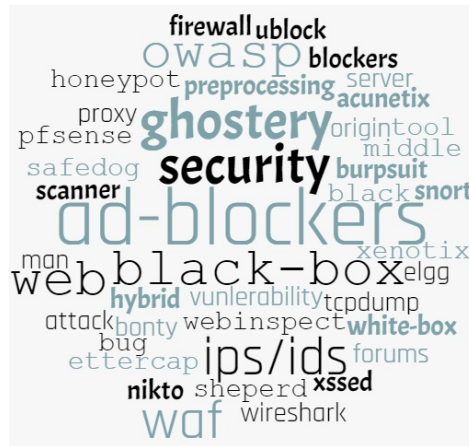


Figure 3.15: Most used tools within the Security Tools category

Beautifulsoup and Selenium are commonly used for analyzing XSS vulnerabilities. These tools can be combined with web-based data delivery methods such as GET and POST [142].



Figure 3.16: Most used tools within the Web Scrapers category

Figure 3.17 shows the distribution of proposed methodologies to detect and mitigate XSS attacks between 2018 and 2023, categorized by attack type: client-side, server-side, hybrid, and DOM-based. Among the analyzed contributions, 62% focused on client-side (C) attacks, 7.2% on both client-side and server-side (C/S) attacks, 6% on server-side (S) attacks, 4% on hybrid attacks, and 2.8% on DOM-based XSS attacks. A small percentage (6%) did not specify the targeted XSS attack type.

Figure 3.18 illustrates a popular approach to detecting or mitigating XSS attacks: examining website vulnerabilities based on their ranking within Alexa.com. This method represents 11% of the evaluated data. However, Alexa.com ceased operations on May 1, 2022, after over two decades of providing website traffic statistics. The list of base domains for finding XSS vulnerabilities was structured using the Top million websites ranking from articles in

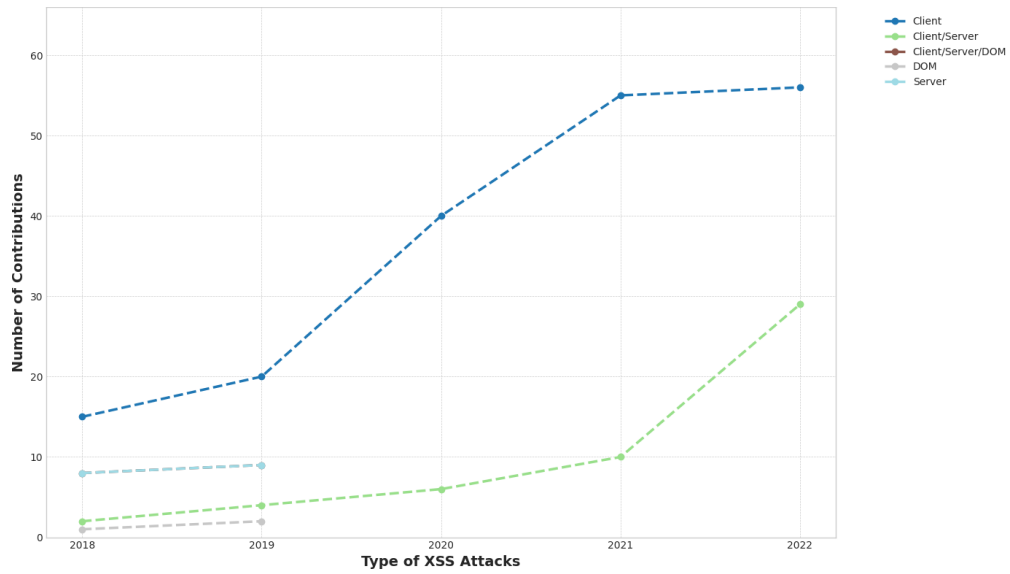


Figure 3.17: Trend of the analyzed works, evaluation of the attribute Type of XSS Attack

2018 [148] and 2022[176].

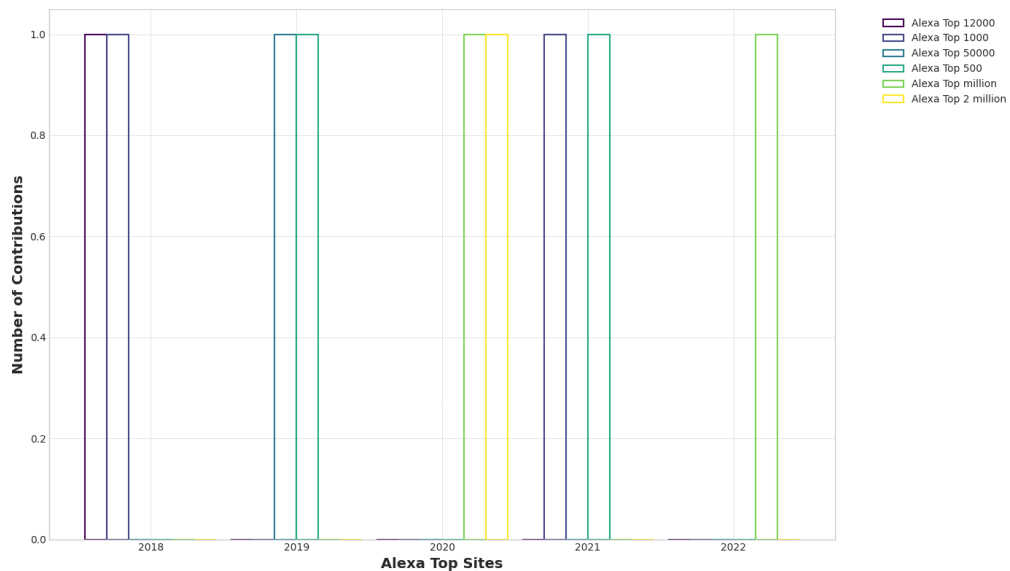


Figure 3.18: Analyzed works supported by domains registered in Alexa.com

We have also categorized authors' other tools or methods to detect/mitigate XSS vulnerabilities/attacks. For example, in the category **Browsers Tools**, the Lightbeam tool [178] stands out in 2019. In the category Hacking Tools, we found the use of the Kalilinux tool [153] in 2020. On the other hand, we have found studies based on analyzing users' browsing history [178]. We have grouped this type of tool in the category **History User Analysis**, and we have only found one proposal in 2019.

We have analyzed which operating system (**OS category**) type has been used in the XSS vulnerability detection/mitigation process in all the proposals. Windows [141], [167], [179], and Ubuntu-based systems [141], [147], [166], [143], [180], [181] have been highlighted. We have grouped proposals under the category **Repository**; in this group, the authors have used script repositories [156], benign and malicious samples of XSS vulnerabilities [149], vulnerability whitelists [163] and secure/insecure code blocks [168].

Answering the Research Question B

RQB ¿ What techniques have been proposed for stealing cookies through XSS attacks?

Table 3.16 below summarizes our analysis of proposals that have included cookie-related information. We have reviewed all relevant articles (27% only) that have studied or analyzed cookies within their objectives; we have determined the proposed techniques for stealing cookies through XSS attacks. We found that only a tiny percentage of the selected studies have considered cookie analysis, e.g., [178], which focuses on the analysis of third-party cookies, [182], which focuses on first-party cookies and synchronization. On the other hand, in [183], we have analyzed the synchronization of cookies when users browse different domains; in [160], a method for cookie rewriting as a method to avoid cookie theft was proposed. Finally, the works that stand out the most are those that have used repositories or cookie datasets in their proposals [170], [136], [138]. The study in [184] discusses cookie theft through XSS attacks. However, this proposal has only indicated that if the attacker cannot use the stolen cookies to impersonate the victim, then cookie theft is meaningless. Our research indicates that preventing cookie theft and misuse is the only current solution, highlighting the need for further investigation into effective theft techniques.

Answering the Research Question C

RQC ¿ How our personal data is filtering through cookie theft executed by XSS attacks?

Our research aims to find information about cookie theft using XSS attacks. However, beyond this primary objective, we have asked ourselves what type of personal or private information could be leaked if an attacker manages to steal our cookies (not only session cookies). A well-known technique for stealing cookies is the XSS attack, which, using javascript codes, can violate a victim's browser by executing malicious code from other sites and performing improper actions such as hijacking sessions, stealing cookies, injecting code, and remote control.

Table 3.16: Summary of the analysis to find information related to cookies

| Ref | Related to cookie analysis? |
|------------|--|
| [169] | Yes, In this proposal a Cryptography Local Proxy has been implemented, but it does not specify how some type of personal data is filtered through the theft of cookies. |
| [176] | Yes, but the scope of the proposal is the analysis of session cookies |
| [185] | Yes, through a lab to demonstrate the theft of session cookies. |
| [186] | Yes, but it is only a theoretical proposal and no information on data filtering is specified. |
| [187] | It only indicates that XSS attacks can steal cookies but does not specify the way |
| [188] | Yes, but it only relates to attacks on the Online Social Network (OSN) |
| [183] | The work has analyzed the synchronization of cookies but it does not deal with the theft of personal information through XSS attacks. |
| [148] | The authors only talk about analyzing the set-cookie header for session cookies, but they do not relate to the theft of personal information through XSS attacks. |
| [189] | Yes, but no solution has been proposed to mitigate cookie theft through XSS attacks. |
| [150] | The synchronization of cookies with third parties has been explained in the study, but privacy is a point that breaks with this research and has been little investigated. |
| [160] | No, only dynamic cookie rewriting has been specified to prevent spoofing |
| [138] | Cookies have been considered, but the authors do not speak of information filtering through XSS attacks. |
| [170] | The authors have mentioned cookies but not the leaking of personal information through XSS attacks |
| [190] | The authors mention the study of cookies, but the tests and results obtained have not been evidenced. |
| [171] | Yes, but does not specify how personal information is leaked through XSS attacks. |
| [172] | It only talks about the synchronization of cookies in a third-party ecosystem. |

Our analysis has shown that certain methods have been proposed to enhance web security. For example, an article in [169] suggests the use of a Local Cryptography Proxy that encrypts cookies each time a user accesses a website. This technique prevents stolen cookies from being reused by replacing them with encrypted ones. Similarly, [160] proposes dynamic cookie rewriting as a way to prevent spoofing.

We have found a research gap that opens the door to many possibilities, including letting end users know how their personal information is leaked by cookie theft through XSS attacks, which can come from experienced attackers or third-party domains. Moreover, this personal information refers not only to session cookies, as the literature suggests, but also to cookies, for example, marketing, which is connected each time we search for a product and another web page suggests similar products.

One major concern is the type of personal information that is being shared through cookies. Is it possible that one's name, last name, location, expenses, credit card information, and preferences are being shared? Additionally, it's interesting to see how various domains that have nothing to do with email are interconnected on an email page, as shown by different web browsers.

Answering the Principal Research Question

XSS attacks on websites refer to the malicious insertion of code into web pages to compromise the security of users visiting those pages. On the other hand, XSS attacks to steal cookies involve exploiting vulnerabilities in browsers to gain unauthorized access to cookies stored in the user's browser.

With the support of the 3 research questions - RQA, RQB and RQC, our Systematic Literature Review revealed a research gap on personal data filtering to compare the effectiveness of XSS attacks on web applications versus cookie theft. Two search strings were used to find related studies, one targeting filtering XSS + cookies, and the other filtering XSS + websites.

The primary research question focused on the relationship between XSS attacks on web applications and the theft of cookies through XSS attacks. The analysis involved careful examination of submissions and extensive research to categorize the data. The goal was to identify connections between XSS attacks on web pages and those targeting cookies. Categories such as Web Browser Tools, Social Networks, URL Analysis, Web Browser Analysis, and Web Scrapers vs Cookie Analysis were examined to identify trends. Figure 3.19 illustrates the trends of each category and their relationship from 2018 to 2023. The research

revealed a clear trend, indicating that attacks targeting websites outnumber those aimed at stealing cookies, with a ratio of 5:1 for XSS attacks on websites to cookie theft through XSS attacks.

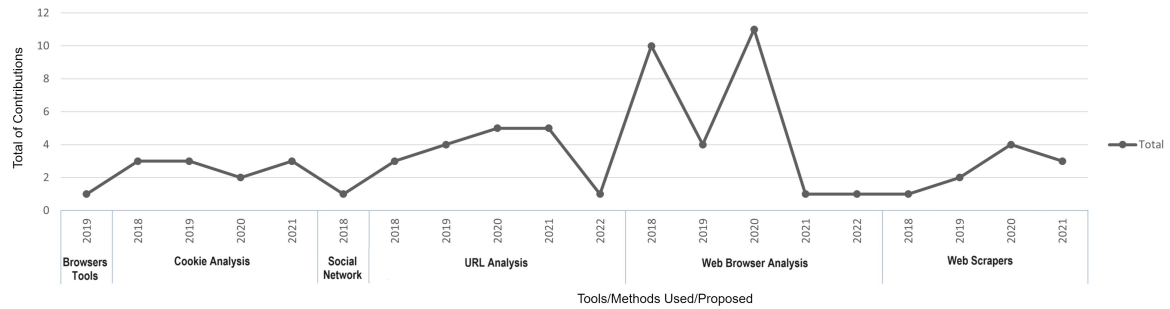


Figure 3.19: Comparison between XSS attacks aimed at web applications and XSS attacks aimed at cookie theft

A clear trend characterizes the relationship between the two variants of XSS attacks: attacks targeting websites are higher than attacks designed to steal cookies. The ratio of XSS attacks on websites vs. cookie theft through XSS attacks is 5:1.

Chapter 4

The roadmap to the BOOKIE framework

Índice

| | | |
|-------|---|-----|
| 4.1 | PHASE A: CHARACTERIZATION | 68 |
| 4.1.1 | Sub Phase 1: XSS Attacks | 68 |
| 4.1.2 | Sub Phase 2: Cookie Behaviour | 77 |
| 4.1.3 | Sub Phase 3: Cookie Theft | 88 |
| 4.1.4 | Sub Phase 4A. Personal Data Filtering | 96 |
| 4.1.5 | Sub Phase 4B: XSS2DENT | 104 |
| 4.2 | PHASE B: REPLICATE AND COLLECT | 113 |
| 4.2.1 | Cookie Collector Phase 1 | 113 |
| 4.2.2 | Cookie Collector Phase 2 | 116 |
| 4.3 | PHASE C: TRAINING AND PREDICT | 124 |
| 4.3.1 | Training | 124 |
| 4.3.2 | Prediction | 126 |
| 4.4 | PHASE D: VISUALIZATION | 134 |
| 4.4.1 | Methodological proposal through a pen testing challenge | 134 |
| 4.4.2 | Using statistical methodology to analyze computer security challenge results. | 136 |
| 4.4.3 | Pen testing results | 137 |
| 4.4.4 | Knowledge assessment results | 139 |

4.1. PHASE A: CHARACTERIZATION

To understand how the framework was structured, this chapter explains the entire path from when the idea was born to when it was implemented. BOOKIE is a framework to teach how personal data can be leaked by stealing cookies using XSS attacks. It comprises four main phases, as shown in Figure 4.1. In each of the phases, the research and experimental process carried out are explained.

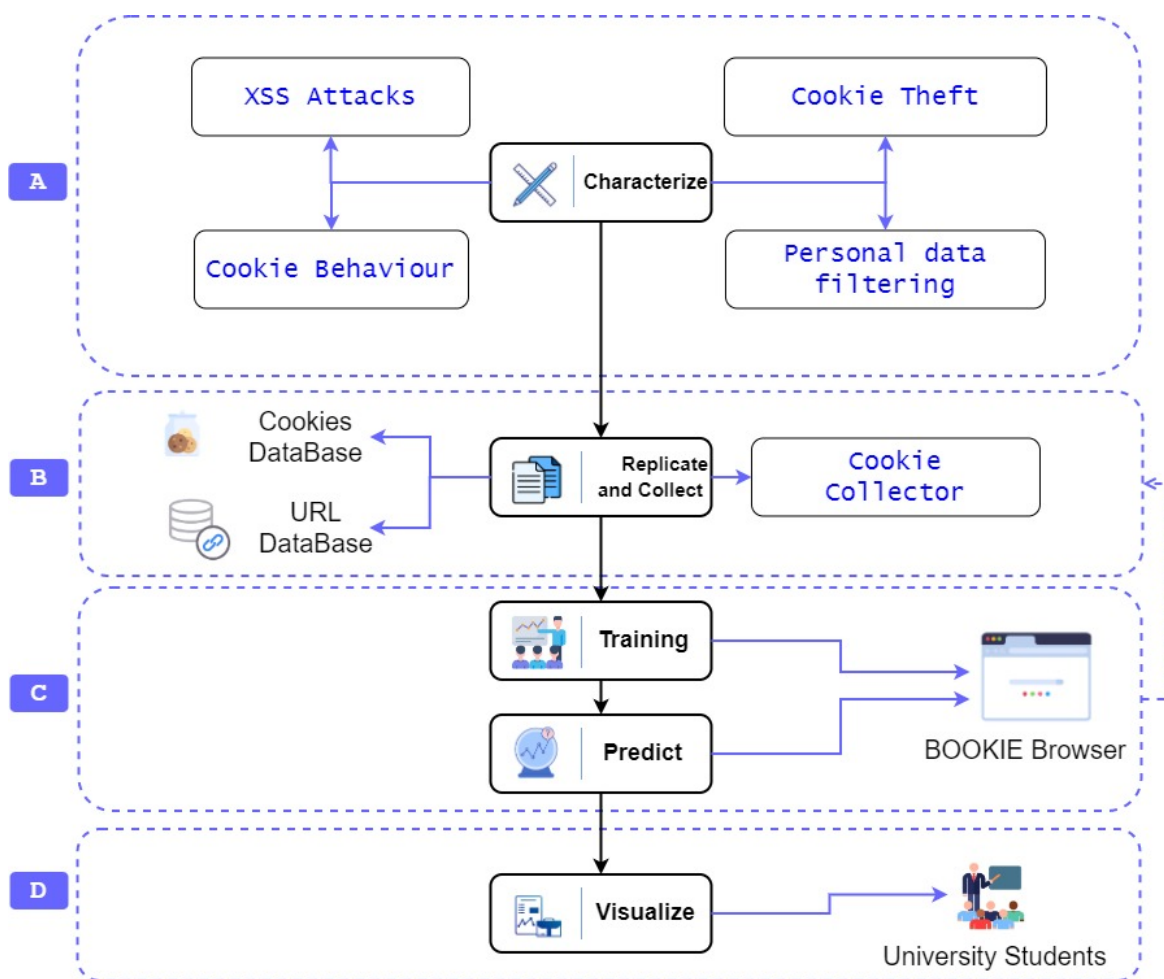


Figure 4.1: BOOKIE: A framework to teach university students about filtering personal data by stealing cookies via xss attacks

4.1.1. Sub Phase 1: XSS Attacks

CookieScout is based on the analysis of the cookies that are created when a user visits different websites. It defines a new approach for analyzing these sites and avoiding XSS attacks, reducing the likelihood of users visiting compromised websites. The model will

prevent the storage of suspicious cookies, the theft of personal information, or the hijacking of these session files. CookieScout has been structured in three parts (Figure 4.2): data collection, data analysis, and the analytical model (algorithm).

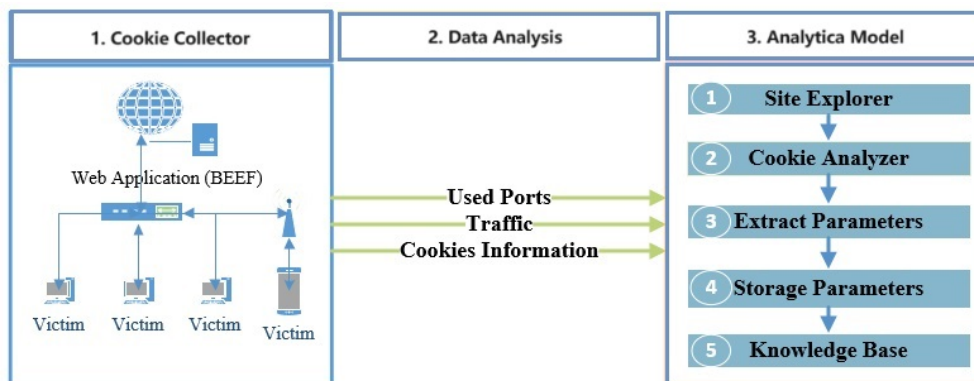


Figure 4.2: CookieScout: An analytic model for prevention of Cross-Site Scripting

Data Collection

For the data collection, two scenarios were proposed for executing the XSS attack: a) within a domain infrastructure and b) in a controlled infrastructure. The Beef framework [191] was used for both cases, which involves identifying the victims and analyzing the traffic they exchange with the attacker. The goal was to analyze the number of packets sent and received, the ports used, and the behavior of cookies created. This first part defined the variables used later in the algorithm..

Execution of the XSS attack

The successful implementation of the attack was using social engineering. A link was configured so that the home page of the client's browsers changed to a modified web page with information that allowed the execution of the attack. The hook was a JavaScript code-named `hook.js` hosted in the same framework.

When executed, it calls the server where Beef Framework is running and sends information about the victim. Once the victim's computer was compromised, it was possible to execute remote commands and additional modules against the victim, including the automatic play of a sound or a pop-up asking about the need to update the Adobe Flash Player plug-in. When the client clicks the web page, it commits itself, and all the machine is recorded in a tree of hooked browsers.

The attack was carried out under two scenarios: a domain infrastructure (`inec.gob.ec`)

and a controlled infrastructure (virtual machines). In the first scenario, the INEC domain server rules configuring the default home page in the users' browser were modified; this domain server was based on the Windows Server operating system. URLs or home pages have been configured in the Group Policy Management settings so that certain users can see the page contaminated with XSS codes as their home page. In the second scenario, virtual machines were configured in Windows and Linux to simulate a real scenario and avoid the invasion of the privacy of real users.

In technical terms, the hook refers to a user interacting with a button or link on a modified web page. This interaction sends the computer into a zombie state, where an external framework can remotely monitor it. The bait website was hosted on the attacking machine's server for testing, accessed through this link: *192.168.10.X:3000/demos/basic.html*.

In the given scenario, users who lacked basic security knowledge trusted their browser's homepage without realizing that it was modified. We used Wireshark and Tshark tools to collect the exchanged traffic, where Wireshark helped us analyze all the data obtained on Windows clients visually while Tshark helped us collect traffic in text mode on Linux clients. We were able to obtain information from the cookie created by the Beef framework, as shown in Table 4.1.

Table 4.1: Cookie information created when a user accesses the vulnerable web page

| Parameter | Description |
|----------------------|-------------------------|
| Name | BEEFHOOK |
| Content | ffMLBvDUSJufXhciehMv... |
| Domain | 192.168.10.X |
| Path | / |
| Creation Date | June 2017 |
| Expiration Date | December 2030 |
| Script Accessibility | Yes |
| Certificate | No |

Following the attack, we conducted a thorough analysis to confirm that popular antivirus solutions, including McAfee, Avast, Nod32, and Norton Security, failed to detect any suspicious activity related to modifying the test homepage. Notably, no warning or alert was triggered during the attack, indicating that it went completely unnoticed. Consequently, all users could connect to the Beef Framework.

Data Analysis

We analyzed the **traffic** between the victims and the attacker by counting the number of exchanged packets in Table 4.2. It was found that this traffic was consistently higher than other website traffic.

Table 4.2: Summary of packages sent/received in the attack (A) towards each victim (V)

| Victim | Total | A → V | V → A |
|----------|---------|---------|---------|
| Victim 1 | 187 Kb | 100 Kb | 87 Kb |
| Victim 2 | 1024 Kb | 757 Kb | 447 Kb |
| Victim 3 | 2076 Kb | 1043 Kb | 1033 Kb |
| Victim 4 | 1098 Kb | 1075 Kb | 833 Kb |

During the attack, the **network ports** used were randomized but sequentially to connect the victims to the framework. As shown in Table 4.3, the destination port varied for each victim, while the source port remained constant at 3000.

A **cookie** has a name and identifier that can be created, modified, or deleted on the client and server. It's sent in HTTP requests and some cookies can have the same name across different websites.

To evaluate how much information a user exchanges with visited sites, traffic analysis was used. Attacker ports are found to be random, making it difficult to detect attacks, and attackers can hide their ports using tools.

The traffic analysis is a reference point to evaluate the amount of information a user exchanges with the sites visited. This implies the consumption of time and the high use of resources. Table 4.5 shows that the ports used to attack the victims are random. Nowadays,

Table 4.3: Summary of ports used in the attack (A) on each victim (V)

| Victim | Source Port (A) | Destination port (V) |
|----------|--------------------|-------------------------|
| Victim 1 | 3000 | 1391 to 1398 |
| Victim 2 | 3000 | 2166 to 2188 |
| Victim 3 | 3000 | 52300 to 52305 |
| Victim 4 | 3000 | 35616 to 35621 |

some tools allow the hiding of ports when executing an attack. That is why this parameter was not included in our algorithm design.

With this analysis, the following parameters of the cookie were chosen and then structured to present our algorithm proposal:

- *Name*: is the name with which the cookie is created.
- *Site*: is the domain or web page visited by the user.
- *Creation date*: It is the date recorded when the user visited a web page.
- *Expiration date*: is the date that indicates when the cookie expires.
- *Execution of commands*: is a property that indicates if it allows to execute commands or not.

Analytical Model

A flowchart in Figure 4.3 was presented based on structured data to classify suspicious cookies. In addition, browsing tests on suspicious sites were performed to gather more information about cookies and compare it with the data. The algorithm runs for each website visited and each cookie created by it, as described in Algorithm 1.

- **Site Explorer**: scans all websites visited by the client, covers the entire web browsing information of each user.
- **Cookie analyser**: analyses the cookies generated by each website visited. According to the results obtained from the data collection, most websites generated 1, 2, 5, or even 31 cookies; therefore, it is a requirement to analyze each website visited to identify the number of cookies created.
- **Extract parameters**: After analyzing the website and its cookies, the following parameters are extracted:
 - *Cookie Name* - [*CookieName*]: it does not correspond to a known name and can be a combination of symbols, letters, numbers or all. It is defined as a string of text.

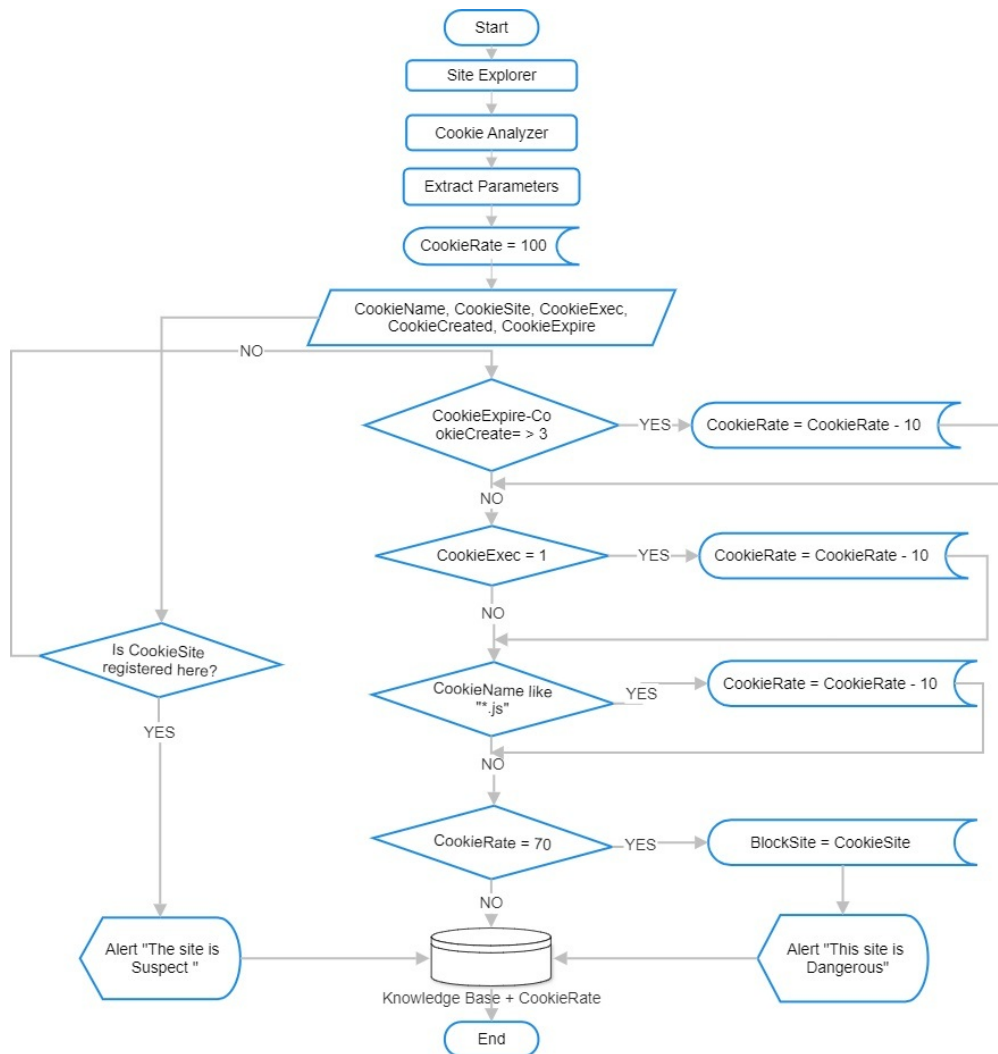


Figure 4.3: Flowchart to rate the reputation of cookies according to their attributes

- *Domain* - [*CookieSite*]: The website that generated the cookie can generate 1 to 31 or more cookies. They can be first-person (own domain) or third-person (third-party domains).
- *Accessibility* - [*CookieExec*]: a boolean variable was identified for the execution of commands, this can be of type: hostOnly cookie, session cookie, secure cookie or httpOnly cookie.
- *Creation date* - [*CookieCreate*]: Corresponds to the day, month, year, and time the cookie was created. It is assigned a variable for the type of date.
- *Expiration date* - [*CookieExpire*]: Cookie expiration dates include day, month, year, and time. It is assigned a variable for the type of date.

In addition, the following variables were declared:

- *[i]*: to count the cookies for each website.
 - *[j]*: to scroll through the entire list of visited websites.
 - *[CookieRate]*: It starts with a value of 100 to qualify the cookie's reputation. This value means that it is reliable.
 - *[BlockSite]*: This variable will be assigned to a website when the cookie has a low reputation. Later, it will keep track of the websites that must be blocked.
- **Storage parameters:** It is responsible for storing the variables, according to the information of the parameters found in the websites visited, with their respective cookies. The following variables were registered: *[CookieName]*, *[CookieSite]*, *[CookieExec]*, *[CookieCreate]*, *[CookieExpire]*.
 - **Conditions:** subsequently, a set of conditions were executed:
 - “*If its difference is greater than 3*”. This condition was run to compare the creation and expiration years of the cookies. Ten points are subtracted from the variable *[CookieRate]* that qualifies the reputation; otherwise, the following condition is executed.
 - “*If the [CookieExec] variable is equal to 1*”. The reputation variable is decreased by 10 points.
 - “*If the cookie name ends in *.js*”. The cookie is a JavaScript type and 10 points are subtracted from the variable where the reputation is stored..
 - “*If the cookie's reputation is less than or equal to 70*”, when a cookie meets all three conditions, it gets 70 points and its site is stored as *BlockSite*. The user is then alerted about it.

For cookies that meet one or two conditions, a gray list is created to qualify them as suspicious. Later this information is stored in the *Knowledge Base*.

- **Knowledge Base:** The algorithm analyzes cookies sequentially. It checks if the visited website is on the list of blocked or suspicious websites in its knowledge base. If yes, it alerts the user on their second visit. If not, it continues with the algorithm's sequence.

Algorithm 1 Algorithm flow diagram proposed

Input: *CookieName* : String
Input: *CookieSite* : String
Input: *CookieExec* : Boolean
Input: *CookieCreate* : Date
Input: *CookieExpire* : Date
Input: *CookieNum* : Numeric
Input: *TotalSiteVisited* : Numeric
Initialize: *CookieRate* \leftarrow 100
Procedure SITE EXPLORER: Count all the sites visited by the user and assign the value to *TotalSiteVisited*
Procedure COOKIE ANALYSER: Count the number of cookies per site visited and assign it to *CookieNum*
Procedure EXTRACT PARAMETERS: Extract the properties of cookies created
Procedure STORAGE PARAMETERS: Save the properties of cookies created
Select: *CookieName, CookieSite, CookieExec, CookieCreate, CookieExpire*
if *CookieSite* is in Knowledge Base **then**
 ALERT: "This site *CookieSite* is suspect"
else
 $j \leftarrow 0$
 while $j < TotalSiteVisited$ **do**
 Count the number of cookies per site visited and assign it to *CookieNum*
 if *CookieNum* > 1 **then**
 $i \leftarrow 0$
 repeat
 if $CookieExpire - CookieCreate > 3$ **or** *CookieExec* **then**
 $CookieRate \leftarrow CookieRate - 10$
 else if *CookieName* ends with ".js" **then**
 $CookieRate \leftarrow CookieRate - 10$
 else
 $i \leftarrow i + 1$
 end if
 until $i \geq CookieNum - 1$
 end if
 if $CookieRate \leq 70$ **then**
 Block *CookieSite*
 ALERT: "This site *CookieSite* is dangerous"
 end if
 $j \leftarrow j + 1$
 end while
end if

The results show that cookies can have the same name for domains or websites visited. A curious fact is the number of cookies generated per site. Tests reveal that a site can generate more than one cookie, with 31 being the most significant number.

Within the set of generated cookies, a subset had the "local storage" attribute, indicating that they can persistently store data on the user's computer. This data may include the user's most recent visit to the corresponding website.

