Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

School of Computer Science & Engineering

Department of Computer Science and Applications

2023-2024

A

PROJECT REPORT

ON

(WEBSITE VULNARABILTY ASSESSMENT AND PENETRATION TESTING)

BY

Nikhil Bhandari

IN PARTIAL FULFILLMENT OF

MASTER OF COMPUTER APPLICATION- SCIENCE

DR. VISHWANATH KARAD MIT WORLD PEACE UNIVERSITY, PUNE-411038

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS

Certificate

This is to certify that**, Nikhil Bhandari** student of MCA Science Semester IV has successfully / partially completed Industrial Training at Xentek Secured OPC Ltd in partial fulfilment of MCA- Science under Dr. Vishwanath Karad MIT World Peace University, for the academic year 2023-2024.

Dr. Mohammad Idrees Bhat	Dr. Jalindar R. Gandal
Internal Guide	Head-MCA-Science
Dr. Rajeshree Khande	Prof. Dr. Lalit Kane
Program Director	Associate Dean Academics

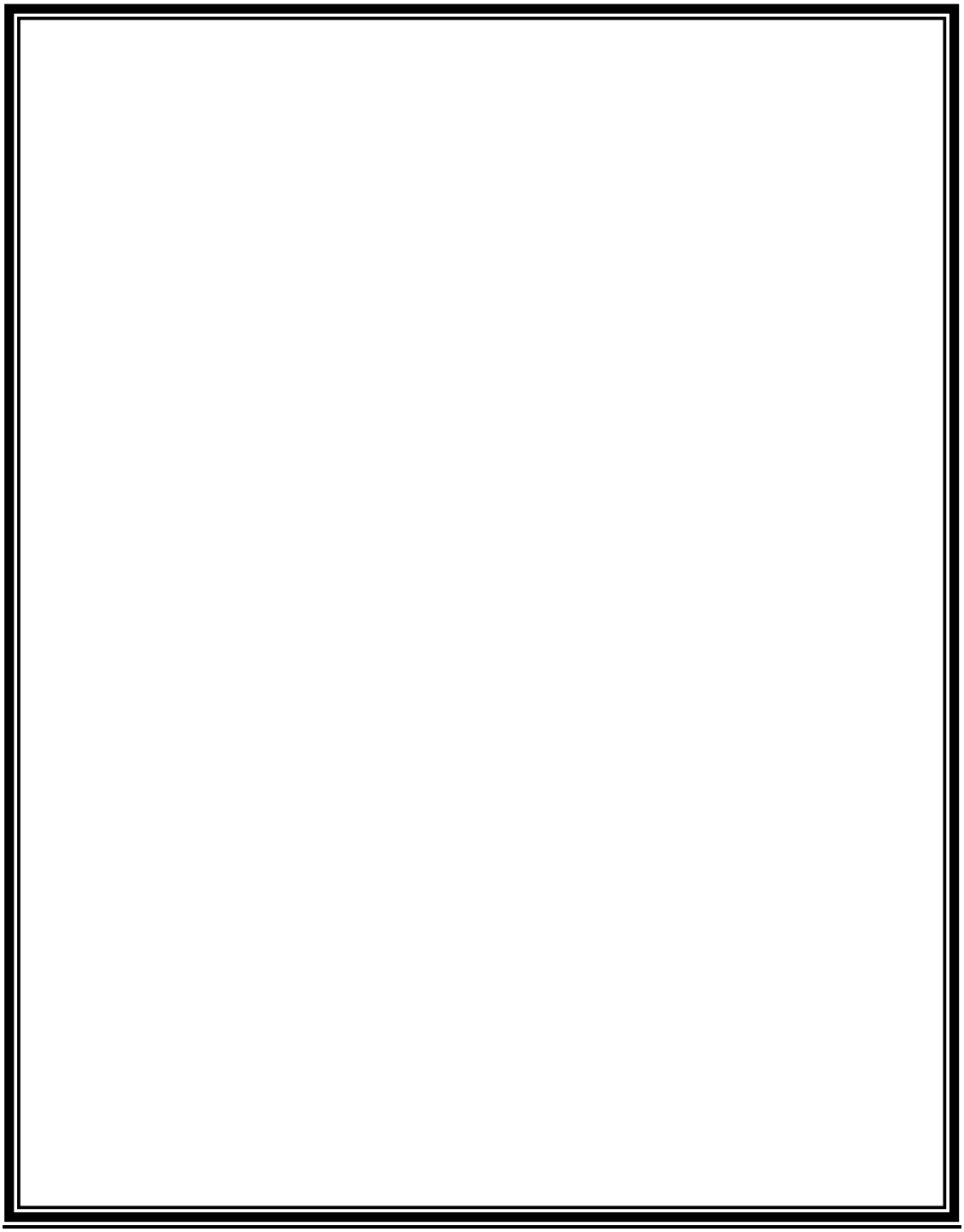
Prof. Dr. Shubhalaxmi Joshi
Asso. Dean, Faculty Affairs

Date: ___/___/_____

External Examiners:

1. _____ 2. _____

INTERNSHIP OFFER LETTER OR COMPLETION CERTIFICATE



ACKNOWLEDGEMENT

I wish to express my gratitude for providing the facilities of the Institute and for his encouragement during the course of this work. I would also 1ike to gratefully acknowledge the enthusiastic supervision of my internship guide, Prof. Mohammad Idrees Bhat, School of Computer Science, MIT-WPU, Pune for his continuous, valuable guidance, patience, constant care and kind encouragement throughout the internship work that made me present this internship report in an efficient manner.

I would like to thank my industry mentors Mr. Santosh Verma and Mr. Deepesh Badgujar for providing me with guidance and help on every step of the way during the course of this internship and for imparting me with invaluable knowledge and teaching me the etiquettes of a professional employee.

Final1y, I wish to thank my family members and my friends who have always been very supportive and encouraging.

Sincerely,

Nikhil Bhandari

Place: _____

Date: _____

DECLARATION

I hereby declare that the project work entitled "Vulnerability Assessment And Penetration Testing" submitted to the MIT-WPU, Pune is a record of an original work done by me under the guidance of Mr. Santosh Verma and Deepesh Badgujar and this project work is submitted in the partial fulfilment of the requirements for the internship and the award of the degree of Master of Computer Application.

The results embodied in this internship report have not been submitted to any other University or Institute for the award of any degree or diploma. I have contributed myself for this project going on in our company.

Nikhil Bhandari

MCA (Science)

INDEX

1. PROBLEM DEFINITION

Introduction to Vulnerability Assessment and Penetration Testing (VAPT)

In an increasingly interconnected digital landscape, where businesses rely heavily on information technology infrastructure, the importance of securing these systems against potential threats cannot be overstated. Vulnerability Assessment and Penetration Testing (VAPT) emerges as a crucial methodology in ensuring the resilience of digital systems against malicious actors and unforeseen vulnerabilities.

VAPT encompasses a comprehensive approach towards assessing, identifying, and mitigating security weaknesses within an organization's IT infrastructure. It involves two primary components: Vulnerability Assessment (VA) and Penetration Testing (PT).

Vulnerability Assessment (VA) is the systematic process of identifying, quantifying, and prioritizing vulnerabilities within a system. This involves the use of automated tools and manual inspection to scan networks, applications, and other digital assets for potential weaknesses. VA provides organizations with insights into their security posture, enabling them to proactively address vulnerabilities before they can be exploited.

Penetration Testing (PT), on the other hand, is a controlled simulation of real-world cyberattacks aimed at exploiting identified vulnerabilities. Unlike VA, PT involves actively attempting to penetrate systems to assess their resilience against potential threats. Penetration testers, often referred to as ethical hackers, employ various techniques and methodologies to mimic the tactics of malicious actors, providing organizations with valuable insights into their security defences and areas for improvement.

The primary objectives of VAPT include:

1. Identifying Vulnerabilities: To uncover weaknesses and flaws within an organization's IT infrastructure, including networks, applications, and devices.

2. Assessing Security Posture: To evaluate the effectiveness of existing security controls and measures in place, and to identify gaps or areas of improvement.

3. Mitigating Risks: To provide actionable recommendations and strategies for mitigating identified vulnerabilities and strengthening overall security posture.

4. Compliance and Regulation: To ensure adherence to industry standards, regulatory requirements, and best practices in cybersecurity.

VAPT plays a critical role in helping organizations stay ahead of evolving cyber threats, safeguarding sensitive data, preserving customer trust, and maintaining business continuity. By regularly conducting VAPT assessments, organizations can proactively identify and address security vulnerabilities, ultimately reducing the risk of data breaches, financial losses, and reputational damage.

In this report, we delve into the intricacies of VAPT, exploring its methodologies, benefits, challenges, and best practices. Through a comprehensive analysis, we aim to provide valuable insights into the role of VAPT in modern cybersecurity strategies and its significance in safeguarding digital assets in an ever-evolving threat landscape.

2. Tools used for VAPT:

2.1 NMAP

Nmap, short for Network Mapper, is a powerful open-source tool used for network discovery and security auditing. It allows users to scan networks to find hosts, services, and open ports, providing valuable information about the network's topology and potential vulnerabilities. Nmap offers various scanning techniques, including TCP SYN, UDP, and ACK scans, making it versatile for different network environments. It's widely utilized by network administrators, security professionals, and hackers alike for assessing network security posture and detecting potential threats. With its extensive features and flexible options, Nmap remains a staple in network reconnaissance and security assessments.