By comparing the cookies generated by suspicious and unreliable websites, it was found that certain websites generated cookies with the name *Facebook.com*, which means that an attacker could steal credentials information if we navigate with an open session of social networks. Other more dangerous websites generated a cookie similar to the one of the framework Beef, with the name *cookies.js*.

The creation and expiration dates of cookies were a good reference to determine which ones were suspicious. The analysis of the websites in Figure 4.4 shows that 29.41% of the websites create cookies with an expiration year greater than 3. The most common years found were 2027, 2075 and 2077.

Comparing with the cookie created in the attack using the framework Beef (13 years of expiration which helped to define the variables *CookieCreate* and *CookieExpire*), we can say that this property was fundamental and the basis for designing the algorithm, since we compared the cookie that creates an ethical hacking program vs a cookie created by a suspicious website. The average found for the expiration date of the cookies was only 2 years. With this information, we classified as dangerous cookies those ones that exceed 3 years of expiration.



Figure 4.4: Statistics of the expiration date of the cookies created

The choice of the name attribute is also crucial in the analysis since it was found that the names of all cookies have no meaning. We found names with a single character, which allows us to analyze cookies with an additional process; we had names like *cookies.js* similar to the JavaScript file that Beef Framework uses to hook victims. Then, the cookie name was included as a variable due to the extension, equivalent to a script that can be exploited in the XSS attack.

Cookies that allow the execution of commands, shown in Figure 4.5, were also part of the algorithm. From an initial test of 17 visited websites, 88.24% of the cookies allow

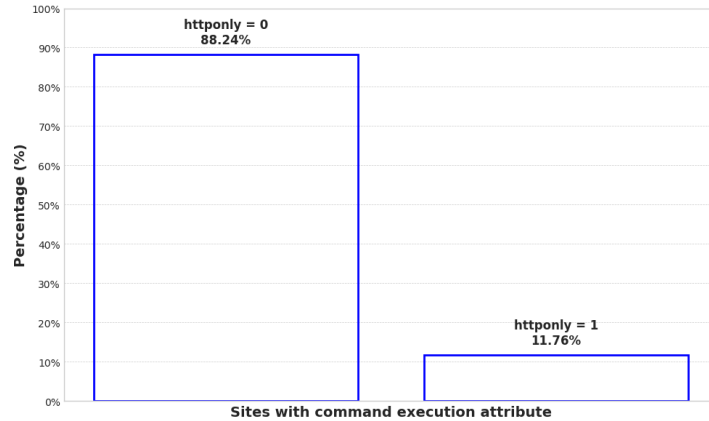


Figure 4.5: Cookies created that allow the execution of commands

the execution of commands, and only 11.76% of the websites created cookies of the type *httpOnly=1*, which It means that it does not allow the execution of XSS attacks.

The algorithm has a sub process that creates a knowledge base for visited websites. This database includes their cookies and reputation levels, which adds value to Phase 1. The algorithm can then compare new websites to the database before analyzing them.

4.1.2. Sub Phase 2: Cookie Behaviour

The DataCookie model has been proposed using the CRISP-DM methodology as a framework to execute the data mining process. The primary objective of this proposal was to construct a model that can categorize cookies based on various attributes extracted from users' navigation databases. DataCookie consists of 6 phases, shown in Figure 4.6:

A. System Setup

This analysis was executed in a public institution. The navigation history of 400 users during a typical workday was obtained to analyze all the domains and sub-domains generated; the WSA (Cisco Ironport) Web Security Control data was used. The privacy of the users was not affected because, as network administrators, we can review the browsing history of the users to create new rules or check the non-compliance of the already configured ones.

B. Processing and cleaning

This first record contains the browsing history of all users. As shown in Fig. 4.7, it contains the URL, URL Category, Disposition, and Threat Type fields. According to the preset rules

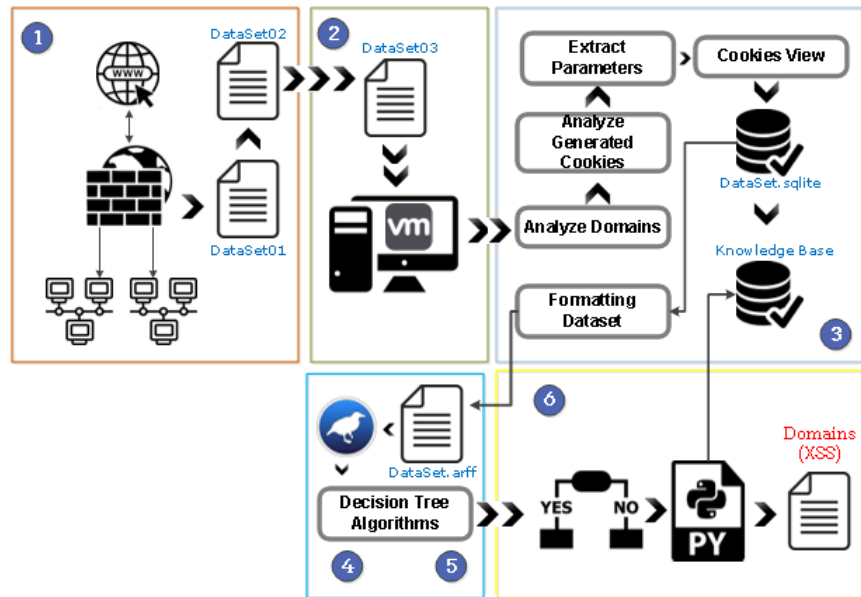


Figure 4.6: DataCookie: Proposed Model for Obtaining Rules from Cookie Mining

of the appliance, there are 77 navigation categories. A total of 231094 records were found.

| URL | URL CATEGORY | Disposition | Threat Type |
|-----------------------------|----------------------|-----------------|--------------|
| http://178.255.156.106/doc | Infrastructure and C | Allow | - |
| "https://tt.onthe.io:443/?k | Business and Indust | Allow | othermalware |
| http://ctldl.windowsupdat | Software Updates | Block - URL Cat | - |
| https://a.wunderlist.com:4 | Computers and Inte | Allow | - |
| "https://graph.facebook.co | Social Networking | Block - URL Cat | - |
| http://comcluster.cxense.c | Computers and Inte | Allow | - |
| http://ds.download.windo | Software Updates | Allow | - |
| https://4-edge-chat.facebo | Social Networking | Allow | - |
| https://microsoft.stream1. | Infrastructure and C | Allow | - |

Figure 4.7: Resume of records obtained from the Web Security Appliance (DataSet01)

The first dataset, textbfDataSet01, was cleaned by a Python script to obtain unique domains. The second data set, **DataSet02**, had 134951 records. The categories were presented among layout rules: allow, block and monitor.

A dataset called **DataSet03** was filtered based on the Disposition and Threat Type fields 4.8. The resulting dataset had 6432 records, which were then reduced to 5113 after eliminating duplicates. During this process, it was discovered that 8 out of 77 categories allowed access to pages with threats. These categories were Advertisements, Astrology, Block, File Transfer, Games, Hacking, Illegal Activities, and Uncategorized.

The next step was determining if any of these domains had XSS vulnerability. The cookies generated by these domains in a virtual machine configured with a Windows 7 operating

| URL | URL CATEGORY | Disposition | Threat Type |
|---|----------------------|-------------|--------------|
| https://settings.luckyorange.net:4 | Infrastructure and C | Allow | adware |
| "https://tt.onthe.io:443/?k[]=3937 | Business and Indust | Allow | othermalware |
| https://static.ads-twitter.com:443/ | Advertisements | Allow | adware |
| https://settings.luckyorange.net:4 | Infrastructure and C | Allow | adware |
| http://cdn.luckyorange.com/w.js | Computers and Inte | Allow | othermalware |
| "https://tt.onthe.io:443/?k[]=3937 | Business and Indust | Allow | othermalware |
| https://settings.luckyorange.net:4 | Infrastructure and C | Allow | adware |
| "https://tt.onthe.io:443/?k[]=3937 | Business and Indust | Allow | othermalware |
| https://static.ads-twitter.com:443/ | Advertisements | Allow | adware |
| https://settings.luckyorange.net:4 | Infrastructure and C | Allow | adware |

Figure 4.8: Resume of records obtained to structure the DataSet03

system with 1GB of RAM and one processor were analyzed. The NAT mode in the network configuration was set to protect the virtual environment. This machine was used to manually access all the domains of the **DataSet03** through the Mozilla Firefox browser. As a result, 1666 cookies were obtained. Not all domains in the dataset generate cookies because they correspond to domains that require a username and password to log in, such as login pages (session cookies).

C. Data Analysis

The RFC 6265 (HTTP State Management Mechanism) provides the normative reference to interpret each parameter of the cookies:

Name: the name of cookie

Value: the value of the cookie. This value is stored on the client computer. It does not store sensitive information.

Expire: the expiration time of the cookie. This is a UNIX time stamp. It is a number represented in seconds.

Path: the path on the server where the cookie will be available. If set to the root (/), the cookie will be available throughout the domain.

Domain: the domain for which the cookie is available. By setting this to a sub domain (www.example.com), the cookie will be available for that sub domain and all other sub domains within this (w2.www.example.com).

Secure: shows that the cookie should only be transmitted over a secure connection from the client. When set to TRUE, the cookie will only be set if a secure connection exists.

HttpOnly: when set to TRUE, the cookie will only be accessible through the HTTP protocol. This means that the cookie will not be accessible by scripting languages, such

as JavaScript. This configuration can effectively help to reduce identity theft through XSS attacks (although it does not support all browsers).

To complete this phase, CookieView and SQLite Studio tools were used. The first one analyzes the parameters of registered cookies acting as a cookie viewer. With the second tool, a central database (**DataSet.sqlite**) was structured to perform the mining work.

Initially, these values were exported to a minable data set. The main fields selected were: *domain*, *name*, *value*, *isSecure* and *isHttpOnly*. Because the original fields corresponding to *ExpirationDate*, *LastAccess* and *CreationDate* were in Coordinated Universal Time (UTC) format, we did an operation to transform to format *dd/mm/yr - hh:mm:ss*. However, only the *ExpirationDate* field was used, and an operation was added to have a value of TRUE if the expiration date was greater than two years; otherwise, FALSE.

D. Modeling

In order to analyze this data in Weka, the records of the data set were exported to .CSV format and later edited in a notepad to format them as follows:

The records that met the conditions proposed in the conceptual model of Phase 1 were classified as Suspicious. Finally, the file was saved formatted with the extension *.arff, which is the format supported by the Weka software. The preview of the final data set is shown in Figure 4.9.

```
1 %COOKIES CLASIFIER FOR XSS PREVENTION
2 @relation 'Clasificador de Cookies'
3
4 %PARAMETROS DE LAS COOKIES DE ACUERDO A RFC
5 @attribute Categoria {Advertisements,Astrology,
6 @attribute Path {root,noroot}
7 @attribute isSecure {TRUE,FALSE}
8 @attribute isHttpOnly {TRUE,FALSE}
9 @attribute Mayor2A {TRUE,FALSE}
10 @attribute Sospechoso {YES,NO}
11
12 %DATOS DE LAS COOKIES RECOGIDAS
13 @data
14 'Illegal Activities',root,FALSE,TRUE,FALSE,YES
15 'Illegal Activities',root,FALSE,FALSE,FALSE,YES
16 'Illegal Activities',root,FALSE,FALSE,FALSE,YES
17 'Illegal Activities',root,FALSE,FALSE,FALSE,YES
18 'Illegal Activities',root,FALSE,FALSE,FALSE,YES
```

Figure 4.9: Preview of DataSet.arff file formatted for analysis using Weka Software

Table 4.5 presents a summary of the description of the attributes and the values that

Table 4.4: Summary of packages sent/received in the attack (A) towards each victim (V)

| Type | Name | Value | Description |
|------------|------------|-------------|---|
| @attribute | Category | - | New database column added for cookie-based class establishment by domain categories. |
| @attribute | Path | root,noroot | Cookie saved in root directory or sub-directory within it. |
| @attribute | isSecure | TRUE/FALSE | Cookie sent via HTTP (FALSE) or HTTPS (TRUE). Original values: 0 and 1. |
| @attribute | isHttpOnly | TRUE/FALSE | Value is TRUE if XSS attacks are not allowed; otherwise, it's FALSE. Initial values were 0 and 1. |
| @attribute | Mayor2A | TRUE/FALSE | The expiration date will be TRUE if it exceeds two years and FALSE if it doesn't. |
| @attribute | Sospechoso | YES/NO | Redirections between domains or files with .js extension will mark the cookie as suspicious (value: YES). |

have been established in the DataSet.arff.

In the next phase, the following algorithms in Weka software were selected to run data mining:

- RepTree
- J48
- RandomForest
- DesicionStump
- HoeffdingTree
- LMT
- RandomTree

We compared the models in the Tree section of the software to analyze their performance

Table 4.5: Final DataSet.arff Attribute Description

| Attribute | Description | Possible Values |
|------------|---|--|
| Category | Set the classes according to the cookies generated by the corresponding domains | Advertisements, Astrology, Block, File Transfer, Service, Games, Hacking, Illegal Activities, Uncategorized URL,etc. |
| Path | If the cookie is saved in the root directory | root, noroot |
| isSecure | If the cookie is sent over HTTP or HTTPS | TRUE, FALSE |
| isHttpOnly | If it does not allow XSS attacks | TRUE, FALSE |
| High2A | If expiration date is greater than 2 years | TRUE, FALSE |
| Suspect | If there is redirection between domains, or have JavaScript files | YES, NO |

and choose the best algorithm. The decision tree's rules were then extracted to classify new cookies.

The objective of the selected algorithm was to predict if a new cookie is suspicious, and we wanted to know how good the prediction would be with future data. We used the Cross-Validation technique with 10 Iterations. This technique aims to overcome the problem of over-fitting and make predictions more general. Thus, the dataset was divided into ten equal parts (folds); each was used once for testing and nine times for training.

E. Performance Evaluation

We evaluated and compared Weka's decision trees performance based on selected analysis parameters.

- Correctly Classified Instances (Correctly)
- Incorrectly Classified Instances (Incorrectly)
- True Positive Rate (TP Rate)
- False Positive Rate (FP Rate)

- Precision
- ROC Area (AUC)

Accuracy of a model is determined by the number of correctly and incorrectly classified instances. True Positives (TP) and True Negatives (TN) are correctly classified, whereas False Positives (FP) and False Negatives (FN) are incorrectly classified. Precision is calculated by dividing the total number of correctly classified instances by the total number of instances. Accuracy is the proportion of positive cases that were correctly predicted.

Executed Trees algorithms collected data on variables such as Correctly/Incorrectly Classified Instances, Mean Absolute/Squared Error, Precision, and Model build time. The results have been summarized in the Table 4.6

Table 4.6: Summary of results of classification tree algorithms

| | Correctly Classified Instances | Incorrectly Classified Instances | Mean Absolute Error | Abso-Root Squared Error | Mean Relative Absolute Error | Relative Abso-Squared Error | Precision | Time to build model |
|----------------------|--------------------------------|----------------------------------|---------------------|-------------------------|------------------------------|-----------------------------|-----------|---------------------|
| RepTree | | | | | | | | |
| Results | 1578 | 88 | 0.0978 | 0.2233 | | | | 0.01s |
| Results(%) | 94.7179% | 5.2821% | | | 34.6592% | 59.4562% | 94.8% | |
| J48 | | | | | | | | |
| Results | 1583 | 83 | 0.0942 | 0.2172 | | | | 0.04s |
| Results(%) | 95.018% | 4.982% | | | 33.3844% | 57.834% | 95.2% | |
| Random Forest | | | | | | | | |
| Results | 1583 | 83 | 0.0898 | 0.2127 | | | | 0.15s |
| Results(%) | 95.018% | 4.982% | | | 31.8139% | 56.6393% | 95.2% | |
| DecisionStump | | | | | | | | |
| Results | 1583 | 83 | 0.0968 | 0.2131 | | | | 0.68s |
| Results(%) | 95.018% | 4.98% | | | 34.28% | 56.75% | 94.7% | |
| HoeffdingTree | | | | | | | | |
| Results | 1581 | 85 | 0.0967 | 0.2195 | | | | 0.02s |
| Results(%) | 94.89% | 5.102% | | | 34.24% | 58.45% | 95.1% | |
| LMT | | | | | | | | |
| Results | 1583 | 83 | 0.0968 | 0.2131 | | | | 0.68s |
| Results(%) | 95.018% | 4.98% | | | 34.28% | 56.75% | 95.2% | |
| RandomTree | | | | | | | | |
| Results | 1581 | 85 | 0.08 | 0.2123 | | | | 0.1s |
| Results(%) | 94.89% | 5.10% | | | 31.32% | 56.52% | 95.1% | |

In Tables 4.7 and 4.8 we present the results of the values obtained for each decision tree model after running the tests with the **DataSet.arff**.

The receiver operating characteristic curve (ROC) are two-dimensional graphs plotting the actual positive rate (TPR) on the Y-axis. The false positive rate (FPR) is plotted on the X-axis (Figure 4.10, Figure 4.11, and Figure 4.12). The excellent performance of our classifier is reflected by a ROC curve which lies in the upper left triangle of the square (Figures 4.11a and Figure 4.11b).

Table 4.7: WEKA Decision Tree Algorithms Output (YES Class)

| Model | Correctly | Incorrectly | TP Rate | FP Rate | Precision | ROC Area |
|---------------|------------------|--------------------|----------------|----------------|------------------|-----------------|
| Reptree | 1578 | 88 | 0.714 | 0.005 | 0.967 | 0.832 |
| J48 | 1583 | 83 | 0.714 | 0.001 | 0.990 | 0.834 |
| Random Forest | 1583 | 83 | 0.714 | 0.001 | 0.990 | 0.927 |
| DesicionStump | 1577 | 89 | 0.714 | 0.006 | 0.952 | 0.829 |
| HoeffdingTree | 1581 | 85 | 0.714 | 0.003 | 0.981 | 0.832 |
| LMT | 1583 | 83 | 0.714 | 0.001 | 0.990 | 0.921 |
| RandomTree | 1581 | 85 | 0.714 | 0.003 | 0.981 | 0.929 |

Table 4.8: WEKA Decision Tree Algorithms Output (NO Class)

| Model | Correctly | Incorrectly | TP Rate | FP Rate | Precision | ROC Area |
|---------------|------------------|--------------------|----------------|----------------|------------------|-----------------|
| Reptree | 1578 | 88 | 0.995 | 0.286 | 0.944 | 0.832 |
| J48 | 1583 | 83 | 0.999 | 0.286 | 0.945 | 0.834 |
| Random Forest | 1583 | 83 | 0.999 | 0.286 | 0.945 | 0.927 |
| DesicionStump | 1577 | 89 | 0.994 | 0.286 | 0.944 | 0.829 |
| HoeffdingTree | 1581 | 85 | 0.997 | 0.286 | 0.945 | 0.832 |
| LMT | 1583 | 83 | 0.999 | 0.286 | 0.945 | 0.921 |
| RandomTree | 1581 | 85 | 0.997 | 0.286 | 0.945 | 0.929 |

Area under the ROC Curve (AUC) provides a value description for the performance of the ROC curve. Our AUC value is a portion of the area of the unit square, so its value will always be between 0 and 1 and usually more significant than 0.5 [192], according to [193] the area under a ROC curve (AUC) is a criterion used in many applications to measure the quality of a classification algorithm. A poor sorter has an AUC of around 0.5 (no discrimination; it is chosen randomly) and 1 (perfect discrimination).

In order to choose the right model, we examined the information provided in Tables 4.7 and 4.8. After analyzing the data, we found that the DecisionStump, HoeffdingTree, LMT, and RandomForest models did not produce a clear final tree structure in the Weka software. As a result, we decided to eliminate these models from consideration.

Our analysis led us to focus on three models: J48, RepTree, and RandomTree. The RandomTree model has an impressive AUC of 0.929, which is higher than both J48 and RepTree

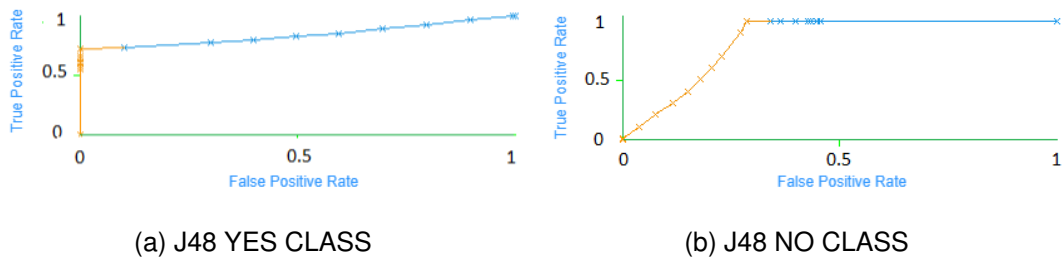


Figure 4.10: ROC Area for J48 model - YES and NO Class

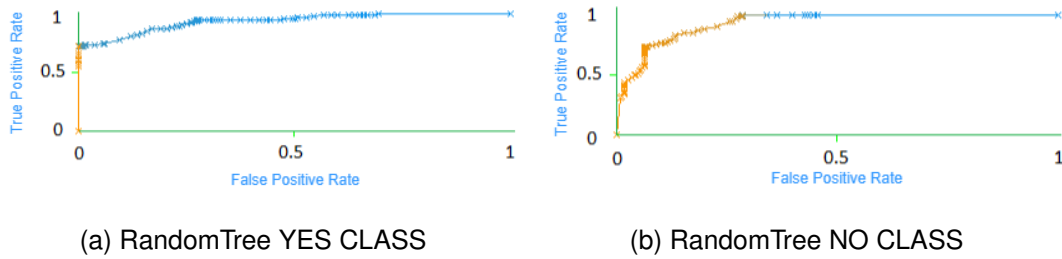


Figure 4.11: ROC Area for RandomTree model - YES and NO Class

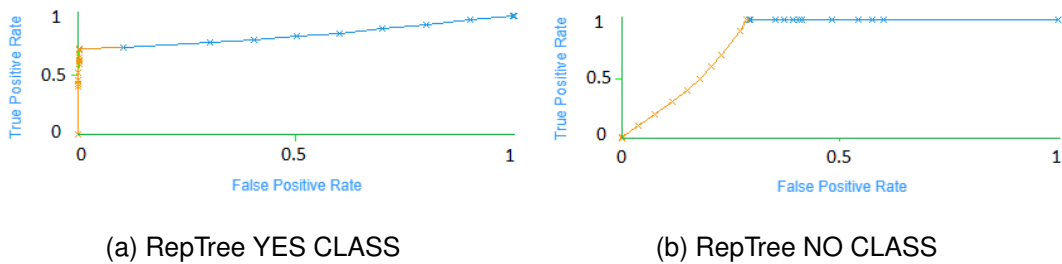


Figure 4.12: ROC Area for RepTree model - YES and NO Class

models. However, the J48 model has superior accuracy compared to the RandomTree and RepTree models, despite the RandomTree model's high AUC.

We also analyze the correctly classified instances; model J48 has a higher value (1583) than RepTree (1578) and RandomTree (1581). Likewise, the J48 model incorrectly classifies fewer instances (83) than the RepTree (88) and RandomTree (85) models. The selected model will then be J48 to obtain the programming rules of our script. Figure 4.13 shows the decision tree structure obtained with the J48 model.

F. Deployment

Our goal was to create a data set from the cookies generated by analyzing users' browsing history. This data set helped to train our model responsible for classifying new cookies from a new data set of navigation obtained from the web control appliance. The rules obtained

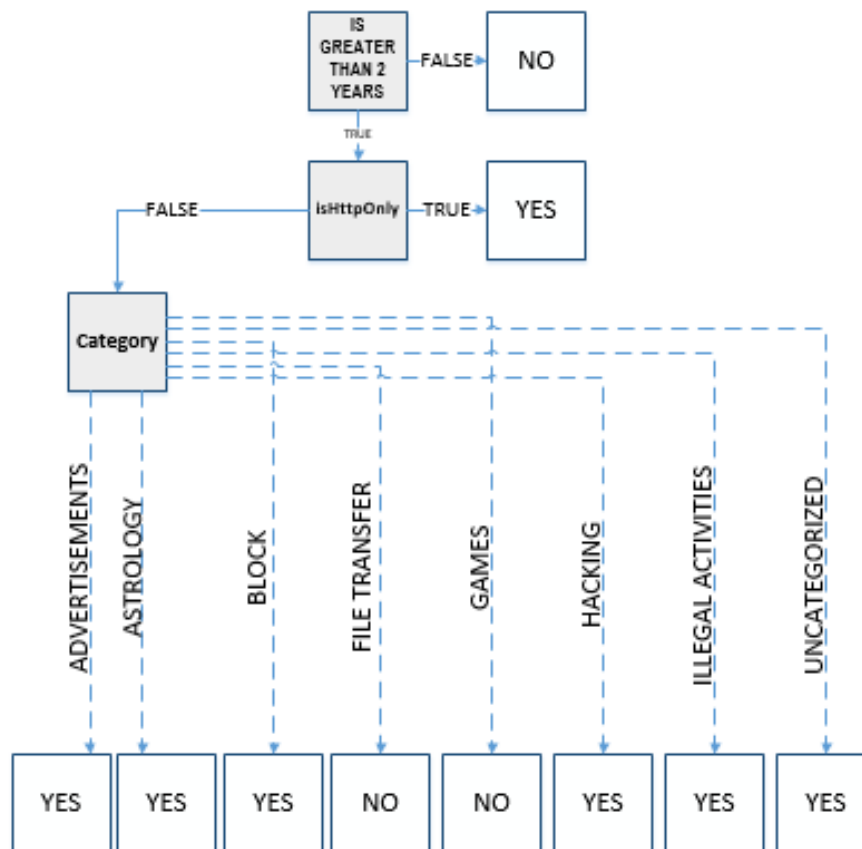


Figure 4.13: Structure of the decision tree obtained with the J48 algorithm

from the decision tree algorithms were applied in phase 3 of our model (Data Analysis) because the navigation records and the cookies information that had been generated were stored there **DataSet.SQLite**. Furthermore, in phase 3 our data set was structured with the attributes of the selected cookies and those that were added (Category).

The rules obtained from the decision tree, using the J48 algorithm, allowed us to develop a script using Python to obtain the domains to which the cookies classified as suspicious belong. In Figure 4.14, a preview of the code developed to structure these rules is shown, while Figure 4.15 shows a view of the output obtained when executing this script.

This script was executed with a new structured data set obtained following the sequence of phase 1 but this time obtaining the navigation record of a different day. Subsequently, the cookies were registered. The final data set was joined by 1185 new cookies registered and managed to rate 26 domains as suspicious through the analysis of their cookies.

In phase 1, we developed a model to classify cookies and prevent XSS. We analyzed the attributes of cookies generated during a controlled attack and showed that XSS attacks can be executed via compromised JavaScript files (*.js).

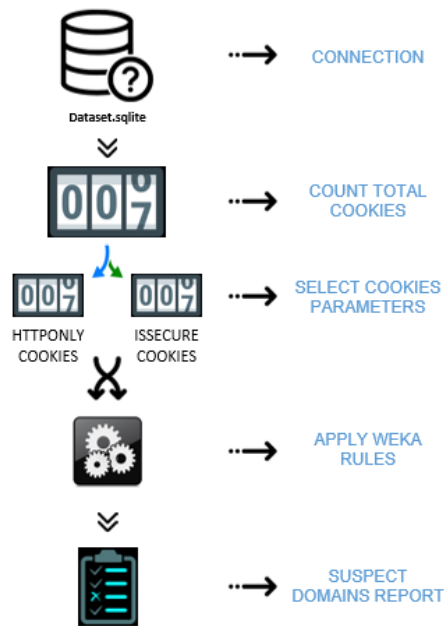


Figure 4.14: View of Python algorithm to obtain the domain of cookies classified as suspicious according to the rules obtained with Weka

```

Shell
>>> %Run sqlite-pythonControl.py

Cookie database opened successfully
Number of cookies found: 1185
Number of cookies isHttponly type found: 93
Number of cookies isSecure type found: 27

Applying rules obtained from WEKA...

Suspect Domain found: codeonclick.com
Suspect Domain found: codeonclick.com
Suspect Domain found: switchads.com
Suspect Domain found: yandex.ru
Suspect Domain found: yandex.ru
Suspect Domain found: powerlinks.com
Suspect Domain found: powerlinks.com
Suspect Domain found: pubmatic.com
Suspect Domain found: deployads.com
Suspect Domain found: afsanalytics.com
Suspect Domain found: admedo.com
Suspect Domain found: adriver.ru
Suspect Domain found: adroll.com

```

Figure 4.15: Suspect Domains Report, Output after executing the script developed in Python

In this second phase, the data collection showed that the visited websites generated suspicious cookies and JavaScript files. As shown in Figure 4.16, we found a script that can

be used to execute an XSS attack. In this case, the disadvantage is that it is sent in plain text for the use of HTTP protocol, which means that it can be easily captured with a network sniffer like Wireshark.

| URL | URL CATEGORY | Disposition | Threat Type |
|---|------------------------|-------------|-------------|
| https://static.ads-twitter.com:443/oct.js | Advertisements | Allow | adware |
| http://cdn.luckyorange.com/w.js | Computers and Internet | Allow | othermalwa |

Figure 4.16: Domain which contains JavaScript files extension susceptible to XSS attacks

Figure 4.17 shows one of the domains found in the **Advertisements** category that contains a phishing attack and whose cookie can be seen in Figure 4.25. It was observed that this page was redirected by an original domain called codeonclick.com.

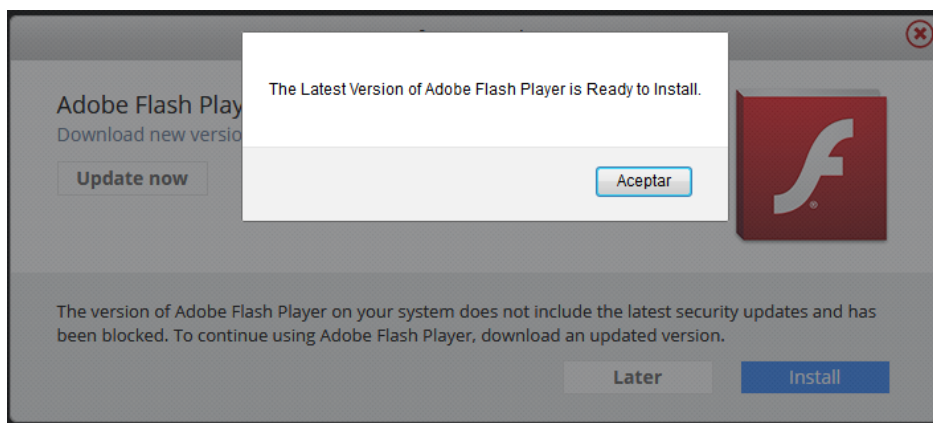


Figure 4.17: Advertisement category with fake Adobe Flash Player update message for phishing.

| Domain/Host | Path | Name | Value | Expiration Date | Secure |
|---|------|---------|----------------|----------------------|--------|
| installupgradenow.bigtrafficsystemsforupgrades.date | / | channel | ronn_pc_ach_a1 | 9/16/2017 7:05:14 PM | No |

Figure 4.18: Cookie generated for the domain codeonclick.com and redirected to the domain http://installupgradenow.bigtrafficsystemsforupgrades.date

4.1.3. Sub Phase 3: Cookie Theft

No matter how sophisticated our security is, the most significant risk will always be the same: users click on the wrong links and send their passwords to the wrong websites. So in Figure 4.19, we have proposed a model to teach computer security tips through a controlled attack of XSS. Our model is composed of the following phases:

1. **Recognition or Contact:** The objective of this section is to select the people that will be the target of our analysis
2. **Collection of information:** The objective of this phase was to stimulate the curiosity of this people.
3. **Compromised blog:** Using the Blogger application to create a blog vulnerable to XSS
4. **Steal of cookies:** Create and steal a test cookie that is created on the computers of the victims
5. **Teach about the executed attack:** Create a blog post to show what happened with the attack

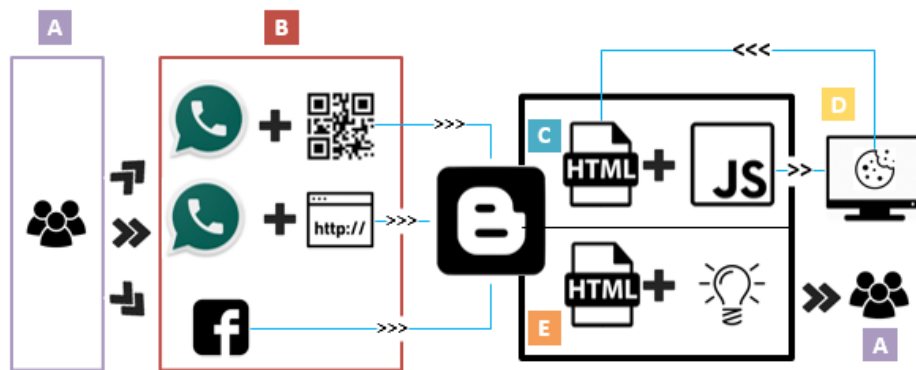


Figure 4.19: Proposed model to teach computer security through XSS attacks

Due to the users' sensitivity, an intrusive attack was not carried out to protect their privacy.

A. Recognition or Contact

In the first phase we will test the trust and curiosity of our contacts using a social engineering technique. We encourage them to review links with relevant information and discover something new. We aimed to teach them "how to steal personal data using a WhatsApp status". However, as mentioned in [194], curiosity does not always equal intelligence. Our study took advantage of this curiosity, and we motivated their curiosity by offering them false information and distracting them through our blog [195].