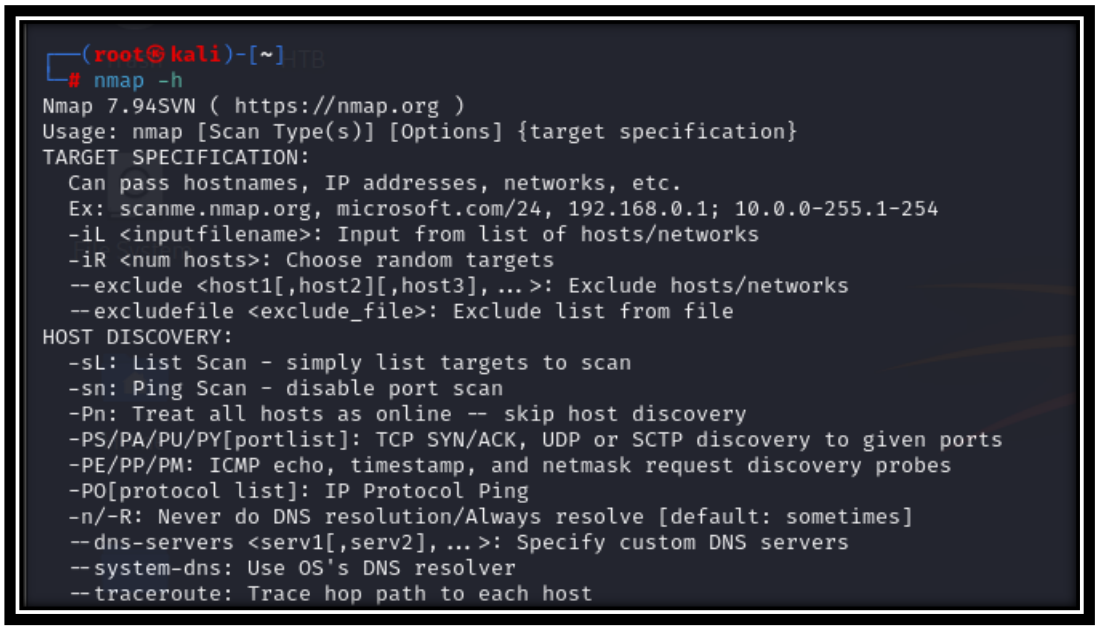


Figure 1 Nmap Help

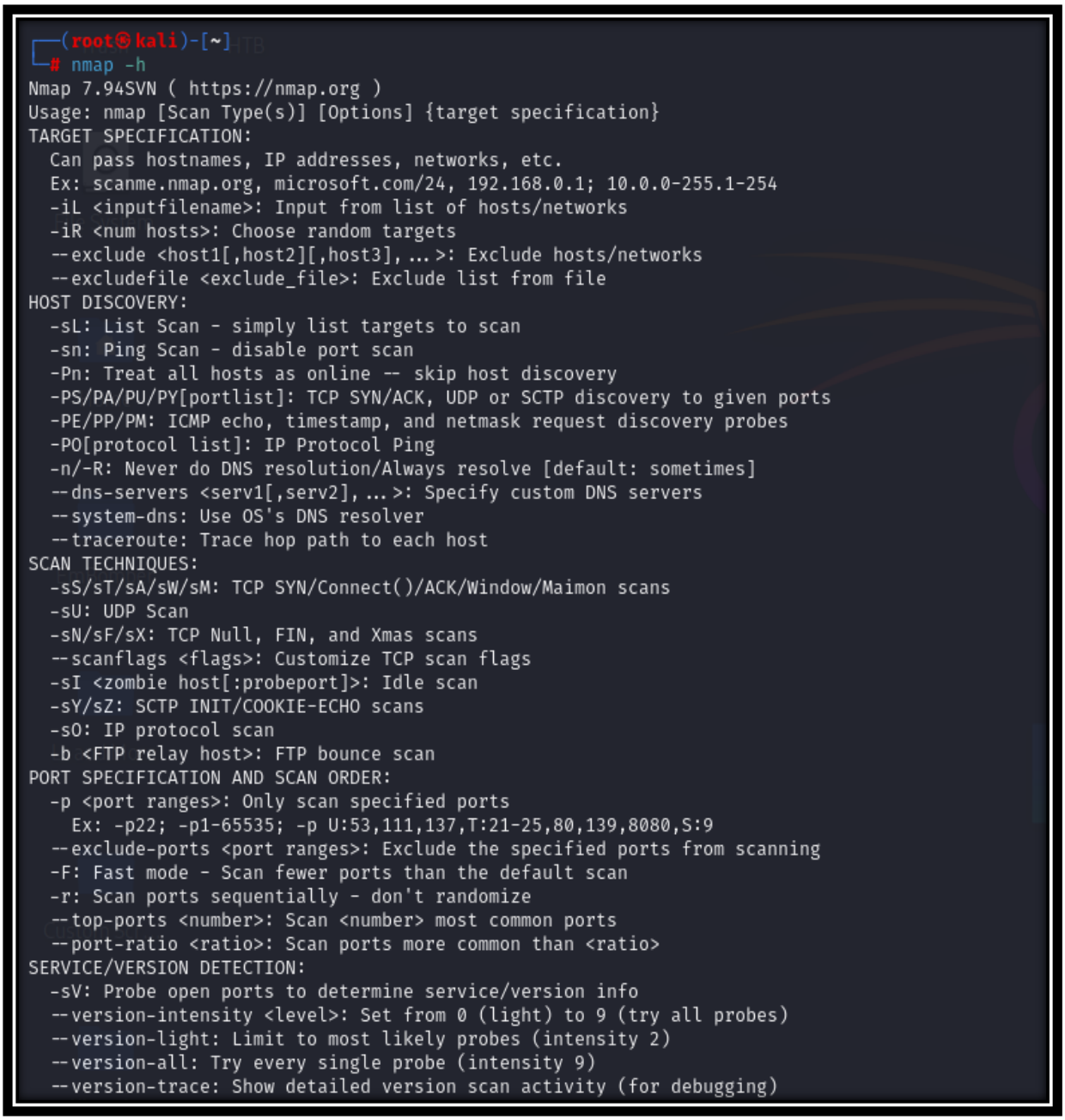


Figure 2 Nmap Help

```
  --script=<Lua scripts>: <Lua scripts> is a comma separated list of
          directories, script-files or script-categories
  --script-args=<n1=v1,[n2=v2, ... ]>: provide arguments to scripts
  --script-args-file=filename: provide NSE script args in a file
  --script-trace: Show all data sent and received
  --script-updatedb: Update the script database.
  --script-help=<Lua scripts>: Show help about scripts.
          <Lua scripts> is a comma-separated list of script-files or
          script-categories.
OS DETECTION:
  -O: Enable OS detection
  --osscan-limit: Limit OS detection to promising targets
  --osscan-guess: Guess OS more aggressively
TIMING AND PERFORMANCE:
  Options which take <time> are in seconds, or append 'ms' (milliseconds),
  's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
  -T<0-5>: Set timing template (higher is faster)
  --min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
  --min-parallelism/max-parallelism <numprobes>: Probe parallelization
  --min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies
      probe round trip time.
  --max-retries <tries>: Caps number of port scan probe retransmissions.
  --host-timeout <time>: Give up on target after this long
  --scan-delay/--max-scan-delay <time>: Adjust delay between probes
  --min-rate <number>: Send packets no slower than <number> per second
  --max-rate <number>: Send packets no faster than <number> per second
FIREWALL/IDS EVASION AND SPOOFING:
  -f; --mtu <val>: fragment packets (optionally w/given MTU)
  -D <decoy1,decoy2[,ME],...>: Cloak a scan with decoys
  -S <IP_Address>: Spoof source address
  -e <iface>: Use specified interface
  -g/--source-port <portnum>: Use given port number
  --proxies <url1,[url2],...>: Relay connections through HTTP/SOCKS4 proxies
  --data <hex string>: Append a custom payload to sent packets
  --data-string <string>: Append a custom ASCII string to sent packets
  --data-length <num>: Append random data to sent packets
  --ip-options <options>: Send packets with specified ip options
  --ttl <val>: Set IP time-to-live field
  --spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address
  --badsum: Send packets with a bogus TCP/UDP/SCTP checksum
OUTPUT:
  -oN/-oX/-oS/-oG <file>: Output scan in normal, XML, s|<rIpt kIddi3,
     and Grepable format, respectively, to the given filename.
  -oA <basename>: Output in the three major formats at once
  -v: Increase verbosity level (use -vv or more for greater effect)
  -d: Increase debugging level (use -dd or more for greater effect)
  --reason: Display the reason a port is in a particular state
  --open: Only show open (or possibly open) ports
```

Figure 3 Nmap Help

2.2 ZENMAP

Zenmap is a graphical user interface (GUI) for Nmap, designed to simplify network scanning and analysis. It provides a user-friendly interface that allows users to perform network discovery, host scanning, and vulnerability detection without needing to use complex command-line options. Zenmap offers features like topology mapping, host grouping, and report generation, making it accessible to both novice and experienced users. It's a valuable tool for network administrators and security professionals to visualize and understand their network's security posture. Zenmap complements Nmap's powerful scanning capabilities with an intuitive interface for easier navigation and analysis.

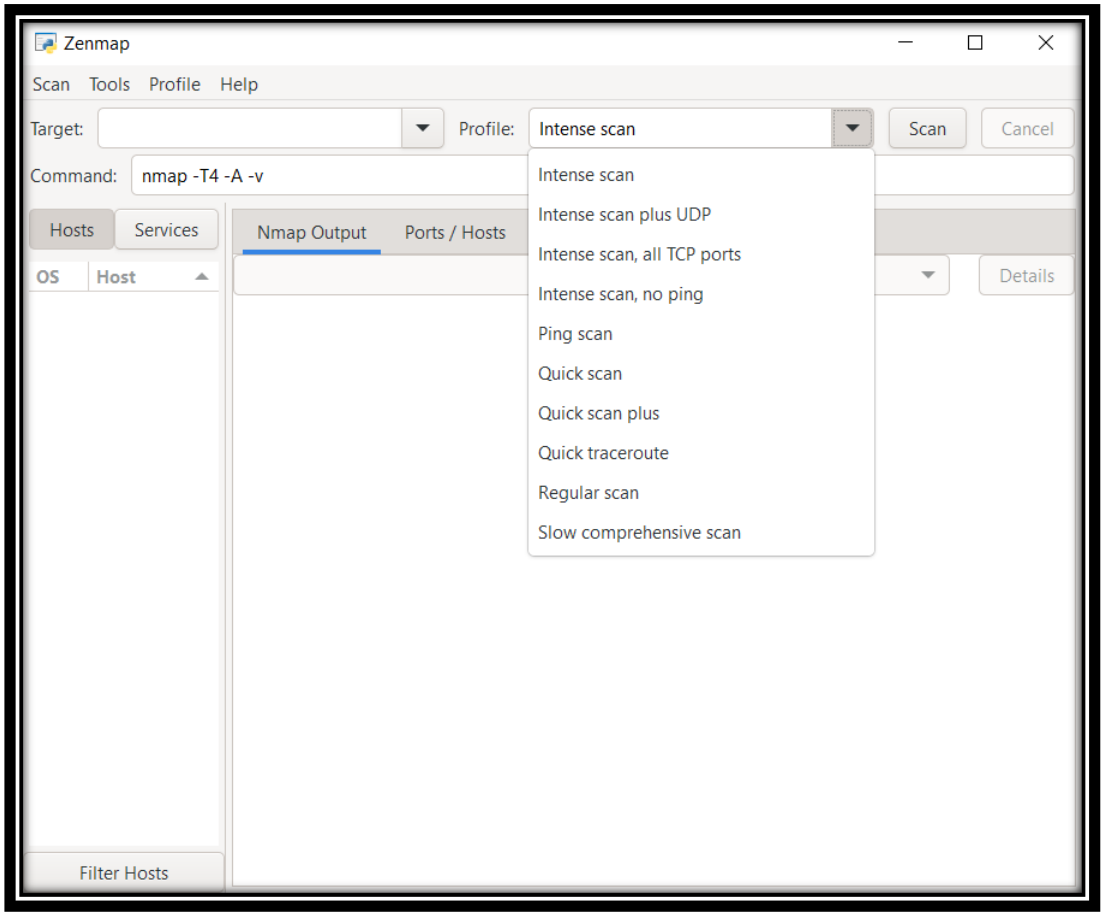


Figure 4 Zenmap UI

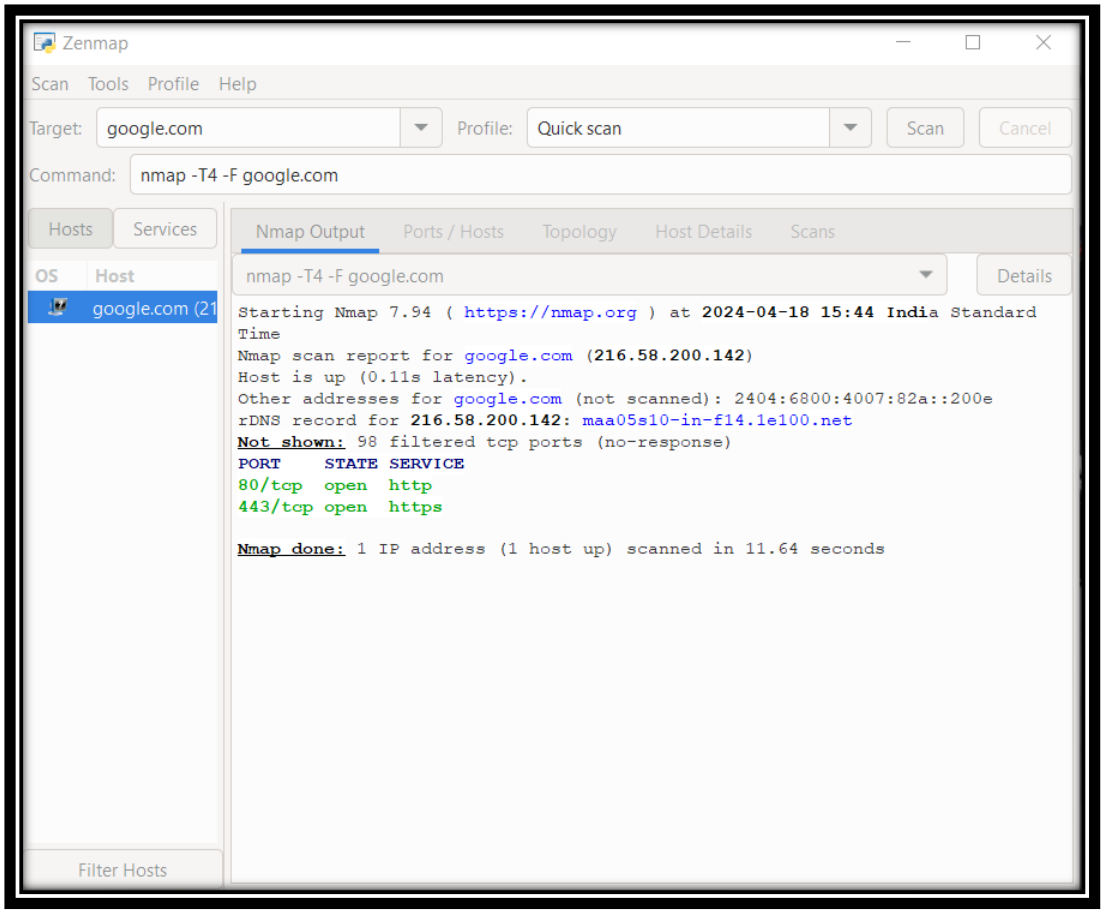


Figure 5 Zenmap Console

2.3 NIKTO

Nikto is an open-source web server scanner used for detecting and assessing potential vulnerabilities in web servers and applications. It performs comprehensive tests on web servers, checking for misconfigurations, outdated software, and common security flaws. Nikto scans for over 6700 potentially dangerous files or programs on web servers, including outdated server software versions and known security issues. It's frequently utilized by security professionals and penetration testers to identify weaknesses in web server configurations and applications. Nikto's straightforward command-line interface and extensive scanning capabilities make it a valuable tool for web security assessments and audits.

Figure 6 Nikto Help

Figure 7 Nikto Help

1. Functionality: Nikto scans web servers to uncover vulnerabilities, misconfigurations, and outdated software.