B. Collection of information

Three real scenarios were proposed to play with the trust of our contacts in the messaging application WhatsApp and the social network Facebook. The objective of this phase was to stimulate people's curiosity. To fulfill this first objective, three attack vectors were used:

WhatsApp status that contains a QR code:

As seen in Figure 4.20, this QR code was generated with the Social Engineering Tools (SET) software, which allowed us to camouflage a link to our compromised website. However, as the sensitivity of our contacts cannot be affected, our blog only contains information that specifies the methodology to obtain their preferences by stealing their cookies. This website was implemented as a personal blog.

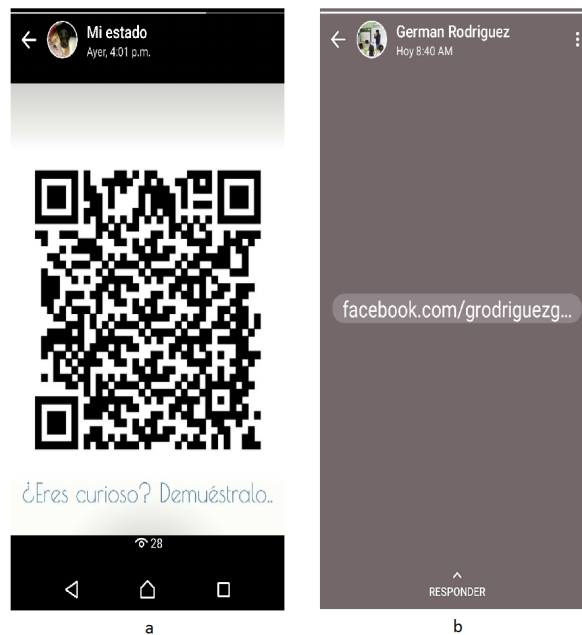


Figure 4.20: a) QR Code generated as a WhatsApp status, b) Status of WhatsApp with link to the compromised blog

A "Visitor Counter" was implemented in this blog to collect visitor characteristics as shown in Figure 4.21.

- Country of origin
- Type of browser used
- Type of operating system

- Date and time of visit





| Country | Device | Operating system | Browser | IP | Time |
|---|---|---|---|------------|-------------------|
|  |  |  |  | ██████████ | 6/8/2017 15:25:58 |

Figure 4.21: Data collected with the hidden visitor counter

The objective of this vector was to count how many contacts were persuaded to use some medium or application that could decode the QR code. With this scenario it was shown that the vector was not very intrusive since the QR Code acts as a static figure and users needed to download a QR Reader application to be able to decode it. We must remember that our contacts have basic computer knowledge. However, 3 contacts managed to decode it and access the implemented blog.

WhatsApp status with a link to the compromised blog:

The objective of this second scenario was to create a link to the compromised blog and put it as a WhatsApp status, as shown in Figure 4.26 b. However, the vector was not very intrusive either; only five contacts accessed the blog after reviewing our WhatsApp status.

A post on Facebook with a fake note:

In this third scenario, showed in Figure 4.22, a post was published on the Facebook social network that read: *¿Do you want to know how to steal information through a WhatsApp status? More information here ...* The same was linked to our compromised blog.

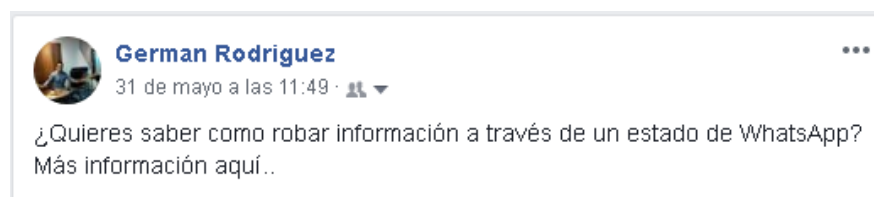


Figure 4.22: Post on Facebook with a false note to redirect to the compromised blog

The compromised blog:

This section explains how the compromised blog was implemented using the Blogger application, which allows the creation and publishing of online blogs. According to the Blogger website [196], to publish content, the user does not have to write any code or install server

or scripting programs. However, here, we show that it is not true and that a blog could be implemented with HTML and JavaScript code that steals a cookie created by the same blog.

We cannot control the sensitivity of our contacts. We gained access to this blog by taking advantage of their trust and curiosity. The blog was designed to demonstrate how their information could be obtained by stealing their cookies. We only extract information from the cookies created by the blog they visited, not from the sites they browsed.

C. Steal of cookies

In the Figure 4.23, we aimed to steal a cookie using JavaScript codes. We conducted a test to obtain the cookie created by the blog and documented the action. The obtained results were added to the blog as an entry.

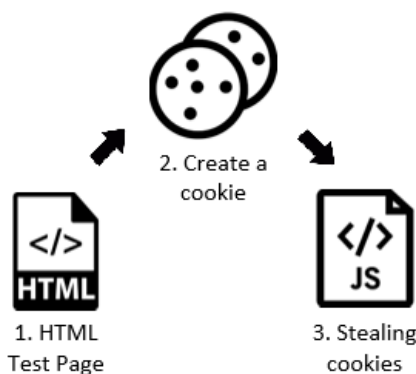


Figure 4.23: Proposed scheme to implement the compromised blog

In summary, this scheme proposes the implementation of our blog with XSS codes in 3 steps:

Create the Test HTML blog:

An entry titled Phishing Test was created using the Blogger application and JavaScript codes to generate and retrieve information from a cookie. This is shown in Figure 4.24.

Create a test cookie on the user's computer:

The JavaScript code was developed within the blog to create a cookie on the user's computer with the information shown in Table 4.9.

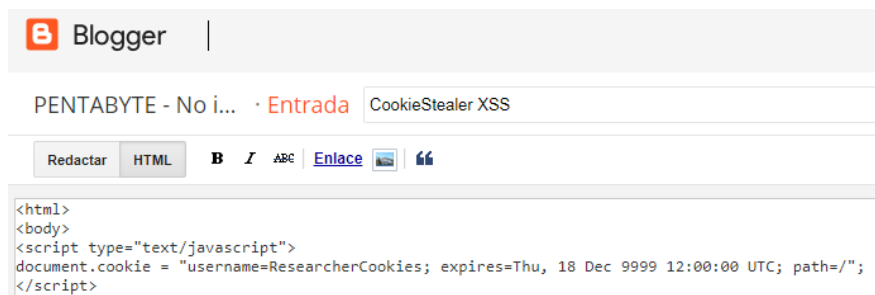


Figure 4.24: Compromised blog design using JavaScript code

Table 4.9: Content of the cookie that will be created on user's computer when visiting the compromised blog.

| Parameters | Value |
|------------|---------------------------------------|
| Name | username |
| Content | ResearcherCokies |
| Domain | pentabyteblog.blogspot.com |
| Root | / |
| Send to | Any type of connection |
| Created | Monday, June 4, 2018, 16:34:39 |
| Expires | Saturday, December 18, 9999, 07:00:00 |

Steal the content of the cookie created:

The JavaScript code was embedded in the blog's HTML code to retrieve the cookie's content. When a user visits the website, the same page will show the content of the cookie that was created on his computer. This can be improved by designing a script that redirects the information of all cookies obtained from each user who visits the blog. However, this would be a more intrusive attack and break the privacy of our contacts.



Figure 4.25: Preview of the message displayed by the blog with the stolen cookie information

In Figure 4.25, we can see the outcome of the process. The result appears as new content on the blog and is labeled as **STOLEN COOKIE = ResearcherCookies**. This label is used to display the cookie content that was created on the page. This demonstration is intended to show how to obtain the cookie's content.

D. Teach about the executed attack

Based on the analysis of scientific literature, it has been found that some proposals are not designed to educate users who have limited knowledge. On the other hand, there are proposals that aim to educate users, but they are limited in their scope as they focus on advanced users [197].

Our goal was to educate users who needed to learn about computer security. We used tips and advice from the same blog that we used to steal the test cookie from the browsers that visited the blog to teach them how to prevent all the actions that users executed in this controlled attack.

With the first and second phases of our model, it was proven that our trusted contacts used their curiosity to access the compromised blog. These are some of the tips that were published in the blog post.

¿Do you know how we get you to see our engaged blog? We gently abuse your curiosity and confidence. Hackers exploit this because they know the first attack movement is always trust.

Did you know that the information we published on the blog was false? Most users are interested in hacking their partners' Facebook or WhatsApp accounts.

¿Did you know that information cannot be stolen through WhatsApp statuses? Users without knowledge of computer security access any information that catches their attention.

Analysis of Results

This section analyzes the results obtained during the data collection and execution of the three attack vectors.

After analyzing Table 4.10, we discovered that only three contacts accessed the compromised blog through a QR code decoding application. This was confirmed by the visit counter,

Table 4.10: Summary of contacts who visited the blog

| - | Through QR Code in WhatsApp | Through Link from WhatsApp | Visiting Facebook Post | TOTAL |
|---------------------------------|-----------------------------|----------------------------|------------------------|-------|
| Contacts that accessed the blog | 3 | 5 | 174 | 182 |

which registered the visits on the day the status was published in WhatsApp. On the other hand, only five contacts accessed the compromised blog by clicking the URL link set as the WhatsApp status. Most of the visits were obtained through Facebook publications with false information. The post received 174 clicks and redirected the users to the compromised blog.

As shown in Figure 4.26, our visitor counter recorded that the operating system most used to access the blog was Android (32% - 58 users), followed by Windows (31% - 56 users) and iPhone (30% - 54 users). On the other hand, the most used browsers were Google Chrome (57% - 103 users), followed by Mobile (30% - 55 users) and Firefox (10.43% - 19 user), as shown in Figure 4.27. It was also recorded that the most significant number of visitors was in Ecuador (140), followed by the United States (29), Peru (6), Spain (3), Italy (2), the Philippines (1) and Chile (1). Our analysis confirms that all the visitors who accessed the blog were not just casual browsers, but individuals genuinely interested in our content. This was evident from their active participation through comments on the blog, seeking more information about the tips we shared.

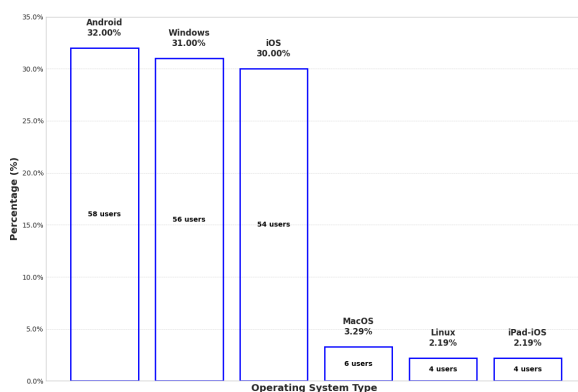


Figure 4.26: Statistics by Operating System obtained from the visitor counter

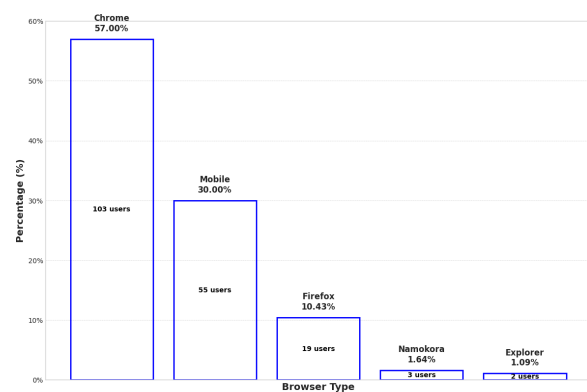


Figure 4.27: Statistics by Browsers obtained from the visitor counter

4.1.4. Sub Phase 4A. Personal Data Filtering

In this phase, we explain the model of our proposal called XSSStudent, which is divided into two parts. The first is shown in Figure 4.28, whose objective was to determine if the Universidad de las Fuerzas Armadas-ESPE users are susceptible to XSS attacks based on QR codes. We also investigated if their mobile devices were installed with some antivirus software or similar that allowed them to detect and block this type of attack. The second section is shown in Figure 4.28, where the proposal to mitigate this type of attack is detailed.

Controlled model to execute the XSS attack in the university

The model of our attack is composed of the following elements:

1. Victims
2. Attack vectors
3. Survey
4. Vulnerable pages

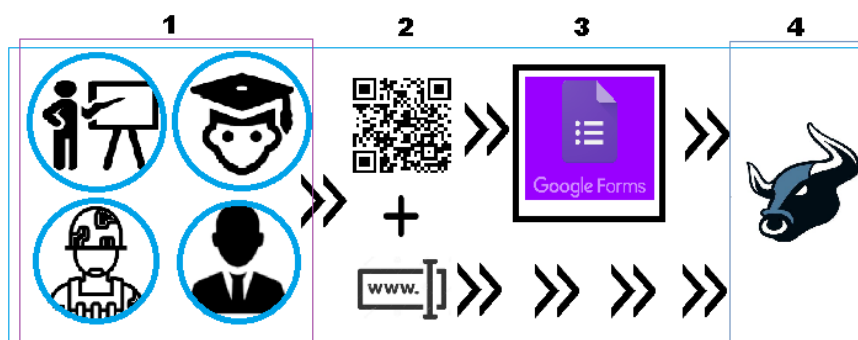


Figure 4.28: XXStudent: Model used to execute the XSS attack

1. Victims

The controlled attack was planned to analyze the security level of users' mobile devices against XSS-type attacks. Four types of users were targeted: teachers, students, administrative personnel, and military personnel of the Universidad de las Fuerzas Armadas sede Santo Domingo de los Tsáchilas. Through flyers placed in all hallways, interested users were invited to check the security level of their mobile devices. We use this methodology based on the study presented in [198].

2. Attack vectors

The access request to the vulnerable web pages was made through a flyer (Figure 4.29) and two attack vectors. On the flyer, we place the following instructions:

- "If you have activated your cell phone's mobile data, please change it to the Wi-Fi university network (this way, we can ensure that the study will be reliable and safe)".
- "If you do not have a QR code reader, you can download any QR Reader type application from your preferred application stores".
- "Once you have the QR Reader application, please scan the following code (Vector 1), click OK, and complete the proposed survey in Google Forms".
- "If you can not download the QR Reader application you can enter through any browser to the following link (with redirection to a server within ESPE domain): <http://tinyurl.com/y6ot6z8u>".

This methodology was established to avoid setting up vulnerable websites using public servers. A local server was configured within the university's local domain to do this.

2. Survey

With the first attack vector, the victim scans the QR code that directs it to a survey prepared in Google Forms. In this survey, we request the following information, as seen in Table 4.11:

The last question in the survey contains the URL link to the website compromised with the XSS script. This link was also camouflaged with the Tinyurl [199] application so that users would be more confident about clicking on it.

3. Vulnerable website

We have used the BeEF software in the same way as in the first phase of our framework. As mentioned, it is a penetration testing tool that analyzes web browsers. A server was configured via the Docker application with the essential requirements, and we published it locally within the university's domain network. This prevented any information leak by users. The software used has two websites with JavaScript XSS called Hook.js.

As seen in Figure 4.30, a bull logo appears with the letters BeEF. In the last survey question, reference is made to this image; if the users could see it, the attack succeeded. The

XSSstudent: estudio del nivel de seguridad de los dispositivos móviles en la ESPE-SD

Este estudio permitirá evaluar el nivel de seguridad de los dispositivos móviles de todo el personal (Estudiantes, Docentes, Militares, Administrativos) ante un ataque conocido como Cross-Site Scripting (CSS).

Instrucciones:

1. Si tienes activado los datos móviles de tu celular por favor cámbiate a la red wifi universitaria (asi aseguramos que el estudio será confiable y seguro).
2. Si no dispones de un lector de códigos QR puedes descargar cualquier aplicación de tipo QR Reader de las tiendas de aplicaciones de tu preferencia.
3. Una vez que tengas la aplicación QR Reader, por favor escanea el siguiente código, da clic en Aceptar y completa la encuesta propuesta en Google Forms.



4. Si no puedes descargar la aplicación QR Reader puedes ingresar por cualquier navegador al siguiente enlace: <http://tiny.url.com/y6ot6z2u>

Figure 4.29: Preview of the flyer that were placed in the hallways of the University



You should be hooked into **BeEF**.
 Have fun while your browser is working against you.
 These links are for demonstrating the "Get Page HREFs" command module:

- [The Browser Exploitation Framework Project homepage](#)
- [BeEF Wiki](#)
- [Browser Hacker's Handbook](#)
- [Slashdot](#)

Have a go at the event logger. Insert your secret here:

You can also load up a more [advanced demo page](#).

Figure 4.30: Website No.1 compromised with the Hook.js script

Table 4.11: Questions posed in the survey to analyze the level of security of the devices.

| Question | Possible Values |
|--|---|
| What kind of respondent are you? | student, teacher, administrative, military |
| If you are a student, what career does it belong to? | Tecnologías de la Información, Agropecuaria, Biotecnología |
| What is your gender? | Male, Female, I do not want to say it |
| How old are you? | [Open answer] |
| What operating system does your mobile device have? | Android, IOS, Windows Phone, Other |
| What is the brand of your mobile device? | Samsung, Motorola, Nokia, Alcatel, LG, Huawei, Sony, ..., Other |
| Do you have any antivirus, firewall or similar software installed on your mobile device? | Yes, No |
| If you have any antivirus, firewall or similar software installed on your mobile device, write its name? | [Open answer] |
| Do you know what phishing means? | Yes, No |
| Do you know what XSS means? | Yes, No |
| Do you know what a QR code is? | Yes, No |
| How did you enter this survey? | Scanning the QR code of the flyer, Manually entering the URL link of the flyer, Through a link sent by whatsapp |
| Enter the following link: http://tinyurl.com/y6ot6z8u and tell us if you could see the following image (bull) on the website | Yes, No |

link on this page was **[http://\(iplocalserver\):3000/demos/essential](http://(iplocalserver):3000/demos/essential)**. The attack is simple but very powerful because just by accessing this web page, the user could join a network of devices in zombie mode, to which remote commands could be sent to perform tasks such as obtaining information from the browser, for example, type, version, name, date and time of access, type of mobile device. Figure 4.31 is also compromised with the Hook.js script.

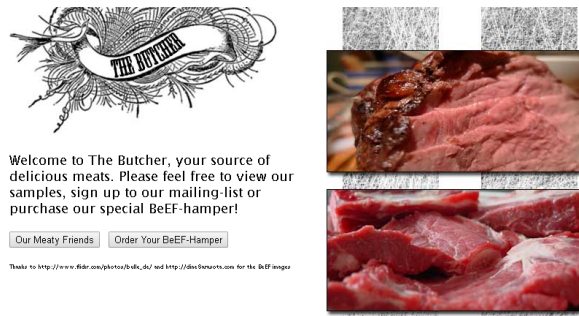


Figure 4.31: Website No.2 compromised with the Hook.js script

However, asking users if they were able to see images of meat when they accessed this vulnerable page was not very appealing, so we decided to only forward to the first page.

4. Proposed model to prevent XSS attacks

In Figure 4.32, we can see our proposal to mitigate this attack that has been executed using the Beef software. This proposal is composed of 4 sections:

XSSStudent QR Reader

URL extractor

Cloud espe.local

XSS Detect

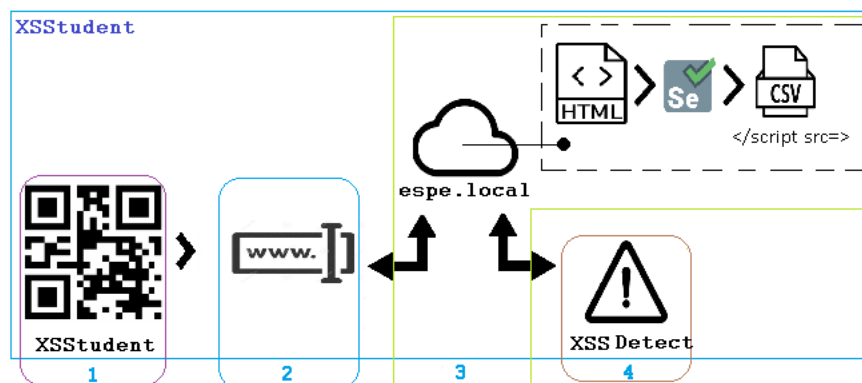


Figure 4.32: XSSStudent: proposed model to prevent XSS attacks

4.1. XSSStudent QR Reader

An application was developed to read QR codes based on APP Inventor software. The BarCodeScanner and WebViewer components were used together. The first is to offer a

reliable reader, and the second is to be an embedded browser that allows control of the web pages that are opened when a QR code is scanned. To ensure the operation of these components, we include the following values in their configuration properties, as seen in Table 4.12:

Table 4.12: Configuring the BarCodeScanner and WebViewer Components properties for QR Reader app

| Component | Properties | Values |
|----------------|---------------------|---|
| BarCodeScanner | AfterScan | The GoToURL block it is enabled and disabled according to our control |
| WebViewer | FollowLinks | disabled |
| | PromptforPermission | enabled |

The GoToUrl block calls the event to open the URL link scanned with the BarCodeScanner component. We could control it through control statements to enable or disable it according to the result of the XSS Detect block. Thus, we ensure that our QR Reader does not open the links after scanning the QR codes. The FollowLinks property also determines whether URL links should be followed. In the disabled state, the malicious links will not be opened until they are analyzed.

4.2. XSStractor

We extracted the URL from a text string and used this script to control its execution. This helped us prevent web pages with XSS commands from running automatically, protecting against potential attacks. We applied this technique to the web pages shown in Figures 4.30 and 4.31.

4.3 Cloud espe.local

Previously, the pages of Figure 4.30 and Figure 4.31 were analyzed. We train a system to identify all the JavaScript type code it references and the code it calls via the string "src = *.js", which are not on the same page. We use Python software Selenium to perform web scraping on these sites. Our web scraper searched for all the code sections that contain the script presented in Figure 4.33:

We have created a database in CSV format that lists URLs of pages that are vulnerable to XSS attacks or contain the *.js extension, which could be called by a third party from

```
<script>
  var commandModuleStr = '<script src="/hook.js" type="text/javascript"></script>';
  document.write(commandModuleStr);
</script>
```

Figure 4.33: JavaScript code that our web scraper searches within the compromised websites

another website. Our web scraper is designed to detect the code that includes the string **src="hook.js"**. We have also replicated this process for a web page that uses XSS-type scripts. The resulting database has been set up on a local server at the local domain. It contains information on URLs that have been previously analyzed and found to be suspicious or contain XSS scripts.

4.4. XSS Detect

If the URL embedded in the QR code scanned with our scanner was within our knowledge base stored in the local cloud, our QR reader displays the alert message "ALERT: suspicious page, contains XSS type vulnerabilities". However, if the page was not marked as suspicious within our secure cloud, the control flag allowed the activation of our Web-Viewer component. This component displayed the web page associated with the URL that was scanned using our QR reader.

All this process has been summarized in the following conceptual Algorithm 2:

Analysis of Results

In order to verify that our link to the compromised website was not detected as malicious by any software, the online application Virus Total [200] was used. The analysis results are shown in the Figures 4.34 and 4.35.

A pentest of the compromised websites with XSS codes was made using the online tool shown in [201]. However, there was no response to detect an XSS attack. A summary of the results of all the online pentest tools that were used for our research, according to [199], is shown in the Table 4.13

Online tools were also used to detect XSS attacks, as shown in [207]; the results are shown in Table 4.14.

The XSSCon [217] software, a Powerful Simple XSS Scanner made with Python 3.7,

Algorithm 2 Proposed algorithm to detect URLs vulnerable to XSS attacks

```

Data: BarCodeScanner.AfterScan ← QResult
Data: ScannedURL ← QResult
Data: XSSResult ← 0
Data: XSSAlert ← "result"
Data: WebViewActive ← 0
if QResult == "/demos/basic.htm" then
    XSSResult ← XSSResult + 1
end if
if QResult == "/demos/butcher/index.htm" then
    XSSResult ← XSSResult + 1
end if
if QResult == "http://tinyurl." then
    XSSResult ← XSSResult + 1
end if
if QResult ∈ "espe.local.csv" then XSSResult ← XSSResult + 1
end if
if XSSResult == 4 then
    XSSAlert ← "URL found suspicious"
    WebViewActive ← 0
else
    XSSAlert ← "URL found not suspicious"
    WebViewActive ← 1
end if

```

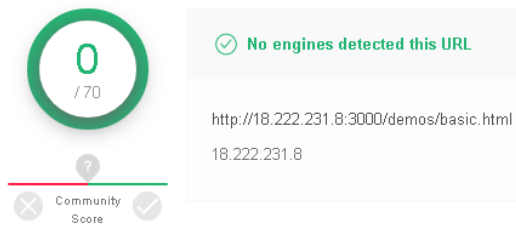


Figure 4.34: Results of the vulnerability check of our website /demos/basic.html

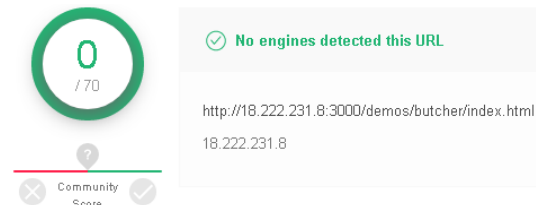


Figure 4.35: Results of the vulnerability check of our website /demos/butcher/index.html

was also tested. However, it did not detect any attack when testing the two compromised websites. Something interesting that was found within the results was information on hidden data collection referring to names, telephones, addresses, and identification of credit cards, as shown in Figure 4.36.

Some browsers compatible with the Android mobile operating system verified the lack of access control to pages vulnerable to XSS. As seen in the Figure 4.37, 93.7% of the university users had this operating system installed on their mobile devices. With these data, the following results were obtained.

Table 4.13: Summary of results of the online pentest tools used

| Tool | Reference | Results | Disadvantages |
|---------------|-----------|--------------|---|
| Netcraft | [202] | not detected | 28 days of evaluation, then it is mandatory to pay \$1.06 monthly, in addition to registering a credit card. If the subscription request is not accepted, the application is not initialized. |
| Pentest-Tools | [203] | not detected | Was not able to detect the XSS attack |
| Ping.eu | [204] | not detected | Does not offer the online URL analysis service |
| Yougetsignal | [205] | not detected | Does not offer the online URL analysis service |
| Urlquery.net | [206] | not detected | Is not functional, Invalid URL or failed to DNS lookup. |

```
[INFO] Collecting form input key....
[INFO] Form key name: yourname value: <script>console.log(5000/3000)<
[INFO] Form key name: phone value: <script>console.log(5000/3000)</sc
[INFO] Form key name: address value: <script>console.log(5000/3000)</
[INFO] Form key name: creditcard value: <script>console.log(5000/3000
```

Figure 4.36: Results obtained using the XSSCon software.

- **Chrome for android:** Does not detect the XSS attack
- **Firefox for android:** Does not detect the XSS attack
- **Android default browser:** Does not detect the XSS attack
- **Opera mini for android:** Does not detect the XSS attack

Finally, Figures 4.37 to 4.44 detail the results obtained with the XSS attack carried out on all users (65% men and 35% women) of the University.

4.1.5. Sub Phase 4B: XSS2DENT

As seen in Figure 4.45, our proposal has been structured in 5 components:

Table 4.14: Summary of results of the online XSS scan testing tools used

| Tool | Reference | Results | Disadvantages |
|-------------|------------------|--|---|
| Find-XSS | [208] | Does not allow online analysis of URLs | it requests the execution of a JavaScript that was detected and blocked by chrome. |
| Quttera | [209] | It is not functional for common users | The links are entered for the URL analysis but the system always throws the message: Provided resource is unreachable |
| XSS-Scanner | [210] | It is not functional for common users | The website is not secure, use only the http protocol. |
| Acunetix | [211] | Without results | It requires a registration in the system to do an online scan. Use the https protocol |
| W3af | [212] | Without results | Is a plugin for identifying persistent cross-site scripting vulnerabilities |
| Ssllabs | [213] | Without results | Does not allow the analysis of URLs that use the IP format, use the https protocol |
| Tenable | [214] | It is not functional for common users | It does not allow the online analysis of URLs, it uses the https protocol. Software must be downloaded. |
| Swascan | [215] | It is not functional for common users | It does not allow the online analysis of URLs, it uses the https protocol. Software must be downloaded. |
| Rapid7 | [216] | It is not functional for common users | It does not allow the online analysis of URLs, it uses the https protocol. Software must be downloaded. |

1. Victims

This section describes the characteristics of the victims, who were the educational establishments of the city of Santo Domingo de los Tsáchilas. No model was established for this selection to calculate a population/sample; instead, students from the Universidad de las Fuerzas Armadas-ESPE first level of the Engineering in Information Technology Career were asked to share the proposed challenge with their ex-partners. Thus, this compromised

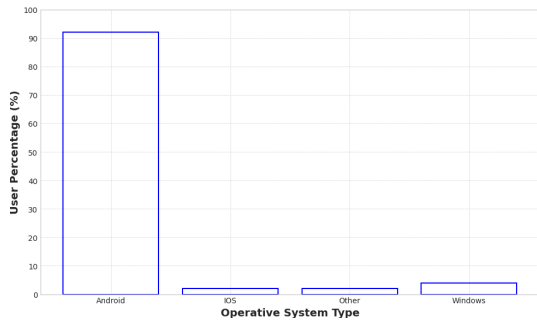


Figure 4.37: Mobile operating systems recorded during data collection

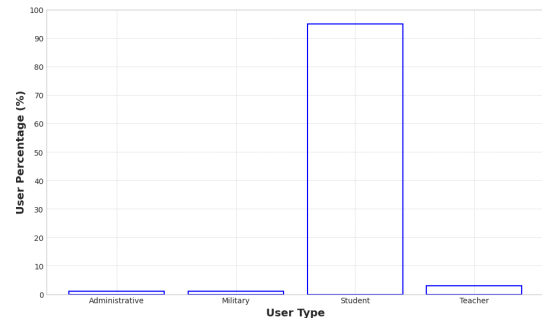


Figure 4.38: Type of users that accessed to the vulnerable website

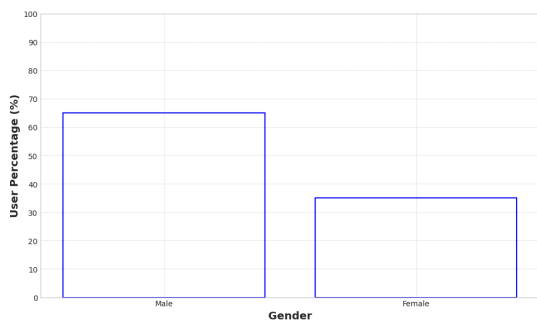


Figure 4.39: Gender of users who accessed the vulnerable website

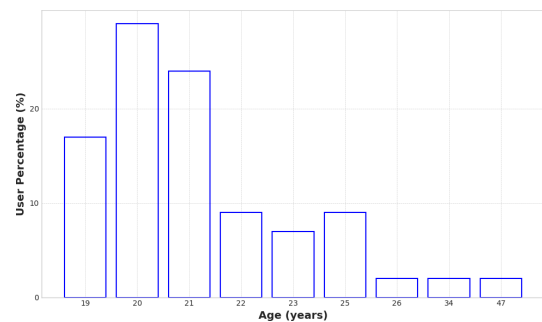


Figure 4.40: Average age of users who accessed the vulnerable website

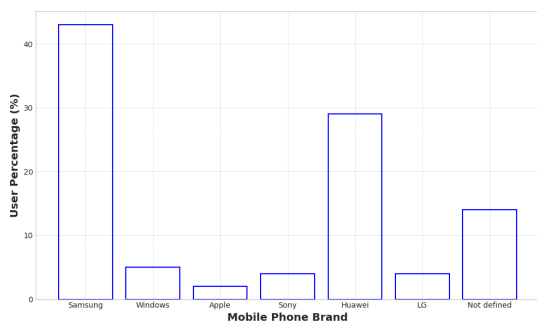


Figure 4.41: Information collected related to the brand of mobile devices

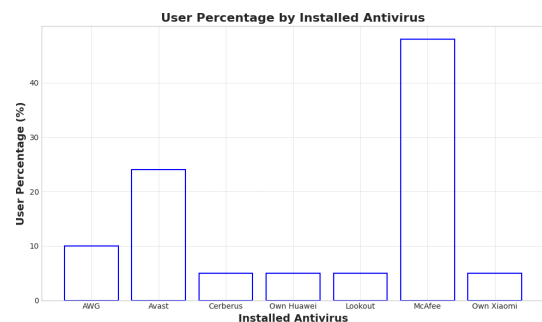


Figure 4.42: Information collected related to the antivirus installed on mobile devices

challenge's propagation level has been analyzed with a combination of 2 attacks. In this space, the following research questions were proposed (RQ1 and RQ2):

RQ1: The educational establishments in Santo Domingo de los Tsáchilas are vulnerable to attacks such as Cross-Site Scripting and clickjacking.

RQ2: By implementing Computer Security challenges, we can empower students to recognize and mitigate the dangers of Cross-Site Scripting and clickjacking attacks.