2. Methodology: It sends HTTP requests to the server, analyzing responses to identify potential issues across files, configurations, and known vulnerabilities.

3. Detection: Nikto checks for outdated software, misconfigured settings, default files/directories, and known vulnerabilities in web applications.

4. Customization: Users can tailor scans by specifying hosts, ports, HTTP methods, SSL settings, and authentication credentials.

2.4 FFUF

FFUF (Fuzz Faster U Fool) is an open-source web fuzzer designed for efficient and fast web content discovery and directory brute-forcing. It's known for its speed and flexibility, allowing users to customize their fuzzing payloads and options extensively. FFUF supports various fuzzing techniques, including wordlists, recursion, and filtering, enabling thorough enumeration of web resources. It's commonly used for web application security testing, particularly in identifying hidden directories, files, and endpoints. With its command-line interface and robust features, FFUF is a popular choice among security professionals and penetration testers for reconnaissance and vulnerability discovery.

Figure 8 Ffuf Help

2.5 ACUNETIX

Acunetix is a leading web vulnerability scanner used for identifying security flaws in web applications. It offers automated scanning capabilities to detect a wide range of vulnerabilities, including SQL injection, cross-site scripting (XSS), and insecure server configurations. Acunetix provides detailed reports and remediation recommendations, facilitating efficient vulnerability management. Its advanced scanning technology ensures comprehensive coverage and accuracy in detecting web application vulnerabilities. Trusted by security professionals and organizations worldwide, Acunetix plays a crucial role in ensuring the security and integrity of web applications.