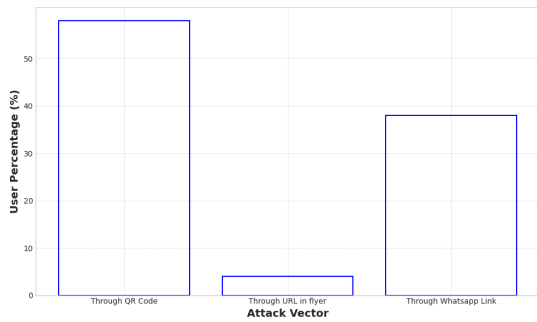


Figure 4.43: Record of the most used attack vector

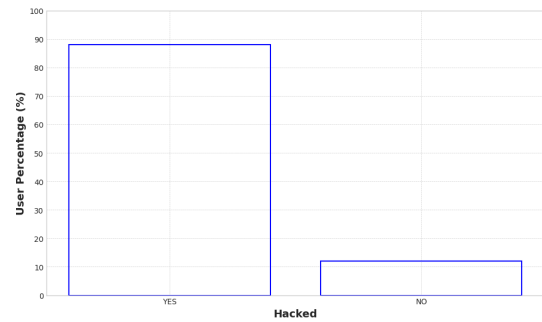


Figure 4.44: Total users hacked with our controlled XSS attack

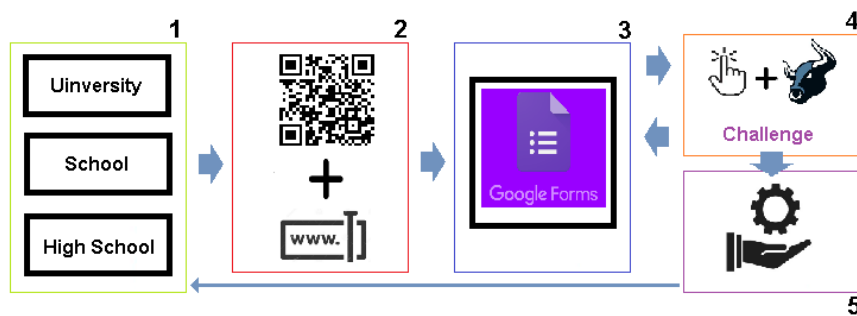


Figure 4.45: XSS2DENT: Proposed model to analyze XSS vulnerabilities in the educational establishments of Santo Domingo de los Tsachilas City

Based on the results obtained, these questions will be answered in the discussion section.

2. Attack Vectors

Flyers with a QR code were designed and delivered to the students for distribution. The code embedded a link to a survey to collect data from people who scanned it. Similarly, students could share the link to this survey through any social network.

3. Survey

This survey used a form designed using Google Forms to collect information about the victims. The attack was within a shortened link using the TinyURL service that was part of the proposed questions. The survey was structured with the variables of Table 4.15 for further analysis:

Table 4.15: Questions included in the proposed survey

| Question | Possible Values |
|---|---|
| Select an option | Student, Teacher, Public Employee, Private Employee |
| Select your current level of education | School, High School, University, Postgraduate |
| What is the name of your School / College or University where you are currently studying (student), or teaching (teacher)? | ESPE SD, PUCE SD, UTE SD, ITSAE SD, ITS Tsachila, UTPL SD, UNIANDES SD, Otros |
| Select your gender | Male, Female |
| How old are you? | Enter your age in numbers only |
| What operating system does your mobile device have? | Android, IOS, Windows Mobile, other |
| What is the brand of your mobile device? | Samsung, Motorola, Nokia, Alcatel, LG, Huawei, Sony, BlackBerry, Sony Ericsson, Apple, Blue, Lenovo, HTC, ZTE, Xiaomi, Windows Phone, Other |
| Do you have any antivirus, firewall or similar software installed on your mobile device? | YES, NO |
| If you have any antivirus, firewall or similar software installed on your mobile device, enter its name | Open answer |
| How did you enter this survey? | Scanning the flyer QR code, Manually entering the URL link of the flyer, Through a link received from a social network |
| Enter the following link: https://tinyurl.com/wmlm4po and solve the Challenge proposed by the Ethical Hacking Club of the ESPE. Find all the security flaws on that website write their names on the following line. If you meet the challenge, we will send you an email to contact you. | Open answer |
| How did you access this challenge? | Wi-Fi access of my School / College / University, Wi-Fi access of my House, Mobile data, I could not access the Challenge, Other |
| If you used mobile data, tell us which operator you have the service | Claro, Movistar, CNT, Tuenti, Other |

4. Challenge

Access to our cybersecurity challenge was included in the survey. This link was masked with the TinyURL online application to hide our server's public IP address and access port. Table 4.16 shows the clues raised for the challenge. Our goal was to distract the victims from clicking on a button that showed the clues of the challenge one by one. In reality, the

page had a clickjacking vulnerability combined with cross-site scripting; if the victims passed the clues with the button, they already joined our zombie network because the XSS script was running automatically and silently.

Table 4.16: Challenge proposed or the students

| Question | Proposed Challenge |
|----------|--|
| Clue 1 | Look for all the security flaws that this web page has |
| Clue 2 | If you are reading this message it is because your antivirus did not detect any of the failures. |
| Clue 3 | A stronger hint: Your mobile device is vulnerable to this type of security flaw |
| Clue 4 | A much stronger clue: Your device being infected becomes Zombie of a Botnet |
| Clue 5 | We have combined 2 types of attack to make detection more difficult |
| Clue 6 | Track Attack 1: What would you do if Mickey Mouse is kidnapped? |
| Clue 7 | Track Attack 2: The second thing you read will be the first thing you write: Extrañas Estudiar Español |
| Clue 8 | If you discover any of the security flaws, write your answer in the Google Form survey you are filling out |

Table 4.16 shows that clues 6 and 7 were raised to be answered in the survey. The victims believed that if the challenge were resolved, they would be integrated into the ESPE Cyber Security Club. The answers for these clues are described in Table 4.17:

Table 4.17: Answers to the questions proposed in the Cybersecurity Challenge

| Question | Description | Answer |
|--|---|--------------|
| Track Attack 1: What would you do if Mickey Mouse is kidnapped? | Mickey Mouse (mouse or click) + kidnapping | Clickjacking |
| Track Attack 2: The second thing you read will be the first thing you write: Extrañas Estudiar Español | The second thing that reads from Extrañas is X, Estudiar = S y Español= S | XSS |

5. Feedback

After data collection, a report describing all the actions taken with our proposed model was presented. This report was sent to the technical staff of the establishments affected by the XSS + Clickjacking scripts. Part of the report contained the recommended actions to block this type of URL. Furthermore, the results of the tools used that did not detect this attack were included in the report.

In order to verify that our link to the compromised website was not detected as malicious by any software, the online applications Virus Total [200] and KOXXS [218] were used. In addition, a pentest of the compromised website with XSS+Clickjacking codes was made using the online tool shown in [201]. However, there was no response to detect an XSS attack.

Analysis of Results

To ensure the fidelity of the data, no samples were calculated for the choice of educational establishments. Failing that, students from the first levels were asked to distribute the challenge since they had more contact with their ex-partners.

According to Figure 4.46, 93.5% of the victims had a mobile device with an Android operating system, and only 6.5% had an IOS. The four most used device brands are seen in Figure 4.47; Samsung covered 47.8% of mobile devices, Huawei with 39.4%, Sony with 8.7%, and Apple with 6.5%.

Figure 4.48 shows that 76.1% of victims had no antivirus software installed on their mobile devices. Furthermore, Figure 4.49 shows that 69.6% of the victims accessed the challenge through a home network connection (home access), 13% accessed it through their university network, another 13% accessed it using mobile data, and 2.2% accessed it through business networks.

Finally, Figure 4.50 shows that 4.3% of the victims accessed the challenge by scanning the QR code of the flyers distributed by the students, 37% accessed through the URL link found in the same flyer and 58.7% accessed through the link shared by the WhatsApp messaging network. On the other hand, Figure 4.51 shows the three universities that were compromised with our combined attack. The Pontificia Universidad Católica del Ecuador (PUCE) with 2.17% of the victims, the Universidad Autónoma de los Andes with 2.17%, and the Universidad Tecnológica Equinoccial (UTE) with 10.86%.

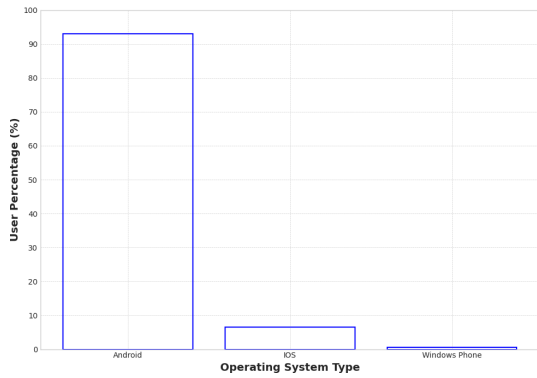


Figure 4.46: Mobile operating system recorded during victim data collection

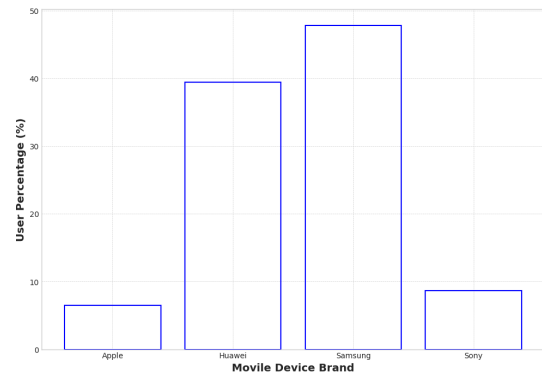


Figure 4.47: Manufacturer registered during victim data collection

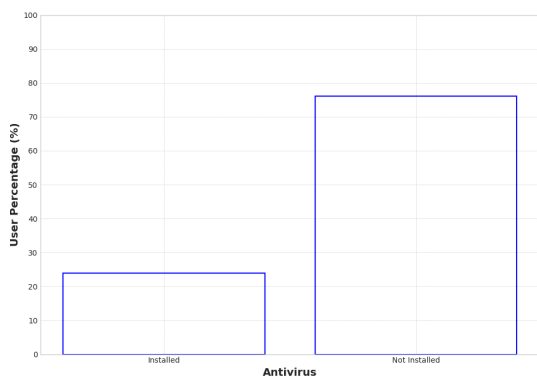


Figure 4.48: Percentage of devices with antivirus software or similar

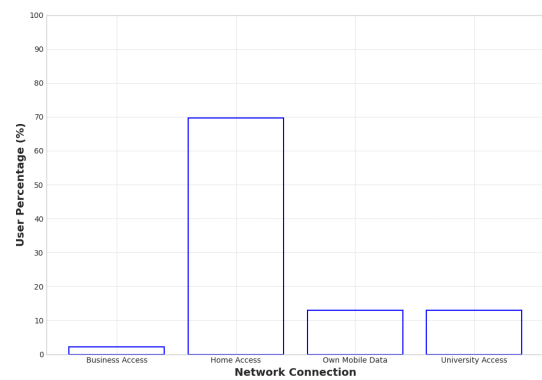


Figure 4.49: Network connection used to access the challenge

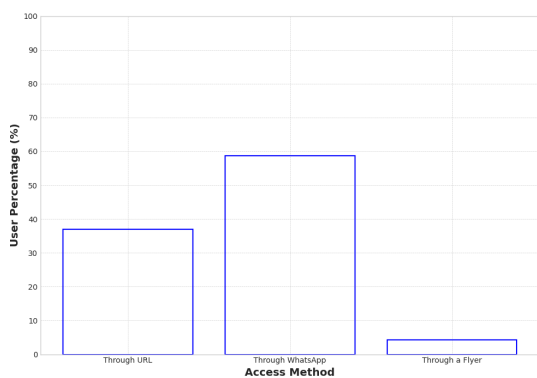


Figure 4.50: Attack vector with the highest records during access to the challenge

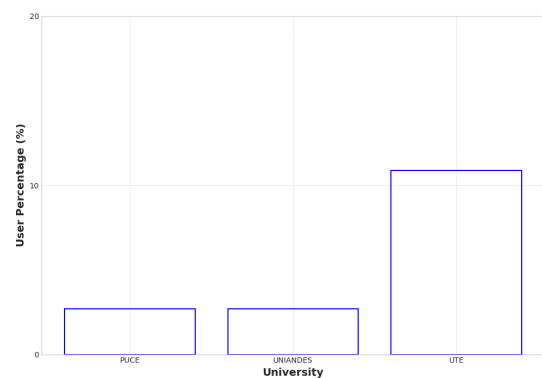


Figure 4.51: Universities and number of users who accessed the challenge from their local network

The results show that three universities in Santo Domingo de los Tsachilas are vulnerable to XSS attacks combined with Clickjacking; this represents a significant security gap in their network and infrastructure. 69.6% of the victims accessed our challenge through their home

network (home access), which opens a new case study to evaluate the level of security of residential internet providers in the city of Santo Domingo de los Tsáchilas against this type of attack.

With these results, a project could be implemented on a mandatory basis, including legal regulations, to analyze the level of vulnerability in all educational establishments in the city and encourage the creation and configuration of new security policies against XSS attacks.

So, we can answer (RA1 and RA2) our research questions:

Research Answer 1 (RA1): It was demonstrated that 3 Universities in Santo Domingo de los Tsáchilas are vulnerable to combined XSS + Clickjacking attacks.

Research Answer 2 (RA2): Feedback as part of our model helped raise awareness among students and authorities at the 3 vulnerable universities.

A limitation of this phase was the access policy implemented in some establishments to access the link of our cybersecurity challenge. Representatives of the ICT departments indicated that we needed special procedures to share the survey with students, so it was impossible to verify whether these establishments were vulnerable to XSS and Clickjacking attacks.

After analyzing our system logs, the following results are presented, which indicate that our cybersecurity challenge extended beyond what was planned. As seen in Figure 4.52, the logs recorded the variables of some connections from other countries (IP, city, country, longitude, latitude, time, and date of connection); the log in the figure shows the information of a zombie connected from Peru.

| | | | |
|------|--------|---|---------------------------|
| 8688 | Zombie | 190.236.8.26 appears to have come back online | 2019-12-05T13:03:29+00:00 |
| 8687 | Zombie | 190.236.8.26 is connecting from: {"city"=>{"geoname_id"=>3936456, "names"=>{"de"=>"Lima", "en"=>"Lima", "es"=>"Lima", "fr"=>"Lima", "ja"=>"リマ", "pt-BR"=>"Lima", "ru"=>"Лима"}}, "continent"=>{"code"=>"SA", "geoname_id"=>6255150, "names"=>{"de"=>"Südamerika", "en"=>"South America", "es"=>"Sudamérica", "fr"=>"Amérique du Sud", "ja"=>"南アメリカ", "pt-BR"=>"América do Sul", "ru"=>"Южная Америка", "zh-CN"=>"南美洲"}}, "country"=>{"geoname_id"=>3932488, "iso_code"=>"PE", "names"=>{"de"=>"Peru", "en"=>"Peru", "es"=>"Perú", "fr"=>"Pérou", "ja"=>"ペルー共和国", "pt-BR"=>"Peru", "ru"=>"Перу", "zh-CN"=>"秘鲁"}, "location"=>{"accuracy_radius"=>10, "latitude"=>-12.0464, "longitude"=>-77.0428, "time_zone"=>"America/Lima"}, "registered_country"=>{"geoname_id"=>3932488, "iso_code"=>"PE", "names"=>{"de"=>"Peru", "en"=>"Peru", "es"=>"Perú", "fr"=>"Pérou", "ja"=>"ペルー共和国", "pt-BR"=>"Peru", "ru"=>"Перу", "zh-CN"=>"秘鲁"}, "subdivisions"=>[{"geoname_id"=>3936451, "iso_code"=>"LMA", "names"=>{"en"=>"Lima"}]}} | 2019-12-05T13:03:29+00:00 |

Figure 4.52: System Log that shows a zombie connected from Peru

Figure 4.53 shows a global map that represents the number of connections from each registered country. In this case, the second country with the most zombie stations was Mexico and the third was Argentina.

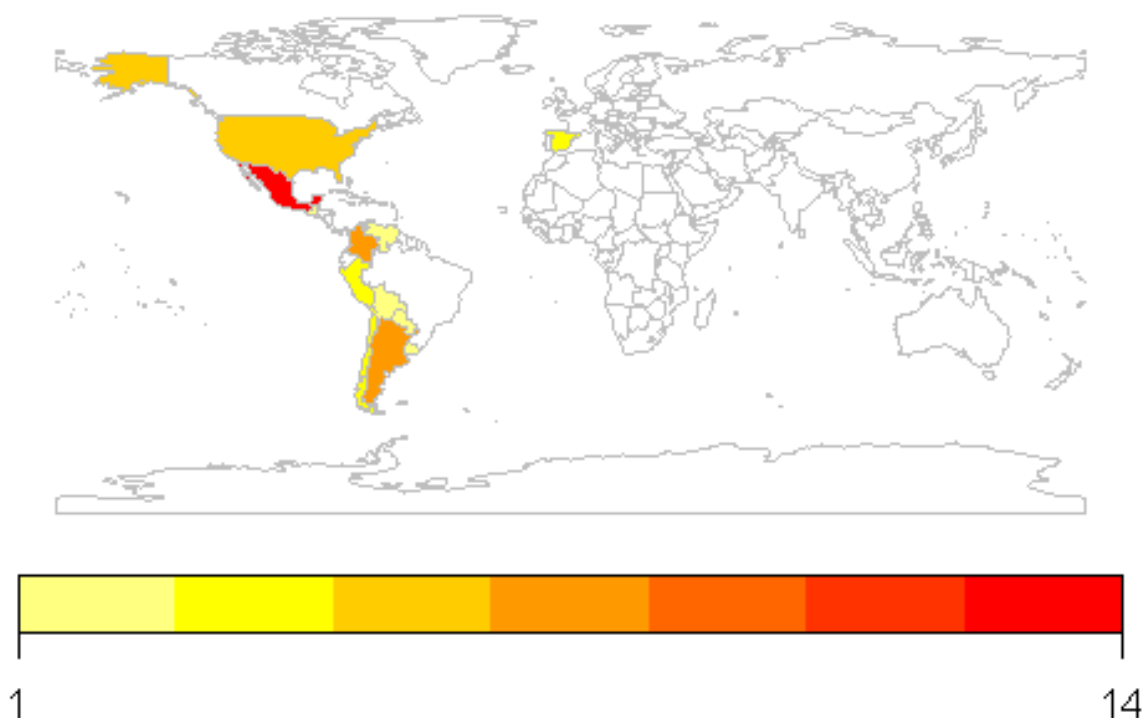


Figure 4.53: World map showing the number of connections of the different countries registered in the Challenge System Log

4.2. PHASE B: REPLICATE AND COLLECT

4.2.1. Cookie Collector Phase 1

We have proposed a novel Cookie Collector, which aims to deliver three datasets with several attributes to predict whether a visited domain is vulnerable to XSS attacks. Figure 4.54 shows the **Phase 1** whose operation is explained below.

Web search histories of Google Chrome, Mozilla Firefox, and Brave were collected from a Computer Science Department lab (30 computers) at Universidad de las Fuerzas Armadas-ESPE in Santo Domingo, from 2019 to 2024 as the first step ① of the research process.

In the second step ②, we analyzed all the attributes of the web search databases obtained from each computer. Table 4.18 provides comprehensive details of the databases' location, tables, and the attributes that contain web search history information for each browser.

Using this information, we create a total database containing all the web search history of the three browsers mentioned above. This is presented in the third step ③ titled **Total**

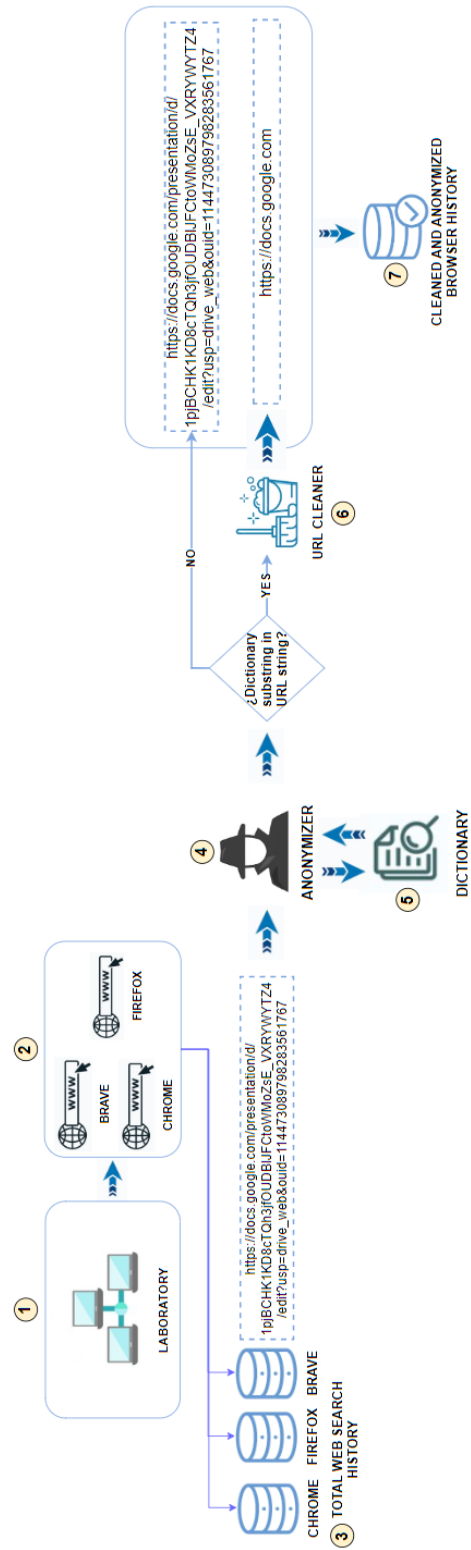


Figure 4.54: CookieCollector Phase 1: The system can anonymize and clean data from URLs in web search history.

Web Search History.

Table 4.18: Location of database containing web search history for Chrome, Firefox, and Brave browsers

| Browser | Location | Database Table | | Attribute |
|---------|---|----------------|---------|-----------|
| | | Name | Name | |
| Mozilla | C:/../AppData/Roaming/Mozilla/Firefox/ | places | moz | url |
| Firefox | Profiles/(default) | | _places | |
| Google | C:/../AppData/Local/Google/Chrome/User | History | urls | url |
| Chrome | Data/Default | | | |
| Brave | C:/../AppData/Local/BraveSoftware/Brave-Browser/User Data/Default | History | urls | url |

We have taken as reference the **Guide for the treatment of personal data in public administration** [219], textually mentions that: *"The authorization of the holder shall not be required when a public institution develops the processing of personal data and has a statistical or scientific purpose; of health protection or security; or is carried out as part of a public policy to guarantee constitutionally recognized rights. In this case, the measures leading to the suppression of the identity of the data subjects must be adopted"*.

In this context, all history data collected in **Total Web Search History** were anonymized using a python script **ANONYMIZER** ④ and following a search treatment of matching first names, last names, usernames, and their combinations within the visited URLs.

A data dictionary **DICTIONARY** ⑤ with all the records of first names, last names, user names, and their combinations, using the student lists provided by the director of the information technology course.

Any match of the content of this dictionary with the content of the parsed URL was searched. If a match was found, the URL was passed through the **URL CLEANER** ⑥, which was another Python script that cleaned up all the characters in the URL and only left the domain name in the format http://domain.com, https://domain.com, https://www.domain.com or http://www.domain.com. The repeated domains were also removed within this process, and the **CLEANED AND ANONYMIZED BROWSER HISTORY** ⑦ was obtained.

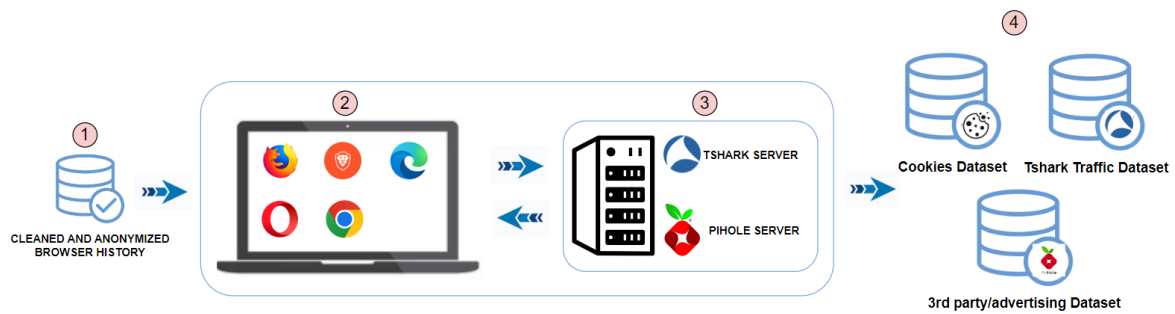


Figure 4.55: CokieCollector Phase 2:Configured test infrastructure to search the URL and extract the cookies-related, generated network traffic and third party domains.

4.2.2. Cookie Collector Phase 2

With the clean and anonymous database ①, we fed the **Phase 2** of our Cookie Collector, as shown in Figure 4.55. This infrastructure was configured with a Windows OS computer with an internet connection, on which the following web browsers were installed: Google Chrome, Mozilla Firefox, Microsoft Edge, Brave, and Opera ②. On the other side, a server with Ubuntu OS was configured ③ with the following active services: PiHole [220] and Tshark [221]. The basis of our proposal was to collect all the cookies generated by each URL (Cookies Dataset), capture the traffic generated while the bot executed the search (Tshark Traffic Dataset), and finally, associate this information with the communications that were established with third-party or advertising domains (3d party/advertising Dataset) ④.

The objective of Phase 2 was to automate the search for each of the URLs through the installed web browsers, simulating the behavior of a real user. It is here where our contribution is appreciated, by means of 5 bots (one for each browser) programmed in Python, and that were automated to execute the sequence shown in Figure 4.56 that we have named **BotSoul**.

To complement the information that each bot collected, we have associated the search for each URL with its HTTP header (HTTP headers correspond to the information transmitted between each web browser and the website when the bot simulated its visit). These headers were part of establishing a request to obtain a response from a server or web platform.

In our evaluation of traffic analyzers used to capture web search data generated by bots, we examined their characteristics and summarized them in Table 4.19. After careful consideration, we selected Tshark for its ability to capture network data packets in real-time and compatibility with Ubuntu. Furthermore, it allowed us to remotely obtain all connection

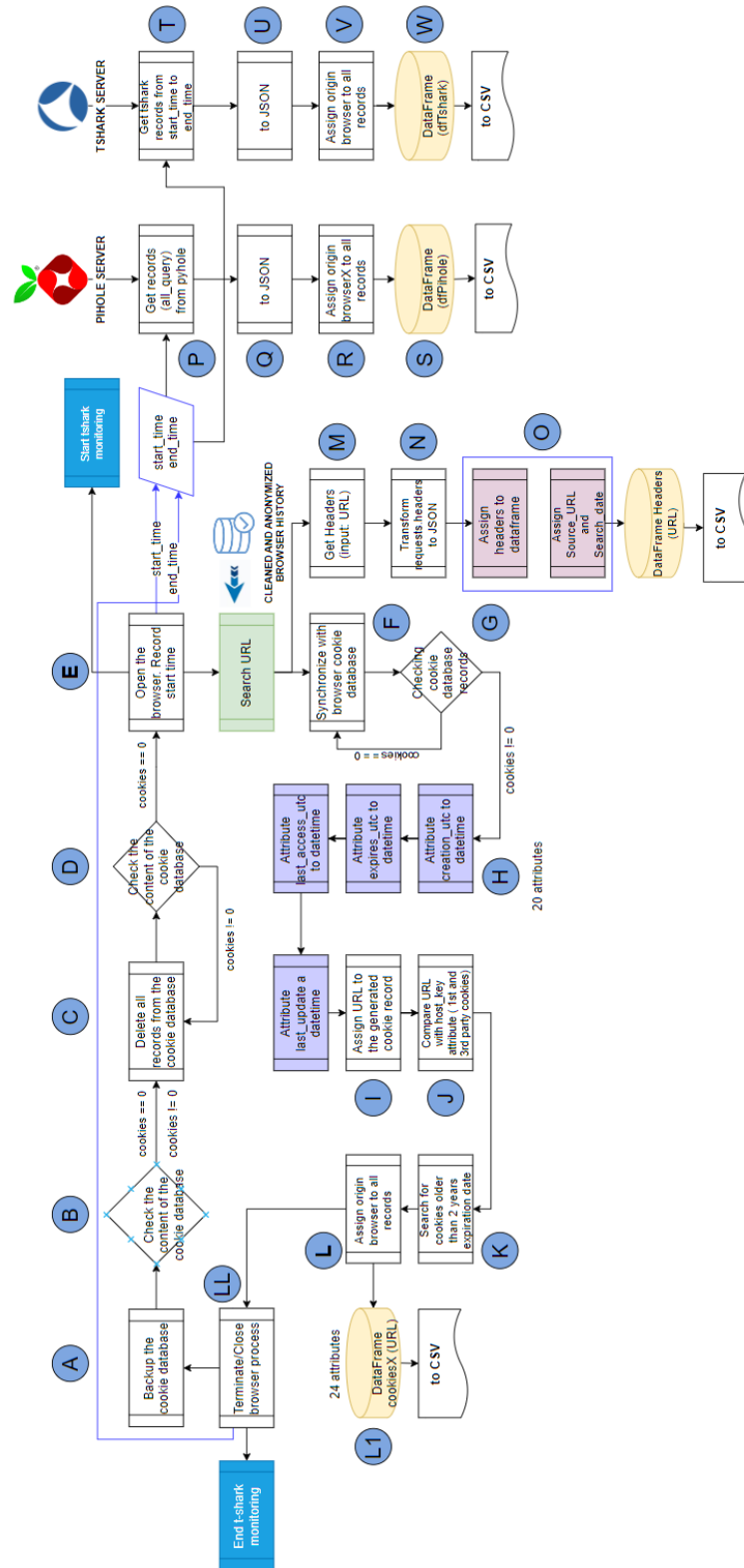


Figure 4.56: BotSoul, graphical description of how our cookie collector operates by automating bots in python.

attributes through commands programmed in each bot.

Table 4.19: Evaluation summary of programs used to capture and analyze network traffic

| Name | Type | User Interface | Limitations |
|---------------|---|--------------------------|---|
| Wireshark | Network Packet Analyzer | Graphical user interface | Problems integrating with our bot using python commands |
| Scapy | Packet Handling Tool | Python compatible | Oriented to the manipulation of network packets |
| NtopNG | Next Generation Network Traffic Monitor | Graphical user interface | Oriented to the analysis of network flows |
| Requests | Python library that makes working with HTTP requests easier | Python compatible | Oriented to sending and receiving data from websites |
| Curl | Allows to make requests to a server and retrieve data from the command line | Python compatible | Used to create network requests to transfer data across a network |
| Wget | command line tool to download files from the internet | Python compatible | Functional only to recover or duplicate an entire website. |
| Network Miner | Forensic network analysis tool | Graphical user interface | Functional to analyze .pcap files (traffic already collected) |
| Arkime | Search and capture system for indexed packets | Graphical user interface | Difficult to handle, problems interfacing with our bot |
| Tshark | Network protocol analyzer | Python compatible | – |

We have evaluated some open-source Robotic Process Automation (RPA) type tools for bot programming. These tools allowed us to automate repetitive and rule-based tasks using software platforms. We have programmed software robots, also known as "bots,"

to simulate human web search actions. Table 4.20 summarizes the characteristics of the evaluated programs used for bot programming. PyAutoGUI [222] was selected because it is a cross-platform GUI automation Python module for humans. It is used to programmatically control the mouse and keyboard.

Table 4.20: Evaluation summary of RPA programs used to test the programming of bots

| Name | Type | Interface Type | Orientation | Control of user's web browsers? |
|---------------------|--|----------------------------|---|--|
| UiPath | Open Source RPA tool | Visual Design | De-Technical and No Technical users | YES |
| Automation Anywhere | Comprehensive platform | User-friendly interface | – | – |
| Blue Prism | Enterprise-grade automation | Drag-and-drop interface | – | – |
| Open RPA | Lightweight open source RPA tool | Simplicity and ease of use | Suitable for smaller automation tasks | – |
| TagUI | Open source RPA | Command execution | For automating web interactions and data extraction | YES |
| Robot Framework | Generic open source automation framework | Command Line Interface | Making it versatile for various automation needs | JavaScript based technology called Playwright. |
| Taskt | Free C# program, built using the .NET | Drag-and-drop interface | Automate processes without any coding | – |

Table 4.21 provides a summary of the program characteristics that were assessed to

gather information from cookies generated when a bot visited a URL in the web search history. The evaluated programs were not able to capture all cookie attributes. As a result, a decision was made to simulate a person visiting a website by opening a browser, entering the web address, and pressing Enter. This is how our algorithm, BotSoul, was developed.

Table 4.21: Evaluation summary of programs used to capture/analyze cookies

| Name | Type | ¿Capture cookies in real time? | ¿Get stored cookies? | Show all cookie attributes? |
|---------------|---|--------------------------------|----------------------|-----------------------------|
| Wireshark | Network Packet Analyzer | YES | NO | NO |
| Scapy | Packet Handling Tool | YES | NO | NO |
| NtopNG | Next Generation Network Traffic Monitor | NO | NO | NO |
| Requests | Python library that makes working with HTTP requests easier | YES | NO | NO |
| Curl | Allows to make requests to a server and retrieve data from the command line | YES | NO | NO |
| Wget | command line tool to download files from the internet | NO | NO | NO |
| Network Miner | Forensic network analysis tool | NO | YES | NO |
| Arkime | search and capture system for indexed packets | NO | NO | NO |

BotSoul Algorithm Operation

Our bots were programmed to search and associate, automatically and autonomously, each searched URL, obtain the cookies generated, capture the traffic resulting from this search (Tshark), and obtain information on connections with third parties (PiHole) through advertising domains. All generated attributes were stored in the database **Final DataBase (Cookies, Tshark Traffic and 3rd party connections)** ④.

Figure 4.56 shows a general view of how the soul of each bot is composed (5 in total, 1

for each web browser). In the Algorithm 3, we explain the operation of each stage of one of the bots we use to control the Google Chrome browser (the same logic was applied to the other bots and browsers).

Algorithm 3 Pseudocode - operation of the BotSoul Algorithm for collecting cookies

```

1: startCaptureTime = 0;
2: endCaptureTime = 0;

3: while Web Browser == Open do
4:   cookiesConnect;
5:   cookiesBackup;
6:   cookiesCheck1;
7:   cookiesDelete;
8: end while

9: procedure COOKIESCONNECT() ▷ Cookies data base initial conection
10:   Connect to cookies data base;
11:   Test connection;
12:   if connection == OK then
13:     Message: Successfully Connection to cookies database
14:   else
15:     Message: Database is locked, close the browser
16:   end if
17: end procedure

18: procedure COOKIESBACKUP() ▷ A
19:   Copy cookies database to %TEMP% directory;
20:   Assign a name to the copied database;
21: end procedure

22: procedure COOKIESCHECK1() ▷ B
23:   if cookies != 0 then
24:     Message: The cookies database still contains records
25:     cookiesDelete;
26:   else
27:     Message The cookies database is empty
28:     cookiesDelete;
29:   end if
30: end procedure

31: procedure COOKIESDELETE() ▷ C
32:   if SELECT count(*) FROM cookies = 0 then
33:     DELETE * from cookies;
34:   else if SELECT count(*) FROM cookies != 0 then
35:     print:Message: Cookie database is empty
36:   end if
37: end procedure

```

Algorithm 3 Pseudocode - operation of the Soul Bot Algorithm for collecting cookies

```
38: procedure COOKIESCHECK2() ▷ D
39:   if cookies == 0 then
40:     Message: (Cookie database is empty)
41:     openBrowser;
42:   else
43:     Message (The cookies database is not empty)
44:     cookiesDelete;
45:   end if
46: end procedure
47: procedure OPENBROWSER() ▷ E
48:   Open web browser;
49:   startCaptureTime = datetime.datetime(now);
50:   URL_search ← Cleaned_Anonymized_History
51:   Search URL_search;
52:   Start tshark monitor process;
53:   Wait 10 seconds until the page loads completely;
54: end procedure
55: procedure SINCRONYZECOOKIEADB ▷ F
56:   query = Select * from cookiesDataBase;
57:   connection = cookiesDataBase;
58:   dfCookies = read_sql(query, connection) ▷ Assign cookies a dataframe
59: end procedure
60: procedure COOKIESCHECK3() ▷ G
61:   if cookies == 0 then
62:     Message: Cookie database is empty"
63:     Refresh web browser;
64:   else
65:     Message: Cookies record found for URL;
66:     UTCtoDatetime();
67:   end if
68: end procedure
69: procedure UTCTODATETIME() ▷ H
70:   Convert Attribute creation_utc to datetime;
71:   Convert Attribute expires_utc to datetime;
72:   Convert Attribute last_access_utc to datetime;
73:   Convert Attribute last_update to datetime;
74: end procedure
75: procedure ASSIGNURL() ▷ I
76:   Assign URL_search to dfCookies ▷ Add new attribute to dataframe
77: end procedure
```

Algorithm 3 Pseudocode - operation of the Soul Bot Algorithm for collecting cookies

```
1: procedure FIRSTPARTYCOOKIES() ▷ J
2:   Compare URL_search with attribute host_key;
3:   if URL_search == host_key attribute then
4:     Message: 1st party cookie found
5:     Assign 1 to dfCookies; ▷ Add new attribute to dataframe
6:   else
7:     Message: 3rd party cookie found;
8:     Assign 0 to dfCookies; ▷ Add new attribute to dataframe
9:   end if
10: end procedure
11: procedure EXPIREDCOOKIES ▷ K
12:   if expire_date - creation_date > 2 then ▷ Compare 2 attributes
13:     Message: cookies with an expiration date greater than 2 years found
14:   else
15:     Message: cookies with expiration date less than 2 years found
16:   end if
17:   Assign (expire_date - creation_date) to dfCookies; ▷ Add new attribute to dataframe
18: end procedure
19: procedure ASSIGNBROWSER ▷ L
20:   Assign browser_origin to dfCookies; ▷ browser_origin options: chrome, firefox, brave, opera or edge
21: end procedure
22: procedure EXPORTCOOKIES ▷ L1
23:   Export dfCookies to dfCookies.csv ▷ 24 or more attributes, depends on the browser running
24: end procedure
25: procedure FINISHPROCESS ▷ LL
26:   Terminate and close all running browser processes
27:   endCaptureTime == datetime.datetime(now);
28:   Finish tshark monitor process;
29: end procedure
30: procedure GETHEADERS ▷ M
31:   r ← request.get(URL_search); ▷ Make HTTP request to URL
32:   headers = r.headers ▷ Capture metadata information from URL
33:   headers_json ← headers ▷ N
34:   headers_df ← headers_json;
35:   headers_df assigns new attribute (browser_origin)
36:   headers_df assigns new attribute (search_date) ▷ O
37:   headers_csv ← headers_df
38: end procedure
```

Algorithm 3 Pseudocode - operation of the Soul Bot Algorithm for collecting cookies

```
1: procedure GETPIHOLERECORDS
2:   startTime  $\leftarrow$  startCaptureTime;
3:   endTime  $\leftarrow$  endCaptureTime;
4:   for (startTime to endTime) do
5:     queryPiHole  $\leftarrow$  GetallPiHoleRecords(all_query)           ▷ P
6:   end for
7:   pihole_json  $\leftarrow$  queryPiHole                               ▷ Q
8:   Assign new attribute browser_Origin to pihole_json;          ▷ R
9:   pihole_dataframe  $\leftarrow$  pihole_json;                         ▷ S
10:  pihole_csv  $\leftarrow$  pihole_dataframe;
11: end procedure
12: procedure GETTSHARKRECORDS
13:  startTime  $\leftarrow$  startCaptureTime;
14:  endTime  $\leftarrow$  endCaptureTime;
15:  for (startTime to endTime) do
16:    queryTshark  $\leftarrow$  GetallTsharkRecords                       ▷ T
17:  end for
18:  tshark_json  $\leftarrow$  queryTshark                               ▷ U
19:  Assign new attribute browser_Origin to tshark_json;          ▷ V
20:  tshark_dataframe  $\leftarrow$  tshark_json;                         ▷ W
21:  tshark_csv  $\leftarrow$  tshark_dataframe;
22: end procedure
```

Our collector uses Python scripts to connect with the cookie database of each browser. A separate script is used for each browser, and the path to the directory where the cookie records are stored is shown in Table 4.22. We start by creating a backup of the cookie database for the designated browser, as the database is blocked when the browser is running and cannot be accessed by other applications.

4.3. PHASE C: TRAINING AND PREDICT

4.3.1. Training

Three large datasets containing essential information were obtained as the final result of the collection phase. It's important to note that the input database for this phase was a clean and anonymous database of search histories (domains). As a result, the first dataset was structured to contain an association of these domains with the information of the cookies that were created when the bot performed a web search. This search action allowed us to collect all types of cookies of interest: first-party, third-party, and session cookies.

The second dataset stored the association of these domains with the generated traffic

Table 4.22: Location of the database that contains the cookies for each web browsers used

| Browser | Location | Database Table | | No. Attributes |
|-----------------|--|------------------|-----------------|----------------|
| | | Name | Name | |
| Mozilla Firefox | /AppData/Roaming/Mozilla/Firefox/Profiles/ user.default-release | cookies | moz _cookies | 15 |
| Google Chrome | /AppData/Local/Google/Chrome/User Data/Default/Network | Cookies | cookies | 19 |
| Brave | /AppData/Local/BraveSoftware/Brave- Browser/User Data/Default/Network | Cookies | cookies | 19 |
| Opera | /AppData/Roaming/Opera Software/Opera Stable/Default/Network | Soft- Cookies | cookies | 19 |
| Edge | /AppData/Local/Microsoft/Edge/User Data/Default/Network | Cookies | cookies | 21 |

(tshark) when the bot searched for that domain using the web browser. A third dataset was created, which associated these domains with the advertising domain records (pihole) generated while the bot was browsing the selected website.

Figure 4.57 provides a preview of the databases, their relationships, and the obtainable attributes, all of which were used as input for the training and prediction phase.

In the initial test, each bot searched 611 domains over 4 hours (approximately 152 domains per hour), generating roughly 4,719 cookies. This means that each domain produced around seven cookies: one first-party and six third-party cookies.

We analyzed all web browsing logs collected from the five computer laboratories of the Universidad de las Fuerzas Armadas, using 440000 unnormalized URLs (40000 normalized domains) as the search base for all bots. In total, it took us approximately 10,9 days to complete the search for the information captured in Figure 4.57, with 5 bots, each working with a different browser 24 hours a day.

With the total cookie database, the logic of sub-phase 1 (CookieScout) and sub-phase 2 (DataCookie) of the characterization phase was applied. First, the algorithm was run to analyze the attributes of all the cookies generated and classify them according to the attribute suspicious for XSS attacks (suspicious / not suspicious). Second, the logic of the classification tree algorithms was applied to train our browser, which we have called BOOKIE

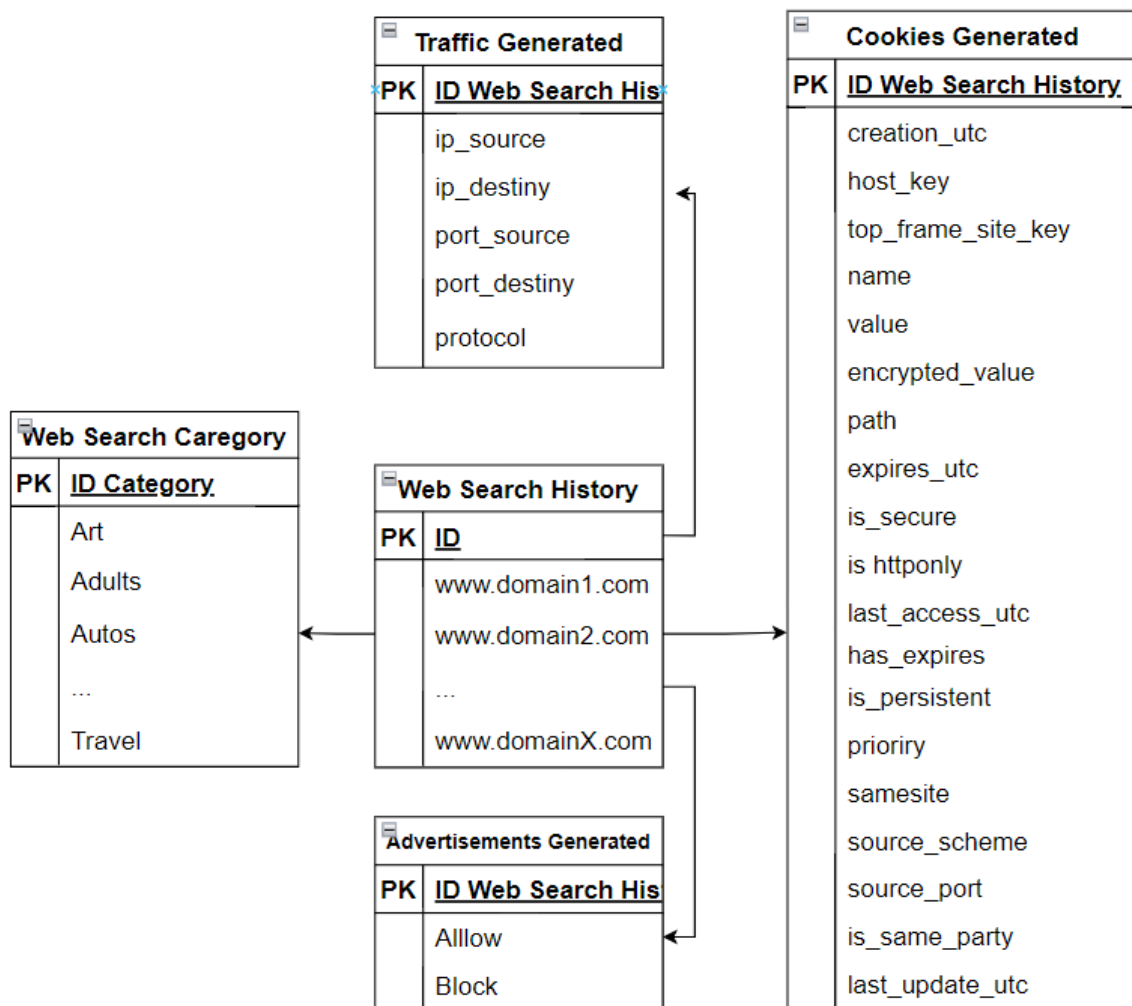


Figure 4.57: Entity relationship diagram of the datasets obtained in the collection phase

(Figures 4.58, 4.59, and 4.60), developed with Python and PyQt.

4.3.2. Prediction

To incorporate web page recommendation capability into the developed browser, we have considered the following types of clustering algorithms, taking into account the nature of the cookie data and the analysis purpose:

- K-Means
- DBSCAN
- Hierarchical Clustering

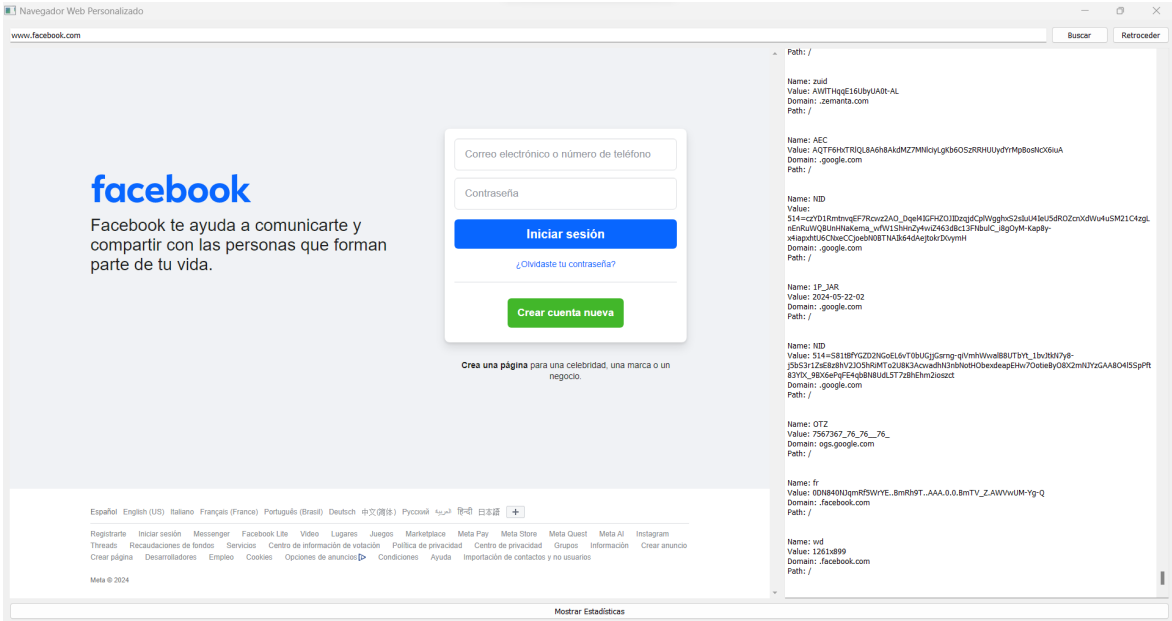


Figure 4.58: Preview of BOOKIE, a browser for search and analyse cookies

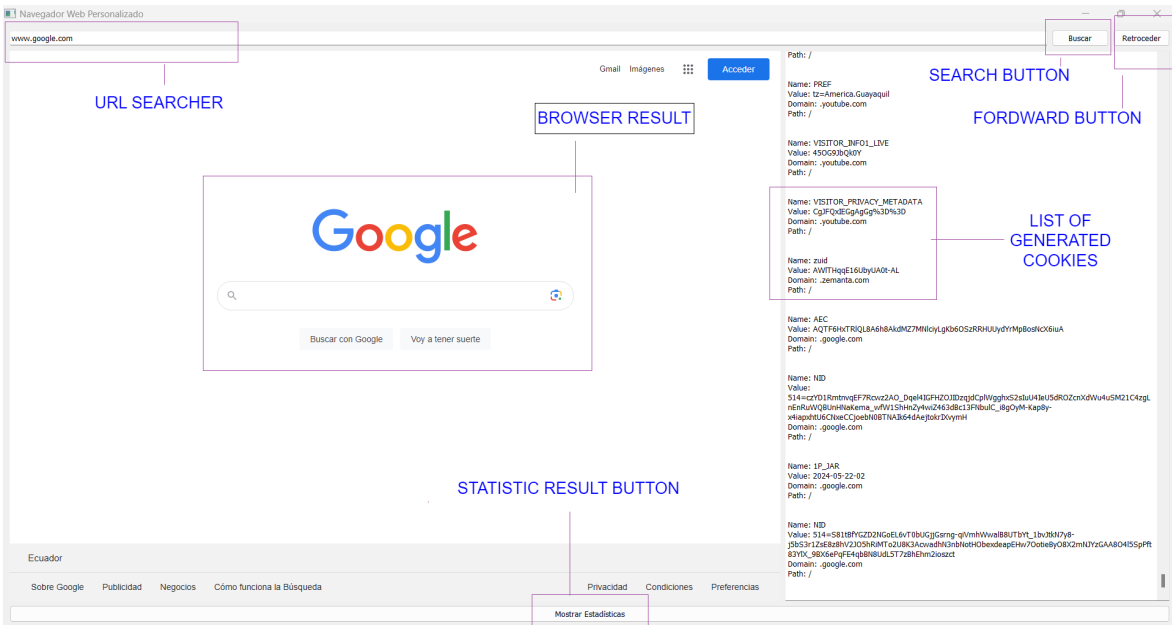


Figure 4.59: Preview of BOOKIE, a browser to predict the level of vulnerability of students to XSS attacks

K-Means

The K-Means algorithm is a clustering technique that partitions data into K clusters, where each data point belongs to the cluster with the closest mean. It is one of the simplest and most popular algorithms. Allowed the identification of common browsing patterns: Cook-

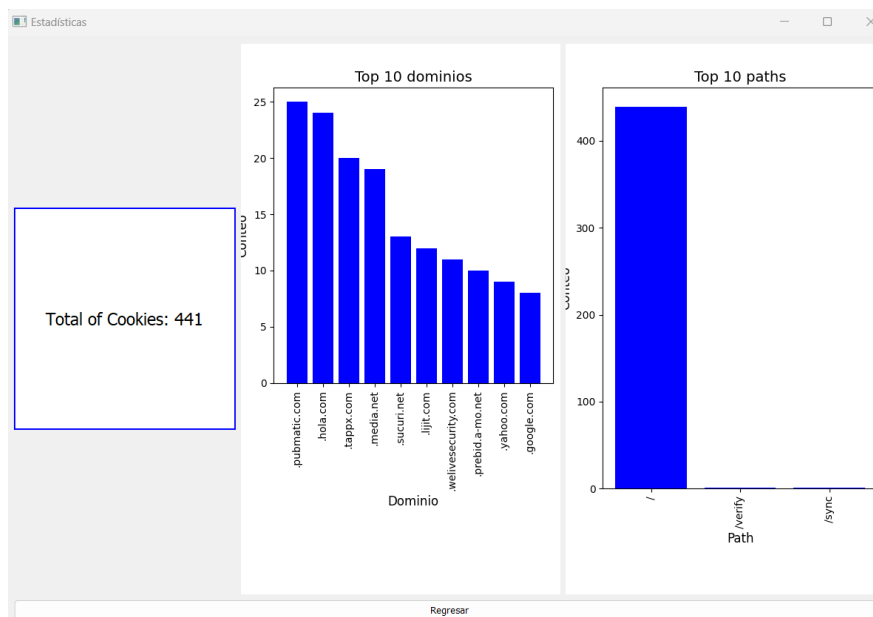


Figure 4.60: Option to display statistical results for each browsed web page

ies with similar characteristics (e.g: lifetime and domain) were grouped, allowing common browsing behaviors to be identified between users. Clusters of cookies that reflect similar behavioral patterns were obtained—identification of user segments with common interests.

Different clusters have been evaluated to perform a hyperparameter analysis for the proposed algorithm, and the evaluation metrics have been compared to determine the optimal number of clusters. In Figure 4.61, four graphs have been generated that show how the evaluation metrics vary with the number of clusters, allowing the user to quickly interpret the results and select the optimal number of clusters. We have determined that the optimal number of clusters for our analysis was 2.

The **Silhouette Score** measures how well the data is grouped within its assigned clusters and how far apart the clusters are from each other. A value close to 1 indicates that the data is well grouped and the clusters are separated.

The **Inertia or Sum of Squares within the Clusters (SSE)** measures the compactness of the clusters, that is, the sum of the square distances between each point and the centroid of the cluster to which it belongs. Lower values indicate that points are closer to their centroids, which generally means more compact clusters.

The **Calinski-Harabasz Index** measures the separation between clusters and the compaction within clusters. Higher values indicate more separated and compact clusters.

The **Davies-Bouldin Index** measures the relationship between dispersion within clusters and separation between clusters. Lower values indicate that the clusters are well-separated

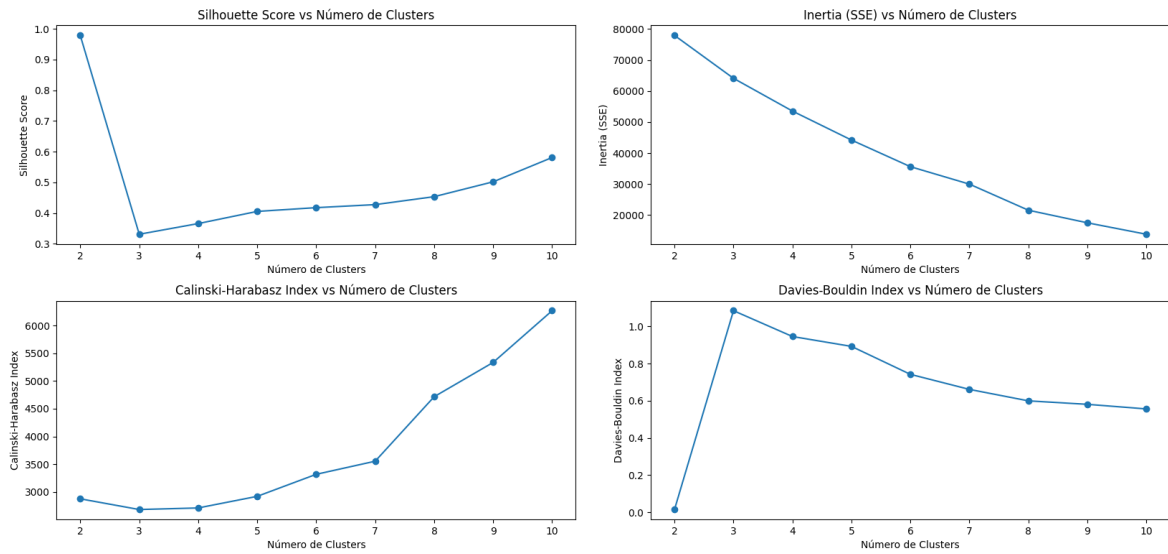


Figure 4.61: Preview of K-MEANS Hyperparameters Analysis

and compact.

Principal Component Analysis (PCA) has been applied to visualize the clusters. It is a dimensionality reduction technique that transforms a set of possibly correlated variables into a smaller set of uncorrelated variables called principal components.

Figure 4.62 shows the graph of the generated clusters. A summary of the hyperparameter evaluation using different numbers of clusters is shown in Table 4.23.

Table 4.23: Hyperparameter analysis for the KMEANS algorithms

| | Clusters = 2 | | Cluster=5 | | Clusters=10 | |
|--------------------------------|--------------|-------------|-----------|-------------|-------------|-------------|
| | With PCA | Without PCA | With PCA | Without PCA | With PCA | Without PCA |
| Silhouette Score | 0.9784 | 0.9784 | 0.2447 | 0.4056 | 0.1718 | 0.5810 |
| Inertia (SSE) | 16689.46 | 77963.1709 | 1401.95 | 44201.2 | 259.1727 | 13799.1350 |
| Calinski-Harabasz Index | 2874.24 | 2874.24 | 1805.9507 | 2918.76 | 890.5486 | 6269.1247 |
| Davies-Bouldin Index | 0.0152 | 0.0152 | 1.5933 | 0.8926 | 3.6921 | 0.5563 |

For n=2 clusters, the results indicate that the clusters are very well defined and separated, although there may be some natural dispersion within them. This is usually a sign of very successful clustering, where the data points within each cluster may be spread out, but the clusters themselves are differentiated.

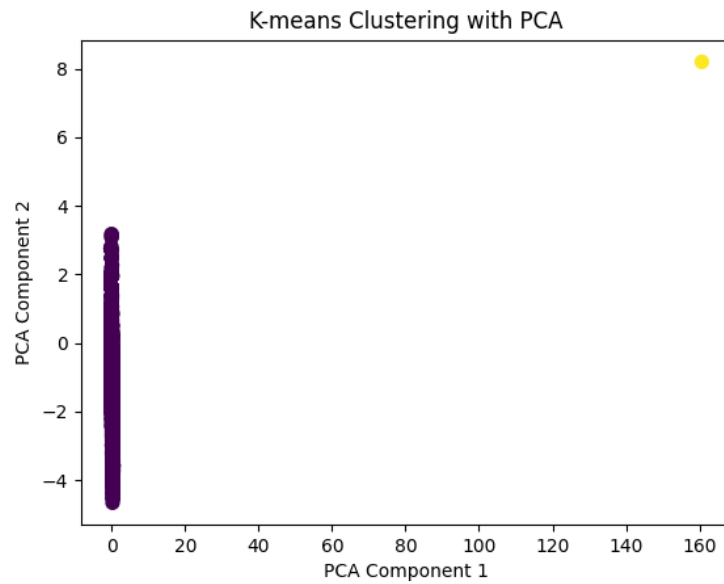


Figure 4.62: Preview of K-MEANS with 2 clusters

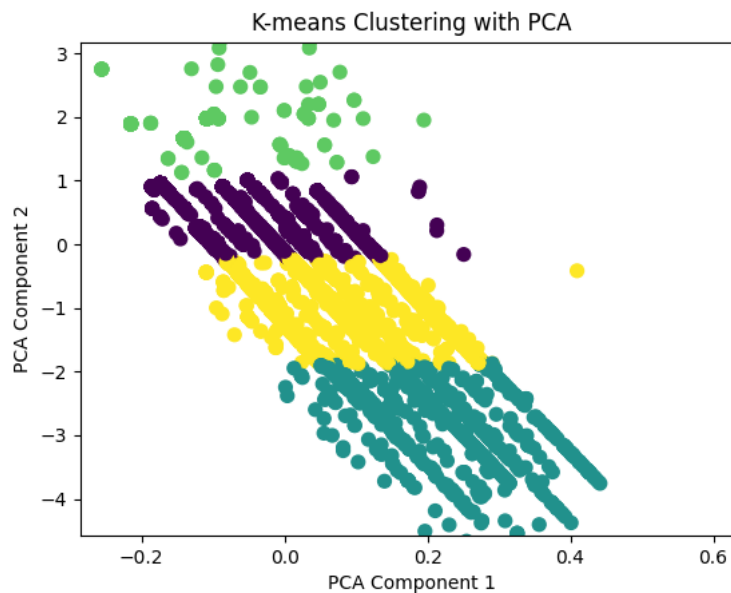


Figure 4.63: Preview of K-MEANS with 5 clusters

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN is a density-based clustering algorithm that can find arbitrary-shaped clusters and handle noise (data points that do not belong to any cluster). It allowed the detection of clusters in noisy data; it was suitable for cookie data that contained noise, such as non-relevant or inconsistent third-party cookies. Clusters of densely connected cookies were obtained, ignoring the noise and improving the handling of atypical or unusual cookies.



Figure 4.64: Preview of K-MEANS with 10 clusters

Table 4.24: Performance tests with different eps values and min_samples

| | eps = 2, min_samples=25 | | eps=5, min_samples=25 | | eps=10, min_samples=25 | |
|--------------------------------|-------------------------|-------------|-----------------------|-------------|------------------------|-------------|
| | With PCA | Without PCA | With PCA | Without PCA | With PCA | Without PCA |
| Silhouette Score | few clusters | 0.5221 | few clusters | 0.5636 | few clusters | 0.8157 |
| Calinski-Harabasz Index | few clusters | 946.0439 | few clusters | 957.4706 | few clusters | 2167.0301 |
| Davies-Bouldin Index | few clusters | 1.5209 | few clusters | 1.3103 | few clusters | 0.1706 |

For eps=10, the results indicate that the DBSCAN model has achieved a perfect data grouping. The clusters (n=2) (Figure 4.65) are well defined and separated, which is reflected in a high Silhouette Score, a high Calinski-Harabasz index, and a low Davies-Bouldin index.

Hierarchical Clustering

Build a hierarchy of clusters using an agglomerative (start with individual points and group them) or divisive (start with all points and divide them) approach. Allowed analysis of hierarchical relationships: Useful for understanding relationships between groups of cookies at different levels of granularity, which helped identify subgroups of users with specific behaviors.

With the implementation of these clustering algorithms, the results were:

- **User Segmentation:** Grouping users into clusters based on the cookies generated allows segments with similar interests and browsing behaviors to be identified.

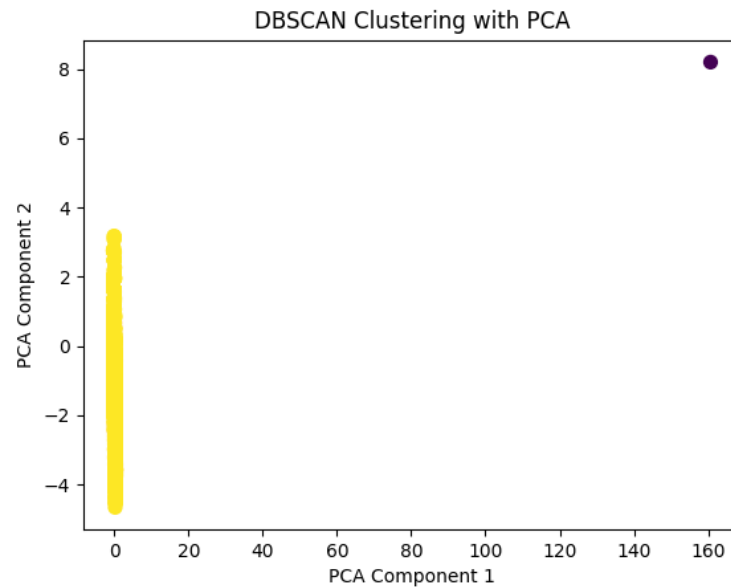


Figure 4.65: Preview of clusters identified with DBSCAN

- **Personalized Recommendations:** Based on the identified clusters, the browser can recommend web pages relevant to each cluster’s browsing patterns.
- **Anomaly Detection and Security:** Identification of cookies that do not fit normal patterns (anomalies), helping to detect possible security threats.
- **User Experience Optimization:** Improving the relevance of recommendations, providing a more personalized and efficient browsing experience for students.

Implementing these algorithms allowed the browser to offer recommendations based on historical cookie data and continually adapt its recommendations as more data was collected, thus improving the accuracy and personalization of web suggestions.

Figure 4.66 shows the dendrogram that illustrates the hierarchical groupings most derived from the hierarchical algorithm. The dendrogram represents the data groupings concerning other representations. Figure 4.67 shows the number of clusters ($n=10$) identified by the hierarchical algorithm.

The results in Table 4.25 indicate good performance of hierarchical clustering; they suggest that hierarchical clustering has identified significant groups in the data, with a good cluster structure and a clear separation between them.

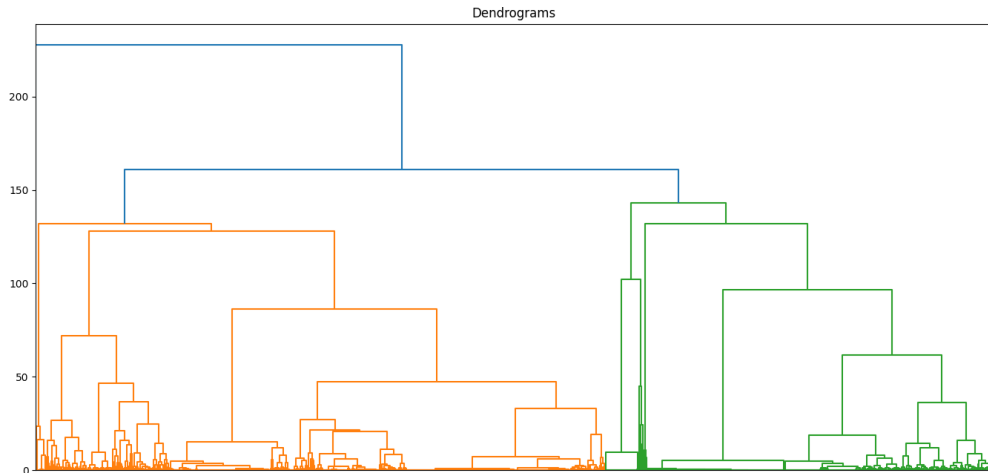


Figure 4.66: Preview of Dendograms identified with Hierarchical Clustering

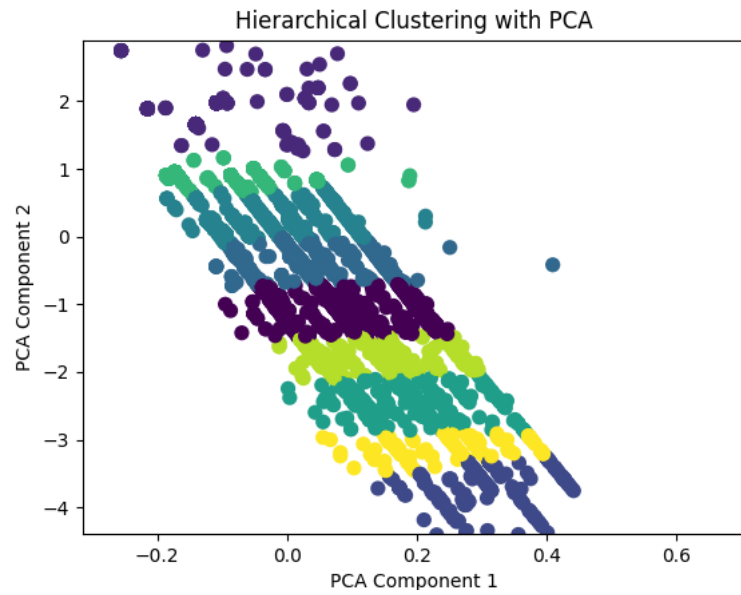


Figure 4.67: Preview of Clusters identified with Hierarchical Clustering

Table 4.25: Performance test for Hierarchical Clustering

| | With PCA | Without PCA |
|--------------------------------|----------|-------------|
| Silhouette Score | 0.1649 | 0.5653 |
| Calinski-Harabasz Index | 886.0800 | 5430.3483 |
| Davies-Bouldin Index | 3.7704 | 0.6645 |

4.4. PHASE D: VISUALIZATION

4.4.1. Methodological proposal through a pen testing challenge

Penetration testing [223] is a cybersecurity practice where a professional simulates a cyber-attack to identify vulnerabilities and test breach security. The pen testing process follows a phase-oriented approach based on the National Institute of Standards and Technology (NIST) rules [224]. ISO 27000 Standard suggests the "Plan, Do, Check, Act" [225] methodology to effectively protect information. This involves evaluating risks, implementing safeguards, reviewing and making necessary changes.