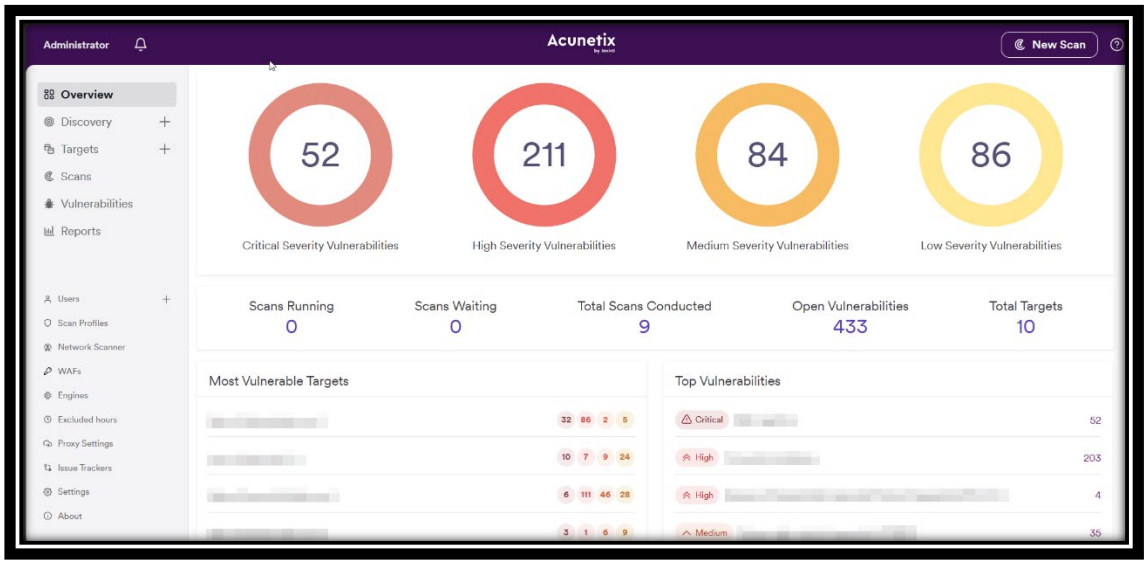


Figure 9 Acunetix Dashboard

Figure 10 Acunetix Dashboard

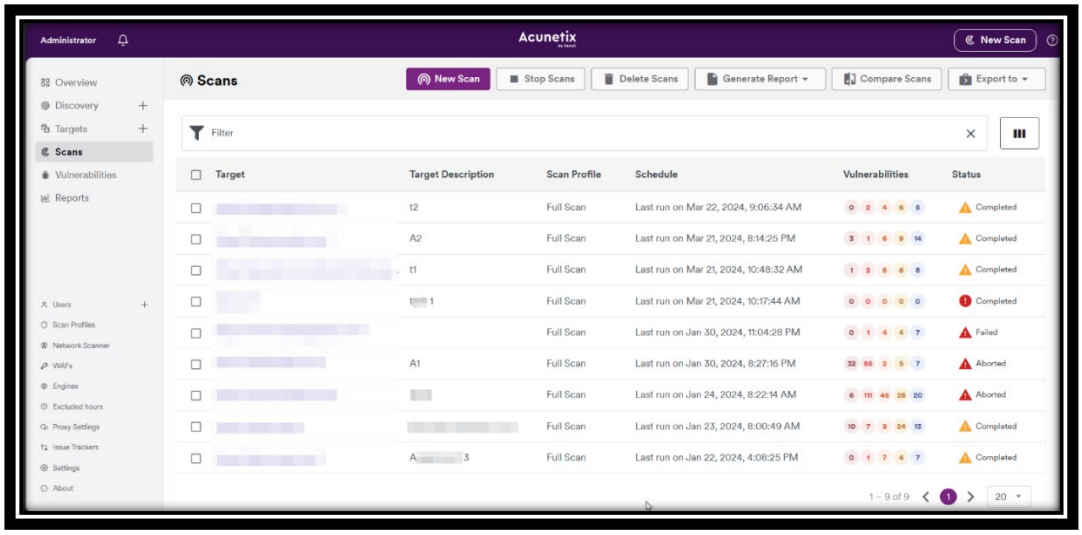


Figure 11 Acunetix Scan Results

2.6 NESSUS

Nessus is a widely-used vulnerability assessment tool that helps identify security issues in networks and systems. It conducts thorough scans to pinpoint vulnerabilities like misconfigurations, outdated software, and weak passwords. Nessus offers a wide range of scanning options, including credentialed and non-credentialed scans, enabling comprehensive assessments. It provides detailed reports with prioritized remediation steps, aiding in effective risk mitigation. Renowned for its accuracy and versatility, Nessus is a cornerstone tool for cybersecurity professionals and organizations seeking to maintain a robust security posture.