Our methodological proposal is based on two methodologies that can be used to execute a penetration test, as shown in Figure 4.68.

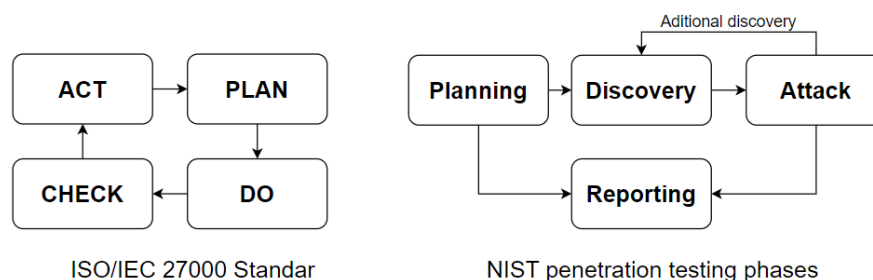


Figure 4.68: NIST methodology for penetration testing and ISO 27000 standard for safeguarding information

Analysis of the proposed methodologies for teaching computer security (Figure 4.69) shows that Xinli et al. [224] taught computer security to college students through case studies and course projects. Our proposal aims to teach security concepts to university students through a complex challenge.

Our study, conducted at the University of the Armed Forces ESPE, proposed a new approach called "**HackMyself**" to teach about cookie theft via XSS attacks. The methodology was evaluated during two academic periods: October 2022 to March 2023 (period 202251) and October 2023 to March 2024 (period 202351).

A security challenge similar to a black box pen testing [225] was proposed for students to analyze their personal computer's web browsers without prior knowledge. The challenge simulated an external third-party attack, and students had no prior knowledge of computer security.

The challenge posed on the first day of classes was the following:

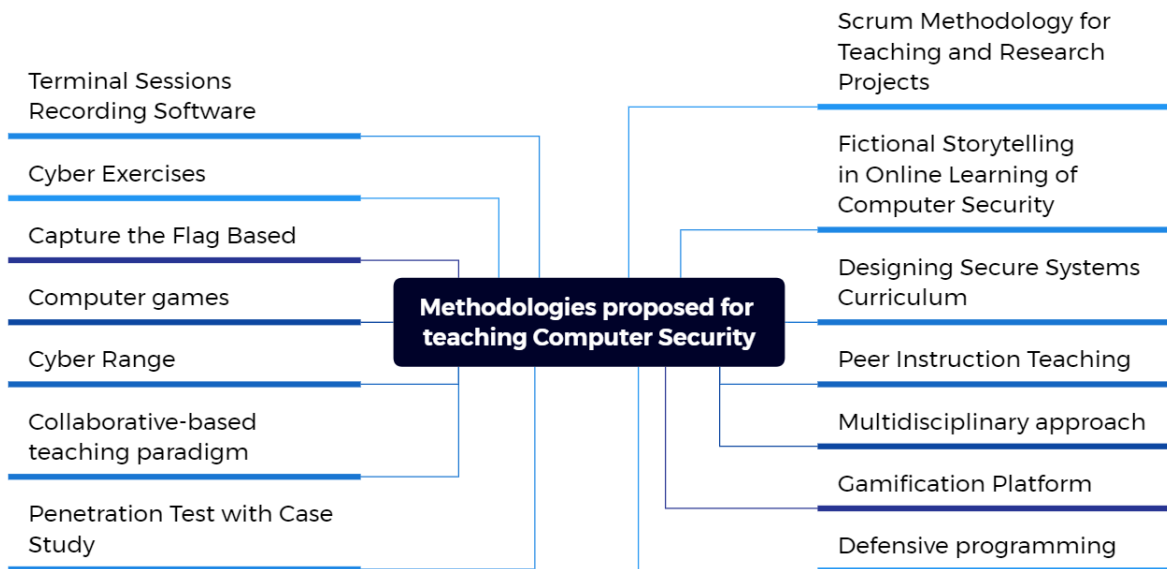


Figure 4.69: Resume of proposed methodologies to teach computer security.

"Decrypt the content of cookies in the web browsers of students' personal computers and display the results on a local web page".

Over a six-month academic period, 16 students from period 202251 and 26 students from period 202351 studied Threat and Vulnerability Analysis (partial 1), Fundamentals of Cryptography (partial 2), and Network Security (partial 3). The first partial covered website vulnerability analysis, including XSS attacks and cookies.

Figure 4.70 shows a recommended panel structure for presenting challenge results. The aim was to display all analyzed data (personal data discovered) on a local web page.

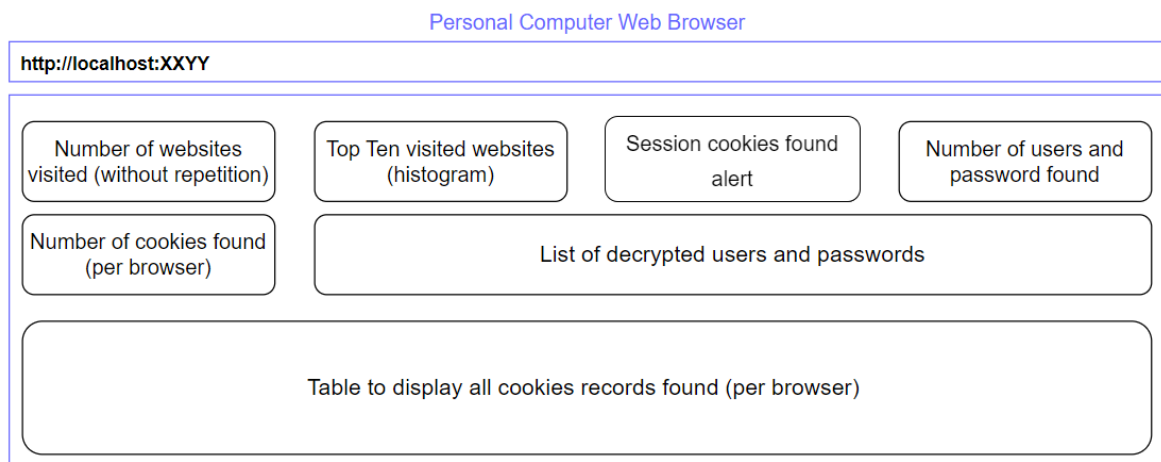


Figure 4.70: General structure to present the resolution of the challenge posed.

The student had to get their browsing history and cookie records from all installed browsers, including Chrome, Firefox, Edge, and Brave. An incentive was given to those who retrieved this information successfully.

The students applied the methodological structure shown in Figure 4.71 to solve a security challenge similar to black box pen testing.

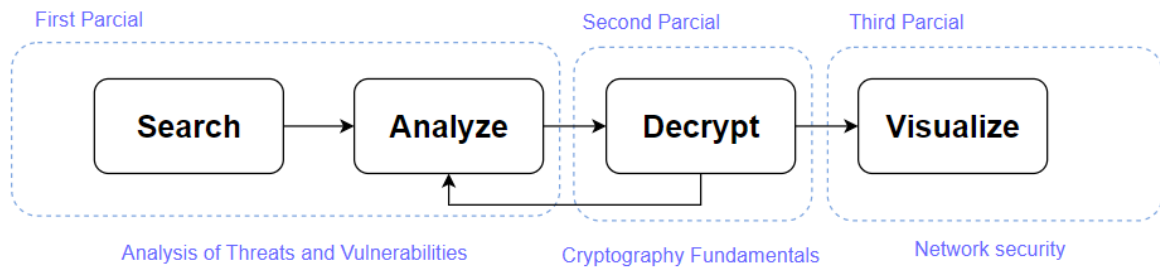


Figure 4.71: HackMyself - Proposed methodology to motivate learning about cookie theft and XSS attacks.

The first two phases, **Search** and **Analyze**, were about identifying threats and vulnerabilities (cookie theft and XSS attacks). Students learned to mitigate them through computer security technologies.

In the third phase **Decrypt**, students learned about cryptography fundamentals and popular algorithms. By the end of the second partial, they were able to apply these concepts

In the last phase **Visualize**, students learned about network security and implementing computer security technologies, such as firewalls, IDS/IPS, and honeypots to visualize the threat activities.

4.4.2. Using statistical methodology to analyze computer security challenge results.

42 students were evaluated based on the final exam grades of each partial (3 variables) and the average evaluation of the challenge (1 variable). The challenge was evaluated as the Final Project of each partial.

We tested two statistical techniques, using Python scripts, to determine when students mastered the skills to solve the security challenge, analyzing the relationship between relevant variables.

- **Linear Regression:** was used to analyze the correlation between the average chal-

lence grade and the partial grades in each unit, to identify any significant association between partial grades and successful resolution of the challenge.

- **Hypothesis test:** we have raised the following hypothesis: *“There is a significant difference between the average grade for solving the challenge and at least one of the grades for the three midterms”*. The t-Student statistical test was applied to evaluate whether this hypothesis was supported by the grade data obtained.

4.4.3. Pen testing results

Figure 4.72 displays the students’ final exam results for each partial and how they relate to the average grade they received for solving the proposed challenge.

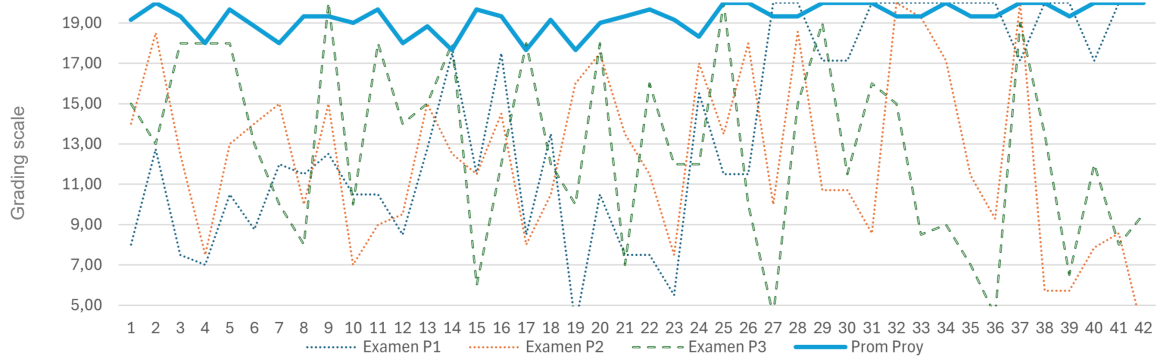


Figure 4.72: Summary of the grades obtained in each partial and the average grades of the challenge solved

Regarding the following equivalence scale: Excellent (18.50-20), Very Good (16-18.49), Well (14-15.99), Regular (13.50-13.99), and Poor (0-13.49), it was observed that in the first partial, 59% of students had a Poor performance and only 29% had an Excellent performance. In the second partial, 57% had a Poor performance, and only 12% had an Excellent performance. Finally, in the third partial, 55% had a Poor performance, and only 10% achieved an Excellent performance.

If we compare the performance of all the students partially with the average grade of the project, which corresponds to the evaluation of their challenge, we find that 83% of students had an Excellent performance. The remaining 17% had a Very Good performance.

Two figures, labeled as Figure 4.73 and Figure 4.74, display the outcome of a project carried out by a student group who successfully decrypted the cookies they found on their

computers. The rubric illustrated in Table 4.26 was employed to assess the progress of their work.

Table 4.26: Rubric used to evaluate the challenge results

| Dashboard Design | Decrypt process | Browsers History found | Users and Passwords found | Total |
|------------------|-----------------|------------------------|---------------------------|-----------|
| 0 - 5 | 0 - 5 | 0 - 5 | 0 - 5 | 20 points |

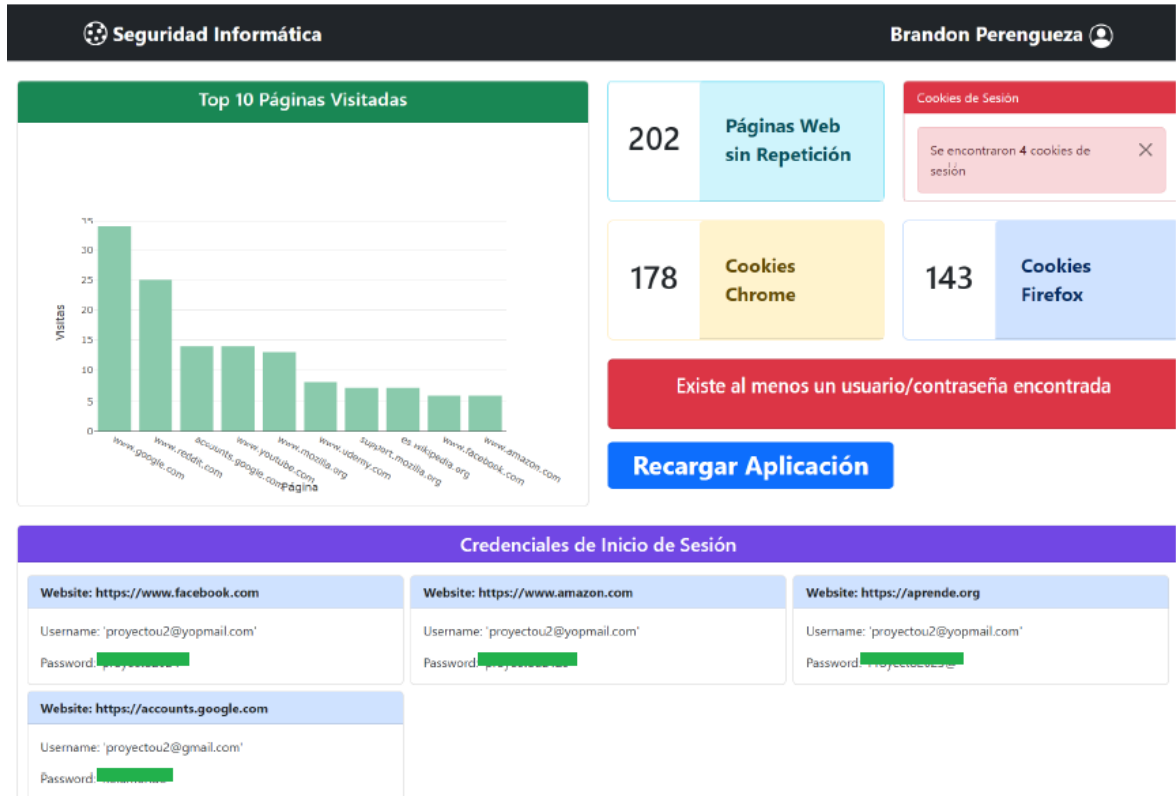


Figure 4.73: Presentation of the solved challenge in dashboard format, average rating 19/20.

| No. | Nombre | Dominio | Path | Creado | Expira | Último Acceso |
|-----|------------------------|-------------------|------|----------------------------|---------------------|----------------------------|
| 1 | WMF-Last-Access | www.wikipedia.org | / | 2024-02-03 18:36:52.112853 | 2024-03-06 07:00:30 | 2024-02-03 18:36:52.112853 |
| 2 | WMF-Last-Access-Global | wikipedia.org | / | 2024-02-03 18:36:52.112853 | 2024-03-06 07:00:30 | 2024-02-03 18:36:52.112853 |
| 3 | NetworkProblemit | www.wikipedia.org | / | 2024-02-03 18:36:52.112155 | 2024-02-03 19:12:15 | 2024-02-03 18:36:52.888881 |
| 4 | ._ga | concepto.de | / | 2024-02-03 18:37:08.119419 | 2028-02-02 18:37:38 | 2024-02-03 18:37:38.119419 |
| 5 | ._gid | concepto.de | / | 2024-02-03 | 2024-02-04 | 2024-02-03 |

| No. | Nombre | Dominio | Path | Creado | Expira | Último Acceso |
|-----|--------|-----------------|------|----------------------------|----------------------------|----------------------------|
| 1 | CMPS | casalemedia.com | / | 2024-02-05 14:48:08.884148 | 2024-05-05 14:48:08.884148 | 2024-02-08 09:50:40.196297 |
| 2 | FCNEC | abriflavo.com | / | 2024-02-05 14:48:08.870776 | 2025-02-04 14:48:08.000000 | 2024-02-05 16:59:32.588585 |
| 3 | ._gxs | abriflavo.com | / | 2024-02-05 14:48:08.899905 | 2025-03-01 14:48:08.000000 | 2024-02-05 16:59:32.588585 |
| 4 | ._gpi | abriflavo.com | / | 2024-02-05 14:48:08.034220 | 2025-03-01 14:48:08.000000 | 2024-02-05 16:59:32.588585 |
| 5 | ._ga | abriflavo.com | / | 2024-02-05 | 2025-03-11 | 2024-02-05 |

Figure 4.74: Presentation of the solved challenge in a dashboard format, including a list of cookies found by the browser.

4.4.4. Knowledge assessment results

Linear regression:

Table 4.27: Linear regression coefficients

| | | | |
|-------------------------------|------------------|------------------|------------------|
| Intercept: | 18,36 | | |
| Variable coefficients: | Partial 1 | Partial 2 | Partial 3 |
| | 0.0629 | -0.0051 | 0.0077 |

The intercept in linear regression represents the expected dependent variable value when all independent variables are zero. In this case, the intercept of 18.36 suggests the final challenge grade is expected to be approximately 18.36/20 points if all partial grades were zero.

Variable coefficients show how partial grades affect the final grade. Positive coefficients mean an increase in partial grade leads to an increase in the final grade, while negative coefficients indicate an inverse relationship.

- A 1-point increase in Partial 1 grade results in around a 0.0629-point increase in the final challenge grade.
- A 1-point increase in the Partial 2 score results in a decrease of about 0.005 points in the final challenge score.
- A 1-point increase in Partial 3 grade results in around a 0.0077-point increase in the final challenge grade.

The coefficients were interpreted under the assumption that the other variables remained constant. For instance, the interpretation of the partial 1 coefficient assumed that the scores of partial 2 and partial 3 did not change.

Hypothesis testing:

The proposed methodology analyzed the influence of the student's midterm performance on the project grade. Two hypotheses were proposed.

- **Null hypothesis (H0):** The average challenge grade is not affected by the grades of the three partials, indicating that performance in each part does not impact the resolution of the challenge.
- **Alternative hypothesis (H1):** If there is a notable difference between the average grade for solving the challenge and any of the grades for the three partials, we can identify which partials impacted the resolution of the challenge the most.

Our Python script uses the t-Student test to compare the grades of each partial with the average grade of challenge and determine if there are any significant differences that allow us to reject the null hypothesis. The script presents the comparisons at a 0.05 significance level, meaning that we accept a 5% chance of making a type I error by incorrectly rejecting the null hypothesis. The results of this analysis are shown in table 4.28.

Table 4.28: Result of the analysis of the hypotheses using the t-Student method

| | First Partial | Second Partial | Third Partial |
|--------------------|-----------------------------|-----------------------------|-----------------------------|
| t-Statistic | -6.8074 | -10.3255 | -9.1392 |
| p Value | 1.5117e-09 | 1.7029e-16 | 3.8024e-14 |
| | Null hypothesis is rejected | Null hypothesis is rejected | Null hypothesis is rejected |

First Partial: The first partial grades average is significantly lower than the project (challenge) grades average. The low p-value (1.5117e-09) shows a significant difference between the two samples. Hence, the null hypothesis is rejected, indicating that the student's performance in the first partial influenced the proposed challenge's resolution.

Second Partial: The second midterm's average grade is significantly lower than the project average, as shown by the negative t statistic. The extremely low p-value (1.7029e-16) confirms a significant difference between the two samples. Therefore, the null hypothesis is rejected, indicating that the students' performance in the second partial also affected the resolution of the proposed challenge.

Third Partial: The t-test shows that the grades for the third partial are significantly lower than the average for the project (challenge). The very low p-value confirms a significant difference between the two samples, so the null hypothesis is rejected. This suggests that the student's performance in the third partial also influenced the resolution of the proposed challenge.

Our study demonstrated the successful decryption of cookies by students without com-

promising privacy or sharing any findings with external parties. However, the decryption key was found in the same computer and directory as the stored cookies, leading us to emphasize the vulnerability of such setups.

The students' experience highlighted the critical need to clear cookies regularly to mitigate potential risks associated with the storage of personal data. Moreover, we quantitatively evaluated the students' knowledge acquisition in cookie decryption through technical statistics. Despite some poor performance in partial tasks, the average challenge score is expected to be around 18.36 out of 20 points, even with zero partial grades.

It is important to note that the performance of each partial task directly affected the resolution of the security challenge, which required students to uncover and analyze their personal information within their cookies. This underscores the complex relationship between technical proficiency, cybersecurity awareness, and practical application in real-world scenarios, providing valuable insights for future educational and security initiatives.

Chapter 5

How to use the BOOKIE Framework

Índice

| | | |
|-------|--|-----|
| 5.1 | Initial considerations | 142 |
| 5.2 | Steps to use BOOKIE Framework | 143 |
| 5.2.1 | Introduction | 143 |
| 5.2.2 | Framework Advantages | 143 |
| 5.2.3 | How to implement the framework through the graphical interface . . | 143 |

5.1. Initial considerations

To utilize the BOOKIE framework, interested parties must consider its practical application. The framework, derived from the analysis of historical and cookie data from university students, can be extended to other areas of study such as school students, administrative staff, office workers, etc. The goal is to make the framework adaptable and applicable to a variety of fields.

The practical application of the BOOKIE framework is showcased through a basic graphical interface created in Python. This interface encompasses all stages of the framework, allowing individuals with basic security knowledge to utilize the tool to exhibit our proposition: The filtering of personal data through the exploitation of XSS attacks to steal cookies.

This section will enable to employ this interface to replicate all the elements of the proposed framework in a real-world scenario.

5.2. Steps to use BOOKIE Framework

5.2.1. Introduction

A series of steps has been proposed to establish the path that allows each of the phases to be applied practically. For example, a teacher could use the framework's management and application to teach computer security topics to his students.

5.2.2. Framework Advantages

As it is a framework that initially analyzes data from web histories and university students' cookies, its application is general. The advantage of the framework is that the data entry (web histories and cookies) will depend on the target audience that will be evaluated.

Applying the framework will also provide an interactive and practical learning environment. However, it is unconditionally required that its use and implementation be by people with basic knowledge of programming, computer security, and database management (not mandatory).

5.2.3. How to implement the framework through the graphical interface

As described in the previous chapter, the development of the framework was a collaborative effort, to establish the 4 main phases: Characterization, Replication and Collection, Training and Prediction and, finally, the Visualization phase.

Our interface was designed to replicate the activities proposed within the BOOKIE framework. Figures 5.1 to 5.8 display the key screens of our graphical interface. The interface was designed for analyzing and evaluating the data of any target audience, making the framework applicable to a wide range of applications.



Figure 5.1: Preview of our graphical interface to apply the BOOKIE framework in real scenarios

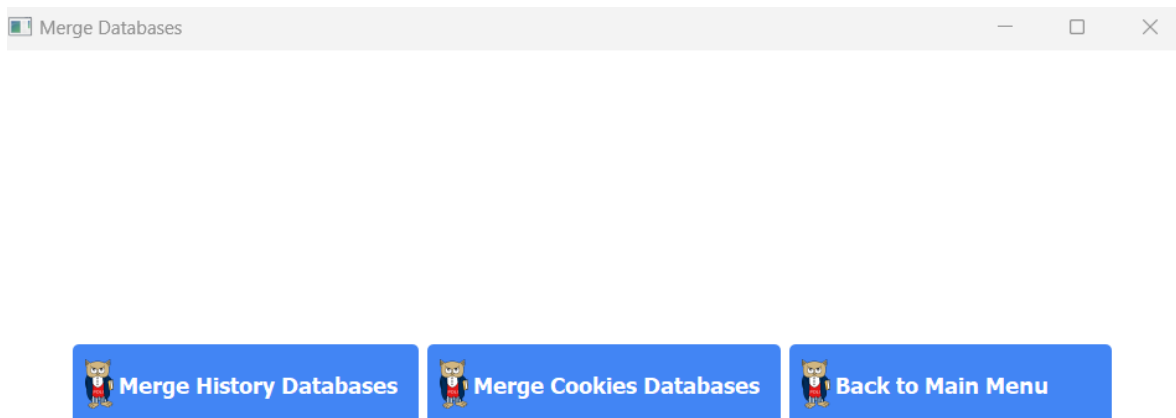


Figure 5.2: Options menu to join browsing history and cookie databases

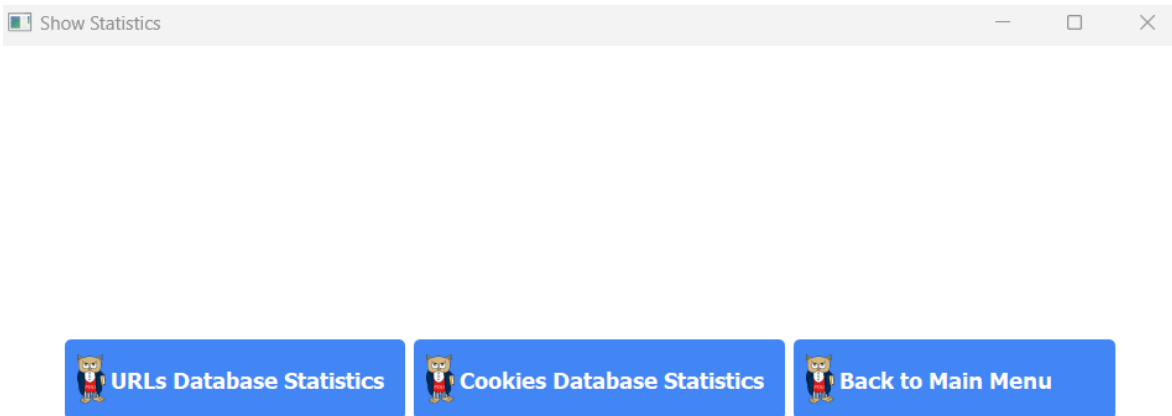


Figure 5.3: Menú de Opciones para mostrar las estadísticas de las bases de historiales de navegación y cookies

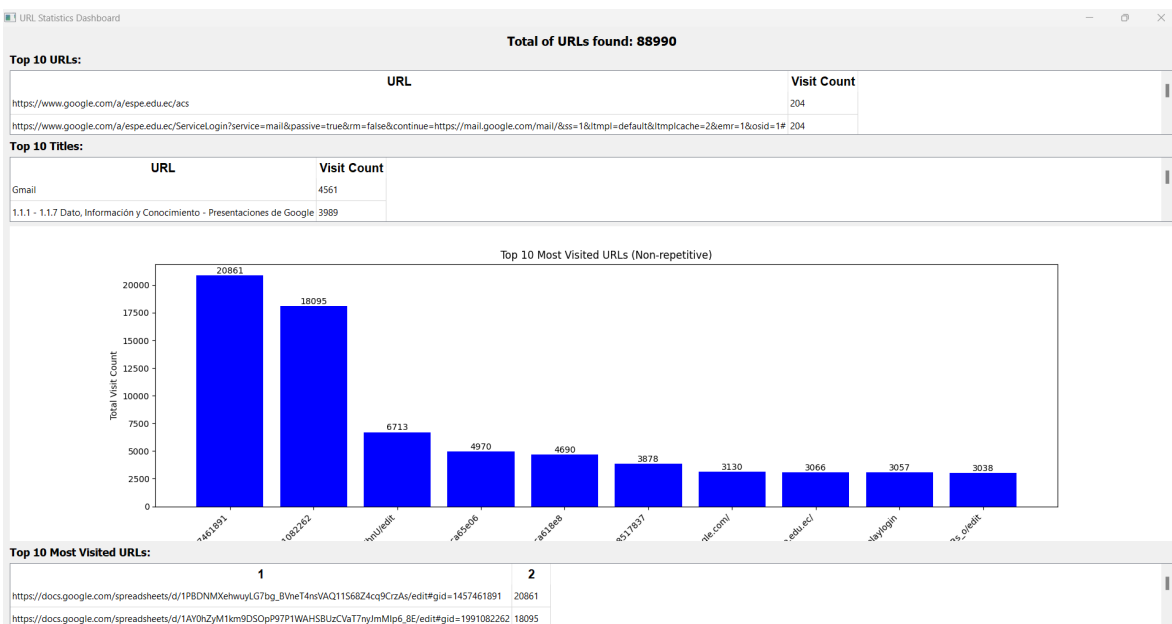


Figure 5.4: Options Menu to display statistics from browsing history and cookie databases

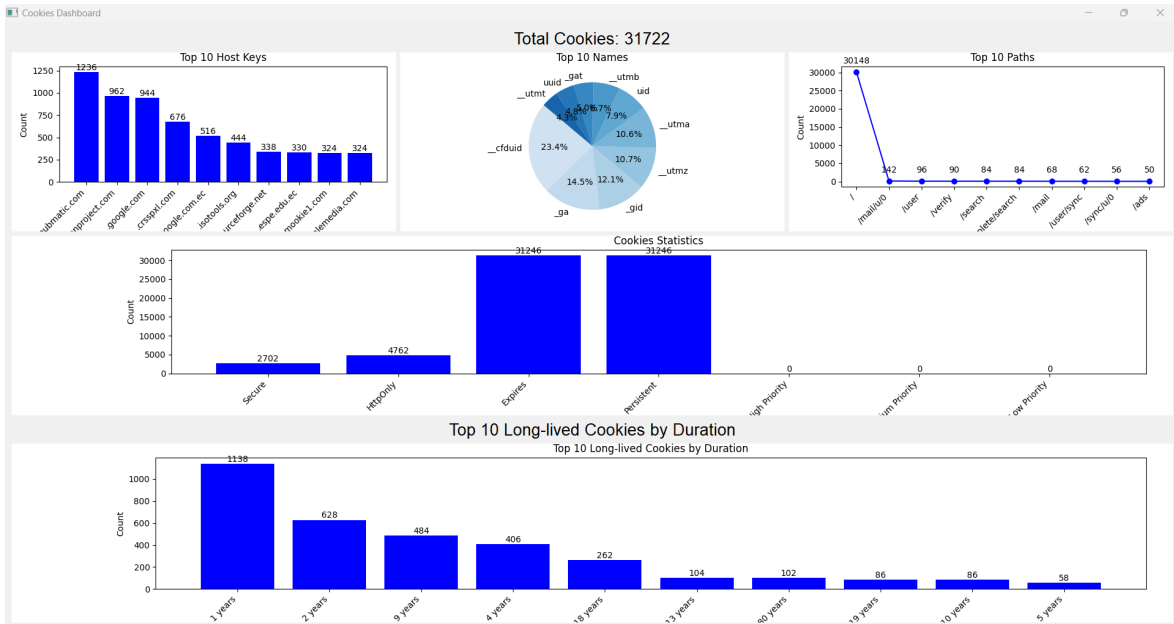


Figure 5.5: Preview of the statistics generated by the cookie databases



Figure 5.6: Preview of web browsing domain statistics before-after being normalized

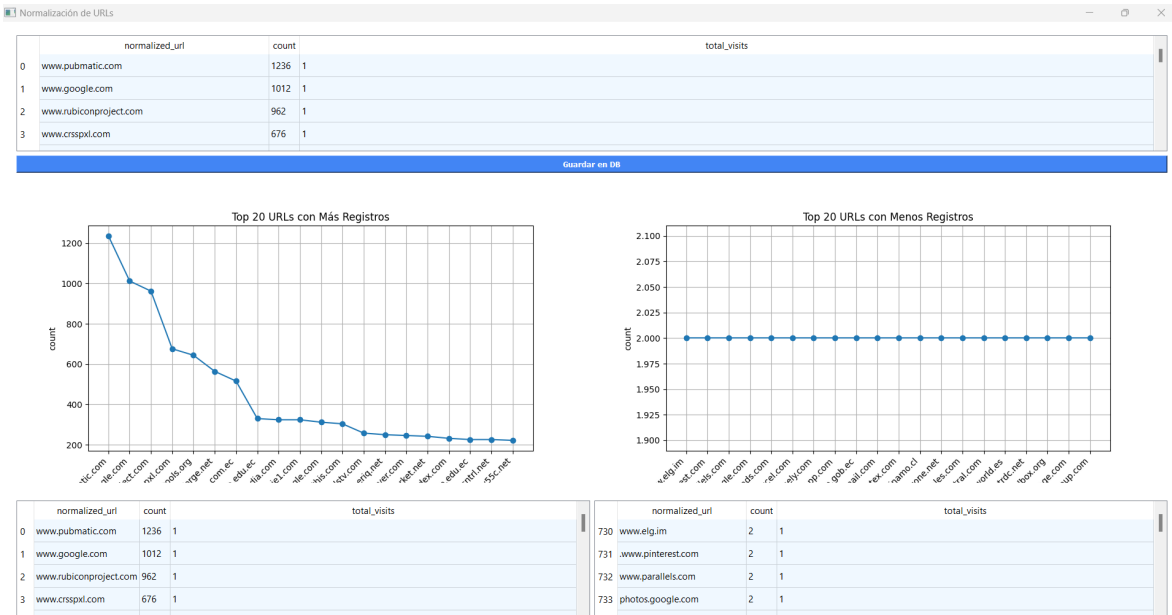


Figure 5.7: Preview of cookie domain statistics before-after being normalized

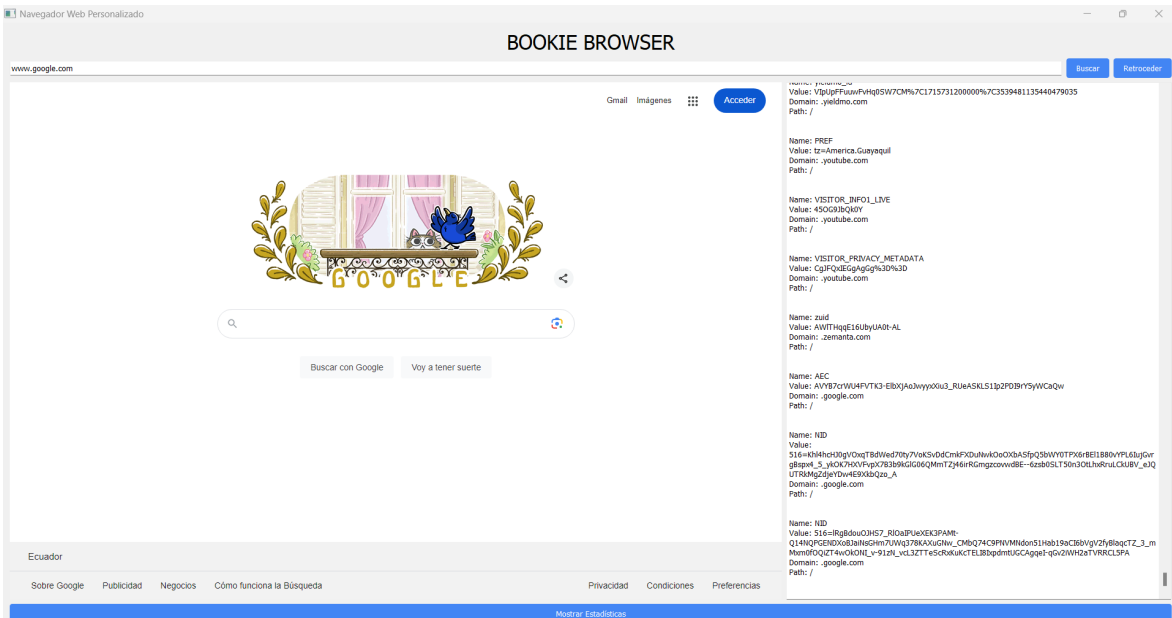


Figure 5.8: BOOKIE browser preview to demonstrate data filtering through cookies

Chapter 6

Conclusions and perspectives

Índice

| | | |
|-------|-----------------------|-----|
| 6.0.1 | Conclusions | 148 |
| 6.0.2 | Future Work | 151 |

6.0.1. Conclusions

- Based on the analysis carried out between 2013 and 2018, it is significantly concluded that using specific tools for web browsers (16.98%) is a notable trend for detecting and combating XSS attacks. Another relevant trend is the analysis of the content of web pages (13.20%), involving examining the code and elements on the pages to identify vulnerabilities. The use of AI methods to classify web pages and mitigate XSS attacks has also been explored to a lesser extent (9.43%). A few strategies have been proposed to detect and mitigate XSS attacks based on the analysis of articles between 2018 and 2023. Practical tools for addressing XSS attacks include multi-layer perceptron, logistic regression, and support vector machines. Python, JAVA, and PHP programming languages are widely used to develop security solutions, allowing the implementation of filters and validations to protect web applications against XSS attacks. Mozilla Firefox and Chrome have been the subject of specific proposals to improve security against XSS through extensions and custom settings. Ad blockers, web application filters, and intrusion detection and prevention systems have also been employed to mitigate the risks associated with XSS attacks. Lastly, tools like Selenium and BeautifulSoup are used to securely extract data from websites, although they are not explicitly designed to prevent XSS attacks.

- Taking as a reference the objective to determine the level of vulnerability of cookies against XSS attacks, the CookieScout phase classifies and qualifies cookies to give them value according to their reputation. Several processes form it; the most important is making previous and subsequent comparisons using a *Knowledge Base*. The analytical model is based on a case study generated by an attack in controlled scenarios. The obtained results evaluate the traffic characteristics the victims exchange with their attackers, the ports used, and the properties of the cookies created when a satisfactory XSS attack is executed. The results show that 88.24% of cookies created have the property for executing commands, and only 11.76% have the *HttpOnly* property, which is a feature of browsers to prevent XSS attacks. As a complement, 29.41% of dangerous websites create cookies with an expiration date of more than two years.
- Based on establishing the relationship between browsing history and cookie attributes to develop a decision tree-based cookie classifier, the DataCookie model was structured. An essential phase allowed the design of the data set with information on the cookies generated based on the users' browsing history. With this dataset, the Weka tool was used to apply an algorithm based on decision trees to obtain rules that allow new cookies to be classified as suspicious. With the rules obtained from the J48 tree decision algorithm, a script was developed in Python to classify a new cookie registry and register the domain of those classified as suspicious.
- We implemented a personal blog with false notes about information theft through WhatsApp status to test the difficulty students face in identifying cookies that transmit personal information. Through JavaScript codes, we created a test cookie, stole its content, and collected information on country of origin, type of operating system, type of browser, and date/time of visit. 95.7% of contacts were accessed through the Facebook post, 1.6% used a QR Reader application to access our WhatsApp status, and 2.7% reviewed our WhatsApp status with the URL link to the blog. It was also found that the Blogger application has no control over the execution of JavaScript codes, making it vulnerable to XSS attacks.
- After an exhaustive analysis of RPA tools for programming and automating tasks using Python scripts, it is concluded that initially evaluated tools could only access first-party cookie data. To overcome this limitation, mouse movements and keyboard text entry were simulated, imitating a user's behavior during web browsing. This technique controlled browsers' opening, searching, and closing, ensuring the automatic genera-

tion of all necessary cookies (first person, third person, and session). In this way, our bot system could directly access the cookie records of the computer on which they were programmed, significantly optimizing data capture and the functionality of the automated system.

- In conclusion, the implementation of clustering algorithms has proven to be an effective strategy to improve several key aspects of the users' browsing experience. The segmentation of users into clusters based on cookies has made it possible to identify similar behavior patterns and interests, which in turn has made it possible to deliver personalized and relevant recommendations for each group. Moreover, this technique has played a crucial role in our security measures, detecting anomalies and security threats by identifying cookies that do not follow normal patterns, thereby ensuring a safe browsing experience. One of the most notable achievements has been continuously optimizing the user experience through more precise and adaptive recommendations, contributing to more efficient and satisfactory navigation, especially in educational environments such as the one studied. In summary, applying these algorithms has significantly strengthened the browser's ability to offer a personalized and secure experience, marking an essential advance in web technology.
- Our study showed that students successfully decrypted cookies without compromising privacy or sharing findings with external parties. However, finding the decryption key on the same computer raised concerns about web browser vulnerability. It emphasized the need to regularly clear cookies to mitigate potential risks. We also evaluated students' knowledge acquisition in cookie decryption through technical statistics. Despite some poor performance in partial tasks, the average challenge score is expected to be around 18.36 out of 20 points. This highlights the complex relationship between technical proficiency, cybersecurity awareness, and practical application in real-world scenarios, providing valuable insights for future initiatives.
- Developing a framework to teach about the leakage of personal data through the theft of cookies via XSS attacks has proven to be an effective tool for university students; however, in its most general form, it could be applied for analysis with a broader audience. The graphical interface designed with Python and PyQt makes it easy to enter and analyze data from anyone's browsing history and cookies, providing an accessible platform for any user. Through the implementation of own web browser, it has been possible to visually obtain cookies from various domains, highlighting that the domain

www.google.com can display around 480 cookies in a single query. This finding underscores the magnitude of the problem and the need for awareness about personal data security.

6.0.2. Future Work

- Building on the trend of using specific tools for web browsers (16.98%), future work could focus on creating more sophisticated and user-friendly browser extensions or plugins for detecting and preventing XSS attacks. These tools could incorporate real-time scanning and alert systems, leveraging machine learning models to improve detection accuracy.
- Given the significance of analyzing web page content (13.20%), research could be directed towards developing automated content analysis frameworks. These frameworks would use advanced code analysis techniques and pattern recognition algorithms to identify potential XSS vulnerabilities within the HTML, JavaScript, and other web elements.
- Expanding on the use of AI methods (9.43%), future studies could explore the integration of deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for classifying web pages and detecting XSS threats. Additionally, AI could be employed to automatically generate and apply security patches.
- Practical tools like multilayer perceptron, logistic regression, and support vector machines have shown promise in recent strategies. Future work could involve optimizing these models specifically for XSS detection, exploring their potential for real-time application in web security systems.
- Since Python, JAVA, and PHP are widely used for developing security solutions, research could focus on creating comprehensive libraries and frameworks in these languages that provide built-in XSS protection. These could include standardized filters, validation functions, and best practice guidelines for developers.
- Future proposals could aim to enhance the security of popular browsers like Mozilla Firefox and Chrome through advanced extensions and custom settings. This could involve collaborating with browser developers to integrate these enhancements directly into the browser's core security features.

- While tools like Selenium and BeautifulSoup are not explicitly designed to prevent XSS attacks, future work could involve modifying these tools to include security checks during data extraction. This could help prevent the inadvertent execution of malicious scripts during web scraping activities.
- Future research could create more complex and realistic case studies by simulating a variety of controlled attack scenarios. This would provide a broader understanding of how different types of XSS attacks affect cookie properties and user data, helping to identify new patterns and vulnerabilities.
- Given that 29.41% of dangerous websites create cookies with an expiration date of more than two years, future studies could examine the long-term impact of these cookies on user security and privacy. This includes analyzing how such cookies can be exploited over time and developing strategies to mitigate these risks.
- Developing educational programs and resources to raise awareness among users about the risks associated with cookies and XSS attacks. This includes creating guides on how to configure browser settings for maximum security and recognizing the signs of a potential XSS attack.
- Building on the evaluation of students' knowledge acquisition, future initiatives could aim to enhance cybersecurity education programs. This could include developing more comprehensive curricula that cover a wider range of topics, such as secure coding practices, encryption techniques, and privacy protection measures.
- Incorporating more practical applications and real-world scenarios into cybersecurity curricula. This could include case studies, hands-on labs, and collaborative projects that simulate actual security challenges students might face in their professional lives.
- Extend the graphical interface to include educational modules on other types of cyber attacks, such as phishing and malware, providing a more complete view of cybersecurity. Improve the graphical interface to make it more intuitive and accessible, including customization options for different levels of technical knowledge. Create simulation modules that allow users to safely experience different attack and defense scenarios, reinforcing practical learning.

Chapter 7

References

- [1] V. Arya, D. Sethi, and J. Paul, “Does digital footprint act as a digital asset?—enhancing brand experience through remarketing,” *International Journal of Information Management*, vol. 49, pp. 142–156, 2019.
- [2] K. Varnali, “Online behavioral advertising: An integrative review,” *Journal of Marketing Communications*, vol. 27, no. 1, pp. 93–114, 2021.
- [3] *Phishing Attack - What is it and How Does it Work? - Check Point Software — checkpoint.com*, <https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-phishing/>.
- [4] G. Franken, T. Van Goethem, and W. Joosen, “Who left open the cookie jar? a comprehensive evaluation of {third-party} cookie policies,” in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 151–168.
- [5] H. Mukhtar, F. Seemi, H. Aslam, and S. Khattak, “Browsing behaviour analysis using data mining,” *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 2, 2019.
- [6] *WSTG - Latest | OWASP Foundation — owasp.org*, https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.
- [7] H. Kwon, H. Nam, S. Lee, C. Hahn, and J. Hur, “(in-) security of cookies in https: Cookie theft by removing cookie flags,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1204–1215, 2019.

- [8] gn, *Online tracking: A 1-million-site measurement and analysis — webtransparency.cs.princeton.edu*, <https://webtransparency.cs.princeton.edu/webcensus/>.
- [9] *CVE - CVE* — cve.mitre.org, <https://cve.mitre.org/>.
- [10] F. Nosheen and U. Qamar, “Flexibility and privacy control by cookie management,” in *2015 Third International Conference on Digital Information, Networking, and Wireless Communications (DINWC)*, 2015, pp. 94–98. DOI: 10.1109/DINWC.2015.7054224.
- [11] W. F. Martín *et al.*, “Metodología de la investigación,” *Universidad de Cienfuegos. Cienfuegos. Cuba*, 345pp, 2006.
- [12] G. E. Rodríguez, D. E. Benavides, J. Torres, P. Flores, and W. Fuertes, “Cookie scout: An analytic model for prevention of cross-site scripting (xss) using a cookie classifier,” in *Proceedings of the International Conference on Information Technology & Systems (ICITS 2018)*, Á. Rocha and T. Guarda, Eds., Cham: Springer International Publishing, 2018, pp. 497–507, ISBN: 978-3-319-73450-7.
- [13] G. E. Rodríguez, J. G. Torres, and E. Benavides-Astudillo, “Datacookie: Sorting cookies using data mining for prevention of cross-site scripting (xss),” in *Emerging Trends in Cybersecurity Applications*, K. Daimi, A. Alsadoon, C. Peoples, and N. El Madhoun, Eds. Cham: Springer International Publishing, 2023, ISBN: 978-3-031-09640-2. DOI: 10.1007/978-3-031-09640-2_8. [Online]. Available: https://doi.org/10.1007/978-3-031-09640-2_8.
- [14] G. Rodríguez, J. Torres, P. Flores, E. Benavides, and P. Proaño, “Trusted phishing: A model to teach computer security through the theft of cookies,” in *Advances in Emerging Trends and Technologies*, M. Botto-Tobar, J. León-Acurio, A. Díaz Cadena, and P. Montiel Díaz, Eds., Cham: Springer International Publishing, 2020, pp. 390–401, ISBN: 978-3-030-32033-1.
- [15] G. Rodriguez, J. Torres, P. Flores, E. Benavides, and D. Nuñez-Agurto, “Xsstudent: Proposal to avoid cross-site scripting (xss) attacks in universities,” in *2019 3rd Cyber Security in Networking Conference (CSNet)*, 2019, pp. 142–149. DOI: 10.1109/CSNet47905.2019.9108965.
- [16] G. E. Rodríguez, J. Torres, and E. Benavides, “Xss2dent, detecting cross-site scripting attacks (xss) vulnerabilities: A case study,” in *Applied Technologies*, M. Botto-Tobar, S. Montes León, P. Torres-Carrión, M. Zambrano Vizuetete, and B. Durakovic,

- Eds., Cham: Springer International Publishing, 2022, pp. 365–380, ISBN: 978-3-031-03884-6.
- [17] T.-O. J. Rodríguez-Galán Germán and C.-M. Luis, “Hackmyself: Decrypting cookies to show the theft of personal data in university students,” in *Lecture Notes in Networks and Systems*, Springer Nature Switzerland, 2024. DOI: Acceptedforpublication.
- [18] P. Joshi and C. -. J. Kuo, “Security and privacy in online social networks: A survey,” in *2011 IEEE International Conference on Multimedia and Expo*, 2011, pp. 1–6. DOI: 10.1109/ICME.2011.6012166.
- [19] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, “Cross-site scripting (xss) attacks and mitigation: A survey,” *Computer Networks*, vol. 166, p. 106960, 2020, ISSN: 1389-1286. DOI: <https://doi.org/10.1016/j.comnet.2019.106960>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128619311247>.
- [20] G. Rodríguez-Galán and J. Torres, “Personal data filtering: A systematic literature review comparing the effectiveness of xss attacks in web applications vs cookie stealing,” *Annals of Telecommunications*, Apr. 2024, ISSN: 1958-9395. DOI: 10.1007/s12243-024-01022-8. [Online]. Available: <https://doi.org/10.1007/s12243-024-01022-8>.
- [21] J. Villares-Jimenez, J. Quinatoa-Medina, and G. Rodríguez-Galán, “Diseño e implementación de bots para automatizar tareas de búsqueda y análisis de vulnerabilidades en sistemas web,” Universidad de las Fuerzas Armadas ESPE, Tech. Rep., 2022.
- [22] J. Villares-Jimenez, J. Quinatoa-Medina, G. Rodríguez-Galán, D. Nuñez-Agurto, and B. Santillán-Tituaña, “Vulnerability of captcha systems using bots with computer vision abilities,” in *Applied Technologies*, M. Botto-Tobar, M. Zambrano Vizuete, S. Montes León, P. Torres-Carrión, and B. Durakovic, Eds., Cham: Springer Nature Switzerland, 2023, pp. 329–343, ISBN: 978-3-031-24985-3.
- [23] S. L. Velagapudi and H. Gupta, “Privacy, security of cookies in http transmission,” in *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, 2019, pp. 22–25. DOI: 10.1109/ISCON47742.2019.9036289.
- [24] A. M. Sathiyaseelan, V. Joseph, and A. Srinivasaraghavan, “A proposed system for preventing session hijacking with modified one-time cookies,” in *2017 International*

- Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, 2017, pp. 451–454. DOI: 10.1109/ICBDACI.2017.8070882.
- [25] A. Aladeokin, P. Zavorsky, and N. Memon, “Analysis and compliance evaluation of cookies-setting websites with privacy protection laws,” in *2017 Twelfth International Conference on Digital Information Management (ICDIM)*, 2017, pp. 121–126. DOI: 10.1109/ICDIM.2017.8244646.
- [26] T. Sakamoto and M. Matsunaga, “After gdpr, still tracking or not? understanding opt-out states for online behavioral advertising,” in *2019 IEEE Security and Privacy Workshops (SPW)*, 2019, pp. 92–99. DOI: 10.1109/SPW.2019.00027.
- [27] B. Li, S.-j. Lv, Y.-s. Zhang, and M. Tian, “The application research of cookies in network security,” in *PROCEEDINGS OF 2013 International Conference on Sensor Network Security Technology and Privacy Communication System*, 2013, pp. 152–155. DOI: 10.1109/SNS-PCS.2013.6553855.
- [28] L. Nazaryan, R. Jin, C. Yue, *et al.*, “Securely outsourcing cookies to the cloud via private information retrieval,” in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2016, pp. 1–8. DOI: 10.1109/WiMOB.2016.7763250.
- [29] R. Gonzalez, L. Jiang, M. Ahmed, *et al.*, “The cookie recipe: Untangling the use of cookies in the wild,” in *2017 Network Traffic Measurement and Analysis Conference (TMA)*, 2017, pp. 1–9. DOI: 10.23919/TMA.2017.8002896.
- [30] O. Kulyk, P. Mayer, M. Volkamer, and O. Käfer, “A concept and evaluation of usable and fine-grained privacy-friendly cookie settings interface,” in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 1058–1063. DOI: 10.1109/TrustCom/BigDataSE.2018.00148.
- [31] Q. Chen, P. Ilija, M. Polychronakis, and A. Kapravelos, “Cookie swap party: Abusing first-party cookies for web tracking,” in *Proceedings of the Web Conference 2021*, ser. WWW '21, Ljubljana, Slovenia: Association for Computing Machinery, 2021, pp. 2117–2129, ISBN: 9781450383127. DOI: 10.1145/3442381.3449837. [Online]. Available: <https://doi.org/10.1145/3442381.3449837>.

- [32] R. Gomer, E. M. Rodrigues, N. Milic-Frayling, and M. Schraefel, "Network analysis of third party tracking: User exposure to tracking cookies through search," in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 1, 2013, pp. 549–556. DOI: 10.1109/WI-IAT.2013.77.
- [33] E. Shuford, T. Kavanaugh, B. Ralph, E. Ceesay, and P. Watters, "Measuring personal privacy breaches using third-party trackers," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (Trust-Com/BigDataSE)*, 2018, pp. 1615–1618. DOI: 10.1109/TrustCom/BigDataSE.2018.00236.
- [34] A. Herps, P. A. Watters, and G. Pineda-Villavicencio, "Measuring surveillance in online advertising: A big data approach," in *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, 2013, pp. 30–35. DOI: 10.1109/CTC.2013.12.
- [35] J. R. Annam, P. K. Ande, B. Kanuri, C. Prasad, B. S. Babu, and P. Tatineni, "User valuation of secrecy framing based on general data protection regulation (gdpr) users," in *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2021, pp. 1215–1219. DOI: 10.1109/ICIRCA51532.2021.9544896.
- [36] S. Thapar, N. Srivastava, A. Girdhar, and A. Bhat, "Host based detection and analysis of pii stealing trackers," in *2016 International Conference on Computing, Communication and Automation (ICCCA)*, 2016, pp. 575–578. DOI: 10.1109/CCAA.2016.7813786.
- [37] S. J. Murdoch, "Hardened stateless session cookies," in *Security Protocols XVI*, B. Christianson, J. A. Malcolm, V. Matyas, and M. Roe, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 93–101.
- [38] M. K. Gupta, M. C. Govil, and G. Singh, "Predicting cross-site scripting (xss) security vulnerabilities in web applications," in *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Jul. 2015, pp. 162–167.
- [39] Verizon. "2017 data breach investigations report." (Apr. 2018), [Online]. Available: <http://www.ictsecuritymagazine.com/wp-content/uploads/2017-Data-Breach-Investigations-Report.pdf>.

- [40] OWASP. "Owasp top 10 application security risks - 2017." (Apr. 2018), [Online]. Available: https://www.owasp.org/index.php/Top_10-2017_Top_10.
- [41] *Top 25 Software Errors | SANS Institute — sans.org*, <https://www.sans.org/top25-software-errors/>, [Accessed 14-05-2024].
- [42] CWE. "Cwe-79: Improper neutralization of input during web page generation ('cross-site scripting')." (Apr. 2018), [Online]. Available: <http://cwe.mitre.org/top25/index.html#CWE-79>.
- [43] Y. F. G. M. Elhakeem and B. I. A. Barry, "Developing a security model to protect websites from cross-site scripting attacks using zend framework application," in *2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE)*, Aug. 2013, pp. 624–629.
- [44] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic, "Noxes: A client-side solution for mitigating cross-site scripting attacks," in *Proceedings of the 2006 ACM symposium on Applied computing*, ACM, 2006, pp. 330–337.
- [45] G. E. Rodríguez, D. E. Benavides, J. Torres, P. Flores, and W. Fuertes, "Cookie scout: An analytic model for prevention of cross-site scripting (xss) using a cookie classifier," in *Proceedings of the International Conference on Information Technology & Systems (ICITS 2018)*, Á. Rocha and T. Guarda, Eds., Cham: Springer International Publishing, 2018, pp. 497–507.
- [46] H. Takahashi, K. Yasunaga, M. Mambo, K. Kim, and H. Y. Youm, "Preventing abuse of cookies stolen by xss," in *2013 Eighth Asia Joint Conference on Information Security*, 2013, pp. 85–89. DOI: 10.1109/ASIAJCIS.2013.20.
- [47] O. J. Falana, I. O. Ebo, C. O. Tinubu, O. A. Adejimi, and A. Ntuk, "Detection of cross-site scripting attacks using dynamic analysis and fuzzy inference system," in *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, 2020, pp. 1–6. DOI: 10.1109/ICMCECS47690.2020.240871.
- [48] WhiteHatSecurity. "2017 application security statistics report." (Apr. 2018), [Online]. Available: <https://www.whitehatsec.com/resources-category/premium-content/web-application-stats-report-2017/>.
- [49] O. B. Al-Khurafi and M. A. Al-Ahmad, "Survey of web application vulnerability attacks," in *2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, Dec. 2015, pp. 154–158. DOI: 10.1109/ACSAT.2015.46.

- [50] G. Shanmugasundaram, S. Ravivarman, and P. Thangavellu, "A study on removal techniques of cross-site scripting from web applications," in *2015 International Conference on Computation of Power, Energy, Information and Communication (IC-CPEIC)*, Apr. 2015, pp. 0436–0442. DOI: 10.1109/ICCPEIC.2015.7259498.
- [51] G. P. Bherde and M. A. Pund, "Recent attack prevention techniques in web service applications," in *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, Sep. 2016, pp. 1174–1180. DOI: 10.1109/ICACDOT.2016.7877771.
- [52] A. Shrivastava, S. Choudhary, and A. Kumar, "Xss vulnerability assessment and prevention in web application," in *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, Oct. 2016, pp. 850–853. DOI: 10.1109/NGCT.2016.7877529.
- [53] M. K. Gupta, M. C. Govil, and G. Singh, "Static analysis approaches to detect sql injection and cross site scripting vulnerabilities in web applications: A survey," in *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, May 2014, pp. 1–5. DOI: 10.1109/ICRAIE.2014.6909173.
- [54] M. K. Gupta, M. C. Govil, and G. Singh, "Static analysis approaches to detect sql injection and cross site scripting vulnerabilities in web applications: A survey," in *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, May 2014, pp. 1–5.
- [55] V. Nithya, S. L. Pandian, and C. Malarvizhi, "A survey on detection and prevention of cross-site scripting attack," *International Journal of Security and Its Applications*, vol. 9, no. 3, pp. 139–52, 2015.
- [56] R. Wang, G. Xu, X. Zeng, X. Li, and Z. Feng, "Tt-xss: A novel taint tracking based dynamic detection framework for dom cross-site scripting," *Journal of Parallel and Distributed Computing*, vol. 118, pp. 100–106, 2018, ISSN: 0743-7315. DOI: <https://doi.org/10.1016/j.jpdc.2017.07.006>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0743731517302186>.
- [57] L. K. Shar and H. B. K. Tan, "Auditing the xss defence features implemented in web application programs," *IET Software*, vol. 6, no. 4, pp. 377–390, Aug. 2012.

- [58] T. S. Mehta and S. Jamwal, "Model to prevent websites from xss vulnerabilities," *IJCSIT) International Journal of Computer Science and Information Technologies*, vol. 6, no. 2, pp. 1059–1067, 2015.
- [59] M. Mohammadi, B. Chu, H. R. Lipford, and E. Murphy-Hill, "Automatic web security unit testing: Xss vulnerability detection," in *2016 IEEE/ACM 11th International Workshop in Automation of Software Test (AST)*, May 2016, pp. 78–84.
- [60] M. Parvez, P. Zavorsky, and N. Khoury, "Analysis of effectiveness of black-box web application scanners in detection of stored sql injection and stored xss vulnerabilities," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec. 2015, pp. 186–191.
- [61] Y. Liu, W. Zhao, D. Wang, and L. Fu, "A xss vulnerability detection approach based on simulating browser behavior," in *2015 2nd International Conference on Information Science and Security (ICISS)*, Dec. 2015, pp. 1–4.
- [62] X. Guo, S. Jin, and Y. Zhang, "Xss vulnerability detection using optimized attack vector repertory," in *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Sep. 2015, pp. 29–36.
- [63] F. Duchene, R. Groz, S. Rawat, and J. L. Richier, "Xss vulnerability detection using model inference assisted evolutionary fuzzing," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, Apr. 2012, pp. 815–817.
- [64] M. E. Ruse and S. Basu, "Detecting cross-site scripting vulnerability using concolic testing," in *2013 10th International Conference on Information Technology: New Generations*, Apr. 2013, pp. 633–638.
- [65] G. Dong, Y. Zhang, X. Wang, P. Wang, and L. Liu, "Detecting cross site scripting vulnerabilities introduced by html5," in *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, May 2014, pp. 319–323.
- [66] T. K. Nguyen and S. O. Hwang, "Large-scale detection of dom-based xss based on publisher and subscriber model," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2016, pp. 975–980.
- [67] J. Pan and X. Mao, "Domxssmicro: A micro benchmark for evaluating dom-based cross-site scripting detection," in *2016 IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 208–215.

- [68] M. K. Gupta, M. C. Govil, and G. Singh, "Text-mining based predictive model to detect xss vulnerable files in web applications," in *2015 Annual IEEE India Conference (INDICON)*, Dec. 2015, pp. 1–6.
- [69] M. K. Gupta, M. C. Govil, G. Singh, and P. Sharma, "Xssdm: Towards detection and mitigation of cross-site scripting vulnerabilities in web applications," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Aug. 2015, pp. 2010–2015.
- [70] M. K. Gupta, M. C. Govil, and G. Singh, "A context-sensitive approach for precise detection of cross-site scripting vulnerabilities," in *2014 10th International Conference on Innovations in Information Technology (IIT)*, Nov. 2014, pp. 7–12.
- [71] H. Shahriar, S. North, W.-C. Chen, and E. Mawangi, "Design and development of anti-xss proxy," in *8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*, Dec. 2013, pp. 484–489.
- [72] D. Guamán, F. Guamán, D. Jaramillo, and M. Sucunuta, "Implementation of techniques and owasp security recommendations to avoid sql and xss attacks using j2ee and ws-security," in *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, Jun. 2017, pp. 1–7.
- [73] H. Takahashi, K. Yasunaga, M. Mambo, K. Kim, and H. Y. Youm, "Preventing abuse of cookies stolen by xss," in *2013 Eighth Asia Joint Conference on Information Security*, Jul. 2013, pp. 85–89.
- [74] S. al Azmi and A. R. Khan, "A comprehensive research on xss scripting attacks on different domains and their verticals," in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 01, Dec. 2015, pp. 677–680.
- [75] C. M. Frenz and J. P. Yoon, "Xssmon: A perl based ids for the detection of potential xss attacks," in *2012 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, May 2012, pp. 1–4.
- [76] I. Yusof and A. S. K. Pathan, "Preventing persistent cross-site scripting (xss) attack by applying pattern filtering approach," in *The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, Nov. 2014, pp. 1–6.

- [77] J. Bozic and F. Wotawa, "Xss pattern for attack modeling in testing," in *2013 8th International Workshop on Automation of Software Test (AST)*, May 2013, pp. 71–74.
- [78] C. H. Wang and Y. S. Zhou, "A new cross-site scripting detection mechanism integrated with html5 and cors properties by using browser extensions," in *2016 International Computer Symposium (ICS)*, Dec. 2016, pp. 264–269.
- [79] R. Wang, X. Jia, Q. Li, and D. Zhang, "Improved n-gram approach for cross-site scripting detection in online social network," in *2015 Science and Information Conference (SAI)*, Jul. 2015, pp. 1206–1212.
- [80] R. Wang, X. Jia, Q. Li, and S. Zhang, "Machine learning based cross-site scripting detection in online social network," in *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS)*, Aug. 2014, pp. 823–826.
- [81] M. R. Zalbina, T. W. Septian, D. Stiawan, M. Y. Idris, A. Heryanto, and R. Budiarto, "Payload recognition and detection of cross site scripting attack," in *2017 2nd International Conference on Anti-Cyber Crimes (ICACC)*, Mar. 2017, pp. 172–176.
- [82] G. Wassermann and Z. Su, "Static detection of cross-site scripting vulnerabilities," in *2008 ACM/IEEE 30th International Conference on Software Engineering*, May 2008, pp. 171–180.
- [83] A. Stasinopoulos, C. Ntantogian, and C. Xenakis, "Bypassing xss auditor: Taking advantage of badly written php code," in *2014 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Dec. 2014, pp. 000 290–000 295.
- [84] T. S. Rocha and E. Souto, "Etssdetector: A tool to automatically detect cross-site scripting vulnerabilities," in *2014 IEEE 13th International Symposium on Network Computing and Applications*, Aug. 2014, pp. 306–309.
- [85] M. K. Gupta, M. C. Govil, and G. Singh, "Predicting cross-site scripting (xss) security vulnerabilities in web applications," in *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Jul. 2015, pp. 162–167.

- [86] B. Panja, T. Gennarelli, and P. Meharia, "Handling cross site scripting attacks using cache check to reduce webpage rendering time with elimination of sanitization and filtering in light weight mobile web browser," in *2015 First Conference on Mobile and Secure Services (MOBISSECSERV)*, Feb. 2015, pp. 1–7.
- [87] L. K. Shar and H. B. K. Tan, "Auditing the xss defence features implemented in web application programs," *IET Software*, vol. 6, no. 4, pp. 377–390, Aug. 2012.
- [88] K. S. Rao, N. Jain, N. Limaje, A. Gupta, M. Jain, and B. Menezes, "Two for the price of one: A combined browser defense against xss and clickjacking," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2016, pp. 1–6.
- [89] B. Mewara, S. Bairwa, J. Gajrani, and V. Jain, "Enhanced browser defense for reflected cross-site scripting," in *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization*, Oct. 2014, pp. 1–6.
- [90] B. Rexha, A. Halili, K. Rrmoku, and D. Imeraj, "Impact of secure programming on web application vulnerabilities," in *2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*, Nov. 2015, pp. 61–66.
- [91] R. Johari and P. Sharma, "A survey on web application vulnerabilities (sqlia, xss) exploitation and security engine for sql injection," in *2012 International Conference on Communication Systems and Network Technologies*, May 2012, pp. 453–458.
- [92] J. Shanmugam and M. Ponnaivaikko, "Xss application worms: New internet infestation and optimized protective measures," in *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, vol. 3, Jul. 2007, pp. 1164–1169.
- [93] H. Zeng, "Research on developing an attack and defense lab environment for cross site scripting education in higher vocational colleges," in *2013 International Conference on Computational and Information Sciences*, Jun. 2013, pp. 1971–1974.
- [94] J. Bozic and F. Wotawa, "Purity: A planning-based security testing tool," in *2015 IEEE International Conference on Software Quality, Reliability and Security - Companion*, Aug. 2015, pp. 46–55.
- [95] J. You and F. Guo, "Improved csrfguard for csrf attacks defense on java ee platform," in *2014 9th International Conference on Computer Science Education*, Aug. 2014, pp. 1115–1120.