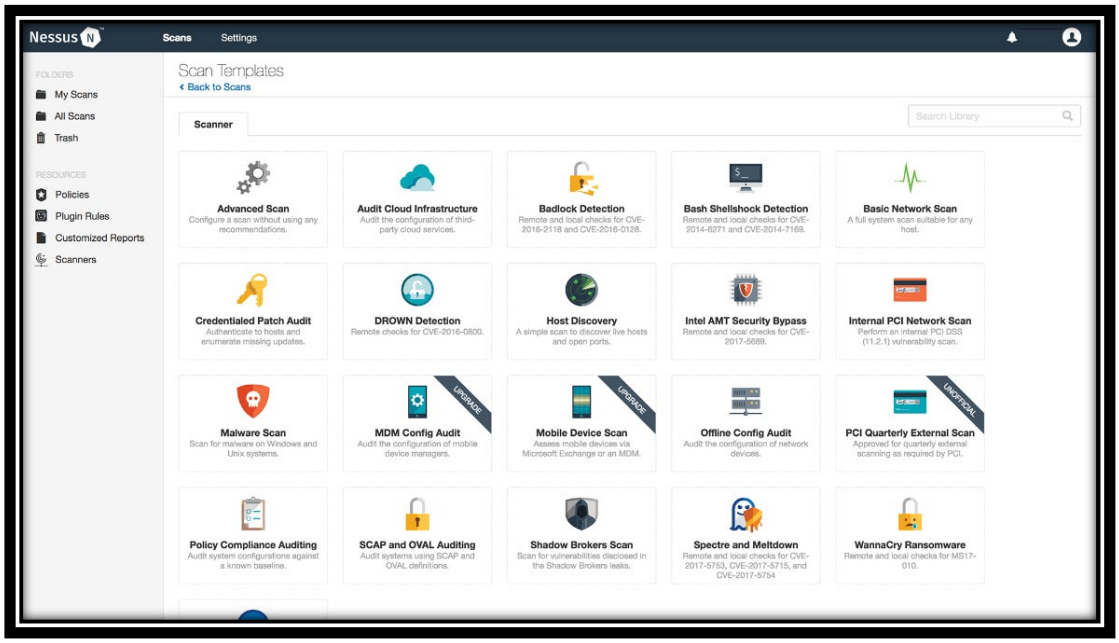


Figure 12 Nessus Scan Templates

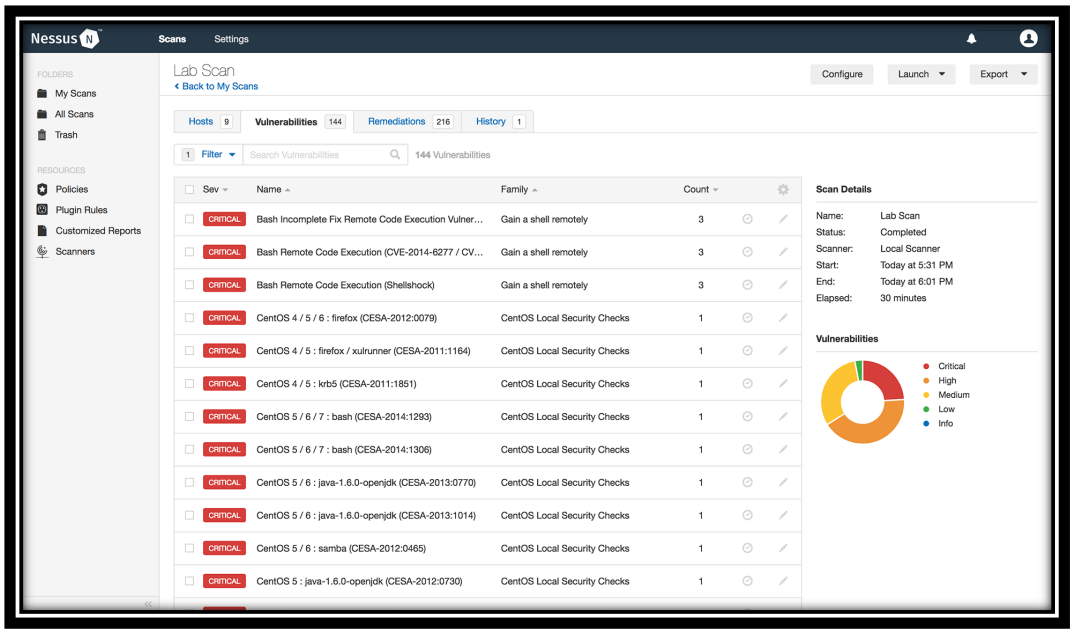


Figure 13 Nessus Scan Results

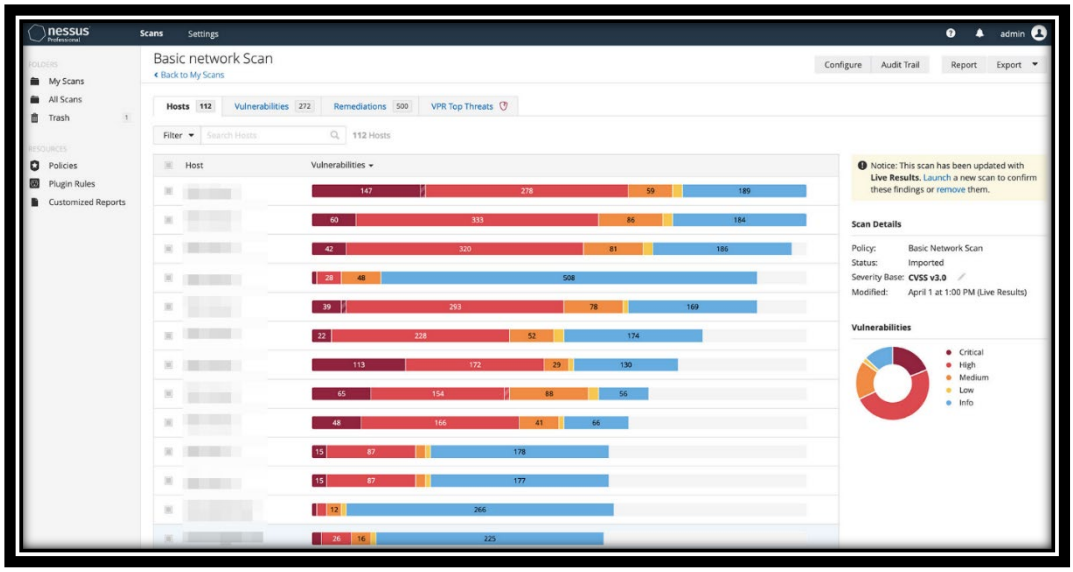


Figure 14 Nessus Basic Scan Result

2.7 BURP SUITE

Burp Suite is a comprehensive web application security testing tool used for manual and automated testing of web applications. It includes a variety of tools such as a proxy, scanner, intruder, repeater, and sequencer, catering to different stages of the security testing process. Burp Suite helps identify security vulnerabilities like SQL injection, cross-site scripting (XSS), and broken authentication. It offers features for intercepting and modifying HTTP/S requests and responses, making it ideal for both analysis and exploitation. Widely favored by security professionals and ethical hackers, Burp Suite provides a powerful and user-friendly platform for securing web applications.