- [96] S. Ding, H. B. K. Tan, L. K. Shar, and B. M. Padmanabhuni, "Towards a hybrid framework for detecting input manipulation vulnerabilities," in *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, vol. 1, Dec. 2013, pp. 363–370.
- [97] J. Pan and X. Mao, "Detecting dom-sourced cross-site scripting in browser extensions," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2017, pp. 24–34.
- [98] I. Dolnák, "Content security policy (csp) as countermeasure to cross site scripting (xss) attacks," in *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, Oct. 2017, pp. 1–4.
- [99] D. A. Suju and G. M. Gandhi, "An automaton based approach for forestalling cross site scripting attacks in web application," in *2015 Seventh International Conference on Advanced Computing (ICoAC)*, Dec. 2015, pp. 1–6.
- [100] R. M. Pandurang and D. C. Karia, "Impact analysis of preventing cross site scripting and sql injection attacks on web application," in *2015 IEEE Bombay Section Symposium (IBSS)*, Sep. 2015, pp. 1–5.
- [101] P. A. Sonewar and N. A. Mhetre, "A novel approach for detection of sql injection and cross site scripting attacks," in *2015 International Conference on Pervasive Computing (ICPC)*, Jan. 2015, pp. 1–4.
- [102] D. Lan, W. ShuTing, Y. Xing, and Z. Wei, "Analysis and prevention for cross-site scripting attack based on encoding," in *2013 IEEE 4th International Conference on Electronics Information and Emergency Communication*, Nov. 2013, pp. 102–105.
- [103] Y. F. G. M. Elhakeem and B. I. A. Barry, "Developing a security model to protect websites from cross-site scripting attacks using zend framework application," in *2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE)*, Aug. 2013, pp. 624–629.
- [104] D. Das, U. Sharma, and D. K. Bhattacharyya, "Detection of cross-site scripting attack under multiple scenarios," *The Computer Journal*, vol. 58, no. 4, pp. 808–822, Apr. 2015.
- [105] Y. Sun and D. He, "Model checking for the defense against cross-site scripting attacks," in *2012 International Conference on Computer Science and Service System*, Aug. 2012, pp. 2161–2164.

- [106] K. Gupta, R. R. Singh, and M. Dixit, "Cross site scripting (xss) attack detection using intrusion detection system," in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Jun. 2017, pp. 199–203.
- [107] I. Dacosta, S. Chakradeo, M. Ahamad, and P. Traynor, "One-time cookies: Preventing session hijacking attacks with stateless authentication tokens," *ACM Trans. Internet Technol.*, vol. 12, no. 1, 1:1–1:24, Jul. 2012, ISSN: 1533-5399.
- [108] S. Calzavara, G. Tolomei, M. Bugliesi, and S. Orlando, "Quite a mess in my cookie jar!: Leveraging machine learning to protect web authentication," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14, Seoul, Korea, 2014, pp. 189–200, ISBN: 978-1-4503-2744-2.
- [109] Y. Mundada, N. Feamster, and B. Krishnamurthy, "Half-baked cookies: Hardening cookie-based authentication for the modern web," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '16, Xi'an, China, 2016, pp. 675–685, ISBN: 978-1-4503-4233-9.
- [110] S. Calzavara, G. Tolomei, A. Casini, M. Bugliesi, and S. Orlando, "A supervised learning approach to protect client authentication on the web," *ACM Trans. Web*, vol. 9, no. 3, 15:1–15:30, Jun. 2015, ISSN: 1559-1131.
- [111] S. N. Bukhari, M. Ahmad Dar, and U. Iqbal, "Reducing attack surface corresponding to type 1 cross-site scripting attacks using secure development life cycle practices," in *2018 Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, Feb. 2018, pp. 1–4. DOI: 10.1109/AEEICB.2018.8480945.
- [112] K. Pranathi, S. Kranthi, A. Srisaila, and P. Madhavalatha, "Attacks on web application caused by cross site scripting," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Mar. 2018, pp. 1754–1759. DOI: 10.1109/ICECA.2018.8474765.
- [113] T. A. Taha and M. Karabatak, "A proposed approach for preventing cross-site scripting," in *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, Mar. 2018, pp. 1–4. DOI: 10.1109/ISDFS.2018.8355356.
- [114] A. P. Sivanesan, A. Mathur, and A. Y. Javaid, "A google chromium browser extension for detecting xss attack in html5 based websites," in *2018 IEEE International Con-*

- ference on Electro/Information Technology (EIT)*, May 2018, pp. 0302–0304. DOI: 10.1109/EIT.2018.8500284.
- [115] D. Zubarev and I. Skarga-Bandurova, “Cross-site scripting for graphic data: Vulnerabilities and prevention,” in *2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Jun. 2019, pp. 154–160. DOI: 10.1109/DESSERT.2019.8770043.
- [116] P. Chen, H. Yu, M. Zhao, and J. Wang, “Research and implementation of cross-site scripting defense method based on moving target defense technology,” in *2018 5th International Conference on Systems and Informatics (ICSAI)*, Nov. 2018, pp. 818–822. DOI: 10.1109/ICSAI.2018.8599463.
- [117] X. Hou, X. Zhao, M. Wu, R. Ma, and Y. Chen, “A dynamic detection technique for xss vulnerabilities,” in *2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, Apr. 2018, pp. 34–43. DOI: 10.1109/ICNISC.2018.00016.
- [118] G. Kaur, B. Pande, A. Bhardwaj, G. Bhagat, and S. Gupta, “Defense against html5 xss attack vectors: A nested context-aware sanitization technique,” in *2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, Jan. 2018, pp. 442–446. DOI: 10.1109/CONFLUENCE.2018.8442855.
- [119] A. Algaith, P. Nunes, F. Jose, I. Gashi, and M. Vieira, “Finding sql injection and cross site scripting vulnerabilities with diverse static analysis tools,” in *2018 14th European Dependable Computing Conference (EDCC)*, Sep. 2018, pp. 57–64. DOI: 10.1109/EDCC.2018.00020.
- [120] F. M. M. Mokbal, W. Dan, A. Imran, L. Jiuchuan, F. Akhtar, and W. Xiaoxi, “Mlpxss: An integrated xss-based attack detection scheme in web applications using multilayer perceptron technique,” *IEEE Access*, vol. 7, pp. 100 567–100 580, 2019. DOI: 10.1109/ACCESS.2019.2927417.
- [121] S. Akaishi and R. Uda, “Classification of xss attacks by machine learning with frequency of appearance and co-occurrence,” in *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, Mar. 2019, pp. 1–6. DOI: 10.1109/CISS.2019.8693047.

- [122] S. Gupta and B. B. Gupta, "Js-san: Defense mechanism for html5-based web applications against javascript code injection vulnerabilities," *Security and Communication Networks*, vol. 9, pp. 1477–1495, 2016.
- [123] S. Gupta and B. B. Gupta, "Php-sensor: A prototype method to discover workflow violation and xss vulnerabilities in php web applications," in *Conf. Computing Frontiers*, 2015.
- [124] B. Gupta, S. Gupta, and P. Chaudhary, "Enhancing the browser-side context-aware sanitization of suspicious html5 code for halting the dom-based xss vulnerabilities in cloud," *Int. J. Cloud Appl. Comput.*, vol. 7, no. 1, pp. 1–31, Jan. 2017, ISSN: 2156-1834. DOI: 10.4018/IJCAC.2017010101. [Online]. Available: <https://doi.org/10.4018/IJCAC.2017010101>.
- [125] "Cross-site scripting (xss) attacks and mitigation: A survey," *Computer Networks*, vol. 166, p. 106960, 2020. DOI: <https://doi.org/10.1016/j.comnet.2019.106960>.
- [126] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," ser. EASE '14, London, England, United Kingdom: Association for Computing Machinery, 2014, ISBN: 9781450324762. DOI: 10.1145/2601248.2601268. [Online]. Available: <https://doi.org/10.1145/2601248.2601268>.
- [127] R. Rabbit. "Researchrabbit." (2023), [Online]. Available: <https://www.researchrabbit.ai/>.
- [128] S. Gupta, B. B. Gupta, and P. Chaudhary, "Hunting for dom-based xss vulnerabilities in mobile cloud-based online social network," *Future Generation Computer Systems*, vol. 79, Jun. 2017. DOI: 10.1016/j.future.2017.05.038.
- [129] T. Bui, S. Rao, M. Antikainen, and T. Aura, "Xss vulnerabilities in cloud-application add-ons," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '20, Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 610–621, ISBN: 9781450367509. DOI: 10.1145/3320269.3384744. [Online]. Available: <https://doi.org/10.1145/3320269.3384744>.
- [130] P. Chaudhary, B. B. Gupta, K. T. Chui, and S. Yamaguchi, "Shielding smart home iot devices against adverse effects of xss using ai model," in *2021 IEEE International Conference on Consumer Electronics (ICCE)*, 2021, pp. 1–5. DOI: 10.1109/ICCE50685.2021.9427591.

- [131] P. Chaudhary, B. B. Gupta, and A. Singh, "Xss armor: Constructing xss defensive framework for preserving big data privacy in internet-of-things (iot) networks," *Journal of Circuits, Systems and Computers*, vol. 31, May 2022. DOI: 10.1142/S021812662250222X.
- [132] P. Chaudhary, B. B. Gupta, C. Choi, and K. T. Chui, "Xsspro: Xss attack detection proxy to defend social networking platforms," in *Computational Data and Social Networks*, S. Chellappan, K.-K. R. Choo, and N. Phan, Eds., Cham: Springer International Publishing, 2020, pp. 411–422, ISBN: 978-3-030-66046-8.
- [133] P. Chaudhary, B. B. Gupta, and S. Gupta, "Defending the osn-based web applications from xss attacks using dynamic javascript code and content isolation," in *Quality, IT and Business Operations: Modeling and Optimization*, P. Kapur, U. Kumar, and A. K. Verma, Eds. Singapore: Springer Singapore, 2018, pp. 107–119, ISBN: 978-981-10-5577-5. DOI: 10.1007/978-981-10-5577-5_9. [Online]. Available: https://doi.org/10.1007/978-981-10-5577-5_9.
- [134] I. Odun-Ayo, W. Abasi, M. Adebisi, and O. Alagbe, "An implementation of real-time detection of cross-site scripting attacks on cloud-based web applications using deep learning," *Bulletin of Electrical Engineering and Informatics*, vol. 10, pp. 2442–2453, Oct. 2021. DOI: 10.11591/eei.v10i5.3168.
- [135] L. Li and L. Wei, "Automatic xss detection and automatic anti-anti-virus payload generation," in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2019, pp. 71–76. DOI: 10.1109/CyberC.2019.00021.
- [136] X. Hu, N. Sastry, and M. Mondal, "Cccc: Corraling cookies into categories with cookiemonster," in *Proceedings of the 13th ACM Web Science Conference 2021*, ser. WebSci '21, Virtual Event, United Kingdom: Association for Computing Machinery, 2021, pp. 234–242, ISBN: 9781450383301. DOI: 10.1145/3447535.3462509. [Online]. Available: <https://doi.org/10.1145/3447535.3462509>.
- [137] F. Mokbal, W. Dan, and X. Wang, "Detect cross-site scripting attacks using average word embedding and support vector machine," *International Journal of Network Security*, vol. 24, pp. 20–28, Jan. 2022. DOI: 10.6633/IJNS.202201.
- [138] U. Tanielian, A.-M. Tusch, and F. Vasile, "Siamese cookie embedding networks for cross-device user matching," in *Companion Proceedings of the The Web Conference 2018*, Republic and Canton of Geneva, CHE: International World Wide Web

- Conferences Steering Committee, 2018, pp. 85–86, ISBN: 9781450356404. DOI: 10.1145/3184558.3186941. [Online]. Available: <https://doi.org/10.1145/3184558.3186941>.
- [139] S. Kascheev and T. Olenchikova, “The detecting cross-site scripting (xss) using machine learning methods,” in *2020 Global Smart Industry Conference (GloSIC), 2020*, pp. 265–270. DOI: 10.1109/GloSIC50886.2020.9267866.
- [140] H.-C. Chen, A. Nshimiyimana, C. Damarjati, and P.-H. Chang, “Detection and prevention of cross-site scripting attack with combined approaches,” in *2021 International Conference on Electronics, Information, and Communication (ICEIC), 2021*, pp. 1–4. DOI: 10.1109/ICEIC51217.2021.9369796.
- [141] G. Kaur, Y. Malik, H. Samuel, and F. Jaafar, “Detecting blind cross-site scripting attacks using machine learning,” in *Proceedings of the 2018 International Conference on Signal Processing and Machine Learning*, ser. SPML '18, Shanghai, China: Association for Computing Machinery, 2018, pp. 22–25, ISBN: 9781450366052. DOI: 10.1145/3297067.3297096. [Online]. Available: <https://doi.org/10.1145/3297067.3297096>.
- [142] S. Akaishi and R. Uda, “Classification of xss attacks by machine learning with frequency of appearance and co-occurrence,” in *2019 53rd Annual Conference on Information Sciences and Systems (CISS), 2019*, pp. 1–6. DOI: 10.1109/CISS.2019.8693047.
- [143] Y. Fang, C. Huang, Y. Xu, and Y. Li, “Rlxss: Optimizing xss detection model to defend against adversarial attacks based on reinforcement learning,” *Future Internet*, vol. 11, p. 177, Aug. 2019. DOI: 10.3390/fi11080177.
- [144] X.-Y. Hou, X.-L. Zhao, M.-J. Wu, R. Ma, and Y.-P. Chen, “A dynamic detection technique for xss vulnerabilities,” in *2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC), 2018*, pp. 34–43.
- [145] C. Lv, L. Zhang, F. Zeng, and J. Zhang, “Adaptive random testing for xss vulnerability,” in *2019 26th Asia-Pacific Software Engineering Conference (APSEC), 2019*, pp. 63–69. DOI: 10.1109/APSEC48747.2019.00018.
- [146] Y. Zhou and P. Wang, “An ensemble learning approach for xss attack detection with domain knowledge and threat intelligence,” *Computers & Security*, vol. 82, pp. 261–269, 2019, ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2018.12.016>.

[Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167404818306370>.

- [147] A. Sinha and S. Tripathy, "Cookiearmor : Safeguarding against cross-site request forgery and session hijacking," *Security and Privacy*, vol. 2, e60, Feb. 2019. DOI: 10.1002/spy2.60.
- [148] A. Lavrenovs and F. J. R. Melón, "Http security headers analysis of top one million websites," in *2018 10th International Conference on Cyber Conflict (CyCon)*, 2018, pp. 345–370. DOI: 10.23919/CYCON.2018.8405025.
- [149] F. Mokbal, M. Mahiuob, W. Dan, *et al.*, "Mlpxss: An integrated xss-based attack detection scheme in web applications using multilayer perceptron technique," *IEEE Access*, vol. 7, pp. 100 567–100 580, 2019. DOI: 10.1109/ACCESS.2019.2927417.
- [150] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," New York, NY, USA: Association for Computing Machinery, 2016, ISBN: 9781450341394.
- [151] D. E. Simos, B. Garn, J. Zivanovic, and M. Leithner, "Practical combinatorial testing for xss detection using locally optimized attack models," in *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2019, pp. 122–130. DOI: 10.1109/ICSTW.2019.00040.
- [152] G. Habibi and N. Surantha, "Xss attack detection with machine learning and n-gram methods," in *2020 International Conference on Information Management and Technology (ICIMTech)*, 2020, pp. 516–520. DOI: 10.1109/ICIMTech50083.2020.9210946.
- [153] O. J. Falana, I. O. Ebo, C. O. Tinubu, O. A. Adejimi, and A. Ntuk, "Detection of cross-site scripting attacks using dynamic analysis and fuzzy inference system," in *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, 2020, pp. 1–6. DOI: 10.1109/ICMCECS47690.2020.240871.
- [154] S. Gupta, B. B. Gupta, and P. Chaudhary, "A client-server javascript code rewriting-based framework to detect the xss worms from online social network," *Concurrency and Computation Practice and Experience*, vol. 31, May 2018.
- [155] V. Papaspirou, L. Maglaras, and M. A. Ferrag, "A tutorial on cross site scripting attack - defense," Dec. 2020. DOI: 10.20944/preprints202012.0063.v1.

- [156] M. Mohammadi, B. Chu, and H. Richter Lipford, "Automated repair of cross-site scripting vulnerabilities through unit testing," in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2019, pp. 370–377. DOI: 10.1109/ISSREW.2019.00098.
- [157] G. Kaur, B. Pande, A. Bhardwaj, G. Bhagat, and S. Gupta, "Defense against html5 xss attack vectors: A nested context-aware sanitization technique," in *2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, 2018, pp. 442–446. DOI: 10.1109/CONFLUENCE.2018.8442855.
- [158] H. Choi, S. Hong, S. Cho, and Y.-G. Kim, "Hxd: Hybrid xss detection by using a headless browser," in *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, 2017, pp. 1–4. DOI: 10.1109/CAIPT.2017.8320672.
- [159] G. Xu, X. Xie, S. Huang, *et al.*, "Jscsp: A novel policy-based xss defense mechanism for browsers," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 862–878, 2022. DOI: 10.1109/TDSC.2020.3009472.
- [160] R. Putthacharoen and P. Bunyatneparat, "Protecting cookies from cross site script attacks using dynamic cookies rewriting technique," in *13th International Conference on Advanced Communication Technology (ICACT2011)*, 2011, pp. 1090–1094.
- [161] S. Gupta and B. B. Gupta, "XSS-secure as a service for the platforms of online social network-based multimedia web applications in cloud," *Multimedia Tools and Applications*, vol. 77, no. 4, pp. 4829–4861, Jul. 2016. DOI: 10.1007/s11042-016-3735-1. [Online]. Available: <https://doi.org/10.1007/s11042-016-3735-1>.
- [162] S. Gupta and B. B. Gupta, "A robust server-side javascript feature injection-based design for jsp web applications against xss vulnerabilities," in *Cyber Security*, M. U. Bokhari, N. Agrawal, and D. Saini, Eds., Singapore: Springer Singapore, 2018, pp. 459–465, ISBN: 978-981-10-8536-9.
- [163] A. K. Dalai, S. D. Ankush, and S. K. Jena, "Xss attack prevention using dom-based filter," in *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*, P. K. Sa, M. N. Sahoo, M. Murugappan, Y. Wu, and B. Majhi, Eds., Singapore: Springer Singapore, 2018, pp. 227–234, ISBN: 978-981-10-3376-6.

- [164] U. Iqbal, P. Snyder, S. Zhu, B. Livshits, Z. Qian, and Z. Shafiq, “Adgraph: A graph-based approach to ad and tracker blocking,” in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 763–776. DOI: 10.1109/SP40000.2020.00005.
- [165] T. A. Taha and M. Karabatak, “A proposed approach for preventing cross-site scripting,” in *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, 2018, pp. 1–4. DOI: 10.1109/ISDFS.2018.8355356.
- [166] D. Zubarev and I. Skarga-Bandurova, “Cross-site scripting for graphic data: Vulnerabilities and prevention,” in *2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 2019, pp. 154–160. DOI: 10.1109/DESSERT.2019.8770043.
- [167] P. Chen, H. Yu, M. Zhao, and J. Wang, “Research and implementation of cross-site scripting defense method based on moving target defense technology,” in *2018 5th International Conference on Systems and Informatics (ICSAI)*, 2018, pp. 818–822. DOI: 10.1109/ICSAI.2018.8599463.
- [168] C. Li, Y. Wang, C. Miao, and C. Huang, “Cross-site scripting guardian: A static xss detector based on data stream input-output association mining,” *Applied Sciences*, vol. 10, no. 14, 2020, ISSN: 2076-3417. DOI: 10.3390/app10144740. [Online]. Available: <https://www.mdpi.com/2076-3417/10/14/4740>.
- [169] D. Dembla, Y. Chaba, K. Yadav, M. Chaba, and A. Kumar, “A novel and efficient technique for prevention of xss attacks using knapsack based cryptography,” *Advances in Mathematics: Scientific Journal*, vol. 9, pp. 4513–4521, Jul. 2020.
- [170] S. Zimmeck and K. Alicki, “Standardizing and implementing do not sell,” in *Proceedings of the 19th Workshop on Privacy in the Electronic Society*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 15–20, ISBN: 9781450380867. [Online]. Available: <https://doi.org/10.1145/3411497.3420224>.
- [171] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, “The web never forgets: Persistent tracking mechanisms in the wild,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’14, New York, NY, USA: Association for Computing Machinery, 2014, pp. 674–689, ISBN: 9781450329576. DOI: 10.1145/2660267.2660347. [Online]. Available: <https://doi.org/10.1145/2660267.2660347>.

- [172] X. Hu and N. Sastry, "What a tangled web we weave: Understanding the interconnectedness of the third party cookie ecosystem," in *12th ACM Conference on Web Science*, ser. WebSci '20, Southampton, United Kingdom: Association for Computing Machinery, 2020, pp. 76–85, ISBN: 9781450379892. DOI: 10.1145/3394231.3397897. [Online]. Available: <https://doi.org/10.1145/3394231.3397897>.
- [173] P. Papadopoulos, N. Kourtellis, and E. Markatos, "Cookie synchronization: Everything you always wanted to know but were afraid to ask," in *The World Wide Web Conference*, ser. WWW '19, San Francisco, CA, USA: Association for Computing Machinery, 2019, pp. 1432–1442, ISBN: 9781450366748. DOI: 10.1145/3308558.3313542. [Online]. Available: <https://doi.org/10.1145/3308558.3313542>.
- [174] S. Syaifuddin, D. Risqiwati, and H. A. Sidharta, "Automation snort rule for xss detection with honeypot," in *2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2018, pp. 584–588. DOI: 10.1109/EECSI.2018.8752961.
- [175] P. Nagarjun and S. S. Ahamad, "Attack data analysis to find cross-site scripting attack patterns," *ARPN Journal of Engineering and Applied Sciences*, vol. 13, no. 17, 2018, ISSN: 1819-6608. [Online]. Available: http://www.arpnjournals.org/jeas/research_papers/rp_2018/jeas_0918_7264.pdf.
- [176] K. Drakonakis, S. Ioannidis, and J. Polakis, "The cookie hunter: Automated black-box auditing for web authentication and authorization flaws," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1953–1970, ISBN: 9781450370899. [Online]. Available: <https://doi.org/10.1145/3372297.3417869>.
- [177] F. M. M. Mokbal, W. Dan, W. Xiaoxi, Z. Wenbin, and F. Lihua, "Xgbxss: An extreme gradient boosting detection framework for cross-site scripting attacks based on hybrid feature selection approach and parameters optimization," *Journal of Information Security and Applications*, vol. 58, p. 102813, 2021, ISSN: 2214-2126. DOI: <https://doi.org/10.1016/j.jisa.2021.102813>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2214212621000533>.
- [178] X. Hu and N. R. Sastry, "Characterising third party cookie usage in the eu after gdpr," *Proceedings of the 10th ACM Conference on Web Science*, 2019.

- [179] I. Khazal and M. Hussain, "Server side method to detect and prevent stored xss attack," *Iraqi Journal for Electrical and Electronic Engineering*, vol. 17, pp. 58–65, Dec. 2021. DOI: 10.37917/ijeee.17.2.8.
- [180] X. Zhang, Y. Zhou, S. Pei, J. Zhuge, and J. Chen, "Adversarial examples detection for xss attacks based on generative adversarial networks," *IEEE Access*, vol. 8, pp. 10 989–10 996, 2020. DOI: 10.1109/ACCESS.2020.2965184.
- [181] R. Wibowo and A. Sulaksono, "Web vulnerability through cross site scripting (xss) detection with owasp security shepherd," *Indonesian Journal of Information Systems*, vol. 3, p. 149, Feb. 2021. DOI: 10.24002/ijis.v3i2.4192.
- [182] E. Papadogiannakis, P. Papadopoulos, N. Kourtellis, and E. P. Markatos, "User tracking in the post-cookie era: How websites bypass gdpr consent to track users," in *Proceedings of the Web Conference 2021*, ser. WWW '21, Ljubljana, Slovenia: Association for Computing Machinery, 2021, pp. 2130–2141, ISBN: 9781450383127. DOI: 10.1145/3442381.3450056. [Online]. Available: <https://doi.org/10.1145/3442381.3450056>.
- [183] P. Papadopoulos, N. Kourtellis, and E. P. Markatos, "Exclusive: How the (synced) cookie monster breached my encrypted vpn session," in *Proceedings of the 11th European Workshop on Systems Security*, ser. EuroSec'18, Porto, Portugal: Association for Computing Machinery, 2018, ISBN: 9781450356527. DOI: 10.1145/3193111.3193117.
- [184] H. Takahashi, K. Yasunaga, M. Mambo, K. Kim, and H. Y. Youm, "Preventing abuse of cookies stolen by xss," in *2013 Eighth Asia Joint Conference on Information Security*, 2013, pp. 85–89. DOI: 10.1109/ASIAJCIS.2013.20.
- [185] K. LaCroix, Y. L. Loo, and Y. B. Choi, "Cookies and sessions: A study of what they are, how they work and how they can be stolen," in *2017 International Conference on Software Security and Assurance (ICSSA)*, 2017, pp. 20–24. DOI: 10.1109/ICSSA.2017.9.
- [186] P. Mishra and C. Gupta, "Cookies in a cross-site scripting: Type, utilization, detection, protection and remediation," in *2020 8th International Conference on Reliability, Inform Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2020, pp. 1056–1059. DOI: 10.1109/ICRITO48877.2020.9198003.

- [187] S. Kumar B J and Sahana, "Detection and avoidance of web vulnerability using xss," Dec. 2020. DOI: 10.35940/ijrte.B1039.078219.
- [188] G. S and G. B, "Evaluation and monitoring of xss defensive solutions: A survey, open research issues and future directions.," in *J Ambient Intell Human Computer*, Nov. 2019, pp. 4377–4405. DOI: 10.1007/s12652-018-1118-3.
- [189] D. Korać, B. Damjanović, and D. Simić, "Information security in m-learning systems: Challenges and threats of using cookies," in *2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2020, pp. 1–6. DOI: 10.1109/INFOTEH48170.2020.9066344.
- [190] H. Takahashi, K. Yasunaga, M. Mambo, K. Kim, and H. Y. Youm, "Preventing abuse of cookies stolen by xss," in *2013 Eighth Asia Joint Conference on Information Security*, 2013, pp. 85–89. DOI: 10.1109/ASIAJCIS.2013.20.
- [191] W. Alcorn. "The browser exploitation framework." (), [Online]. Available: <https://beefproject.com/>. (accessed: 24.04.2024).
- [192] T. Fawcett, "An introduction to roc analysis," in *Pattern Recogn Lett*, 2006, pp. 861–74.
- [193] C. Cortes and M. Mohri, "Auc optimization vs. error rate minimization," in *Advances in neural information processing systems*, 2004, pp. 313–320.
- [194] ¿Es la curiosidad un signo de inteligencia? — *es.quora.com*, <https://es.quora.com/Es-la-curiosidad-un-signo-de-inteligencia>, [Accessed 24-04-2014].
- [195] *PENTABYTE - No instales, No configures..Desarrolla* — *pentabyte-blog.blogspot.com*, <http://pentabyteblog.blogspot.com/>, [Accessed: 24.04.2014].
- [196] *Blogger.com - Create a unique and beautiful blog easily.* — *blogger.com*, <https://www.blogger.com/about/?bpli=1&ppli=1>, [Accessed 30-04-2018].
- [197] A. O. Mahdi, M. I. Alhabbash, and S. S. A. Naser, "An intelligent tutoring system for teaching advanced topics in information security," *World Wide Journal of Multidisciplinary Research and Development*, vol. 2, no. 12, pp. 1–9, 2016.
- [198] T. Vidas, E. Owusu, S. Wang, C. Zeng, L. F. Cranor, and N. Christin, "Qrishing: The susceptibility of smartphone users to qr code phishing attacks," in *Financial Cryptography and Data Security*, A. A. Adams, M. Brenner, and M. Smith, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 52–69, ISBN: 978-3-642-41320-9.

- [199] *URL Shortener, Branded Short Links & Analytics | TinyURL — tinyurl.com*, <https://tinyurl.com/app>, []
- [200] VirusTotal, *Virustotal*, <https://www.virustotal.com/>, [June 2019].
- [201] Pentest-Tools. “Pentest yourself. don’t get hacked.” July 2019. (), [Online]. Available: <https://pentest-tools.com/home>.
- [202] Netcraft. “Internet security and data mining.” July 2019. (), [Online]. Available: <https://www.netcraft.com/>.
- [203] Pentest-Tools. “Pentest yourself. don’t get hacked.” July 2019. (), [Online]. Available: <https://pentest-tools.com/home>.
- [204] Ping.eu. “Online ping, traceroute, dns lookup.” July 2019. (), [Online]. Available: <https://ping.eu/>.
- [205] K. O. Design. “Network tools by yougetsignal.” July 2019. (), [Online]. Available: <https://www.yougetsignal.com/>.
- [206] U. Q. NET. “Url scanner.” July 2019. (), [Online]. Available: <https://urlquery.net/info>.
- [207] Riskemy. “9 cross-site scripting (xss) scan testing tools online.” July 2019. (), [Online]. Available: <https://riskemy.com/xss-testing-tool/>.
- [208] Find-XSS.net. “Web monitoring.” July 2019. (), [Online]. Available: <https://find-xss.net/?l=en>.
- [209] Quttera. “Threatsign! website anti-malware.” July 2019. (), [Online]. Available: <https://quttera.com/>.
- [210] XSS-Scanner. “Cross site scripting scanner.” July 2019. (), [Online]. Available: <https://xss-scanner.com/>.
- [211] Acunetix. “Xss vulnerability scanning with acunetix.” July 2019. (), [Online]. Available: <https://www.acunetix.com/vulnerability-scanner/xss-vulnerability-scanning/>.
- [212] w3af.org. “Xss.” July 2019. (), [Online]. Available: <http://w3af.org/plugins/audit/xss>.
- [213] Q. Inc. “Ssl/tls capabilities of your browser.” July 2019. (), [Online]. Available: <https://www.ssllabs.com/ssltest/viewMyClient.html>.

- [214] T. Inc. “Web app scanning.” July 2019. (), [Online]. Available: <https://www.tenable.com/products/tenable-io/web-application-scanning>.
- [215] S. SRL. “Cross site scripting: What do you need to know about it?” July 2019. (), [Online]. Available: <https://www.swascan.com/cross-site-scripting/>.
- [216] Rapid7. “Web application security and scanning.” July 2019. (), [Online]. Available: <https://www.rapid7.com/fundamentals/web-application-security/>.
- [217] GitHub/menkrep1337. “Xsscon.” July 2019. (), [Online]. Available: <https://github.com/menkrep1337/XSSCon>.
- [218] KNOXXS. “The best tool to find and prove XSS flaws.” (2020, May 10). ().
- [219] *Guía para tratamiento de datos personales en administración pública*. [Online]. Available: <https://www.gobiernoelectronico.gob.ec/wp-content/uploads/2019/11/Gu%C3%ADa-de-protecci%C3%B3n-de-datos-personales.pdf>.
- [220] D. Schaper, *Pi-hole network-wide ad blocking*, Jan. 2024. [Online]. Available: <https://pi-hole.net/>.
- [221] Wireshark, *Tshark(1) manual page*, Jan. 2024. [Online]. Available: <https://www.wireshark.org/docs/man-pages/tshark.html>.
- [222] Sphinx, *Pyautogui’s documentation*, Apr. 2019. [Online]. Available: <https://pyautogui.readthedocs.io/en/latest/>.
- [223] Cisco. “What is penetration testing?” (2021), [Online]. Available: <https://www.cisco.com/c/en/us/products/security/what-is-pen-testing.html>. (accessed: 18.03.2024).
- [224] “Introducing penetration test with case study and course project in cybersecurity education,” *Journal of The Colloquium for Information Systems Security Education*, 2022. DOI: 10.53735/cisse.v9i1.148.
- [225] Nuclio. “¿qué es el pentesting?” (), [Online]. Available: <https://nuclio.school/blog/que-es-el-pentesting/#Que-fases-tiene-el-Pentesting>. (accessed: 19.03.2024).
- [226] H. Kopka, P. W. Daly, and S. Rahtz, *Guide to LATEX*. Addison-Wesley Boston, MA, 2004, vol. 4.
- [227] L. Lamport, *LaTeX*. SCompany Cyfronet, 1991.

- [228] F. Mittelbach, M. Goossens, J. Braams, D. Carlisle, and C. Rowley, *The LATEX companion*. Addison-Wesley Professional, 2004.
- [229] A. Alabrah and M. Bassiouni, “Robust and fast authentication of session cookies in collaborative and social media using position-indexed hashing,” in *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Work-sharing*, 2013, pp. 241–249. DOI: 10.4108/icst.collaboratecom.2013.254126.
- [230] T. Vidas, E. Owusu, S. Wang, C. Zeng, L. F. Cranor, and N. Christin, “Qrishing: The susceptibility of smartphone users to qr code phishing attacks,” in *Financial Cryptography and Data Security*, A. A. Adams, M. Brenner, and M. Smith, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 52–69, ISBN: 978-3-642-41320-9.