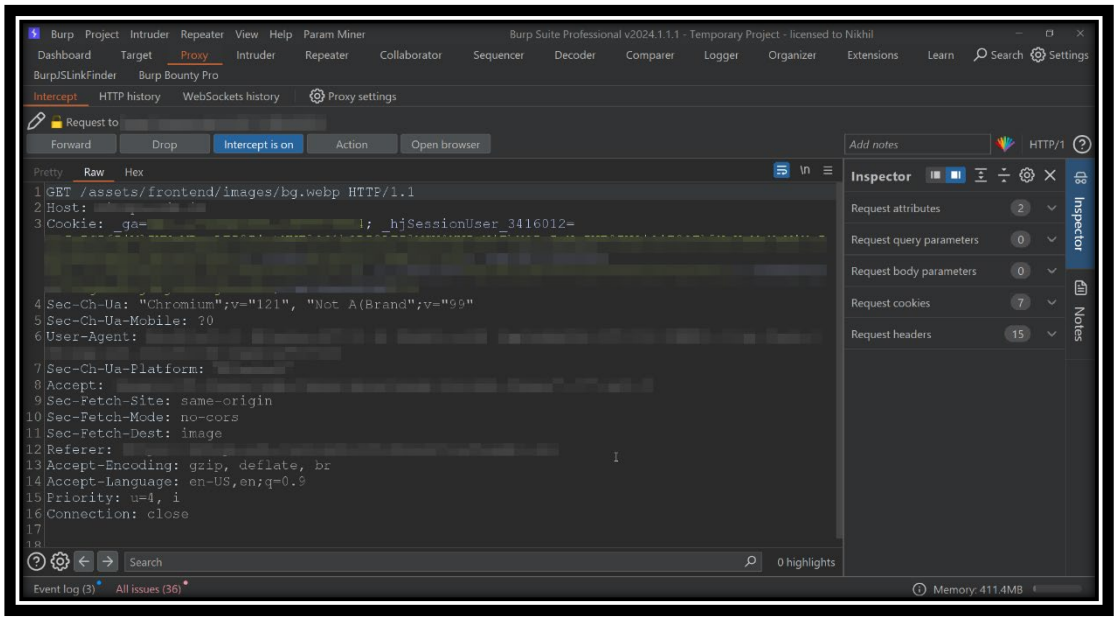


Figure 15 Burp Suite Proxy Tab

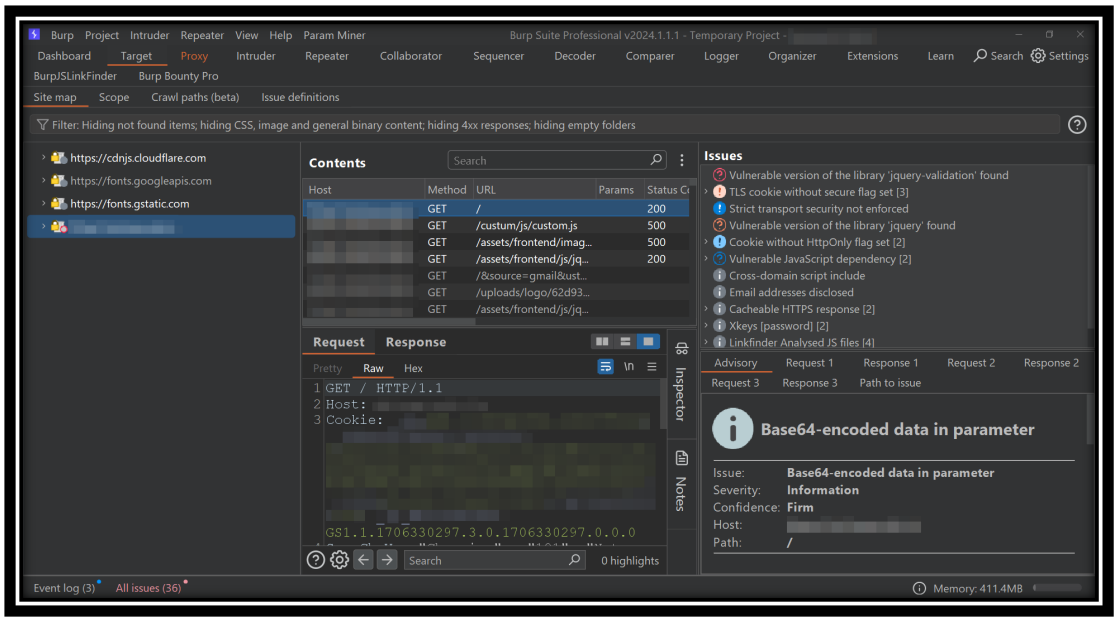


Figure 16 Burp Suite Target Tab

2.8 SUBLISTER

Sublist3r is a reconnaissance tool designed for subdomain enumeration through various search engines and services. It leverages multiple sources to discover subdomains associated with a target domain, aiding in reconnaissance and footprinting. Sublist3r supports both passive and active subdomain discovery techniques, enhancing its effectiveness in uncovering hidden assets. It's commonly used by security professionals and bug bounty hunters to expand attack surfaces and identify potential vulnerabilities. With its simplicity and efficiency, Sublist3r streamlines the process of subdomain enumeration for security assessments and penetration testing.

Figure 17 Sublister Console

2.9 WAPPALYZER

Wappalyzer is a browser extension and web application that identifies the technologies used on websites. It detects web frameworks, content management systems (CMS), server software, analytics tools, and more, providing insights into a site's technology stack. Wappalyzer helps users analyze competitor websites, understand market trends, and gather intelligence for web development and marketing strategies. It's widely used by developers, marketers, and security professionals to gain visibility into the technologies powering websites. With its user-friendly interface and extensive database of detected technologies, Wappalyzer offers valuable insights into the web ecosystem.

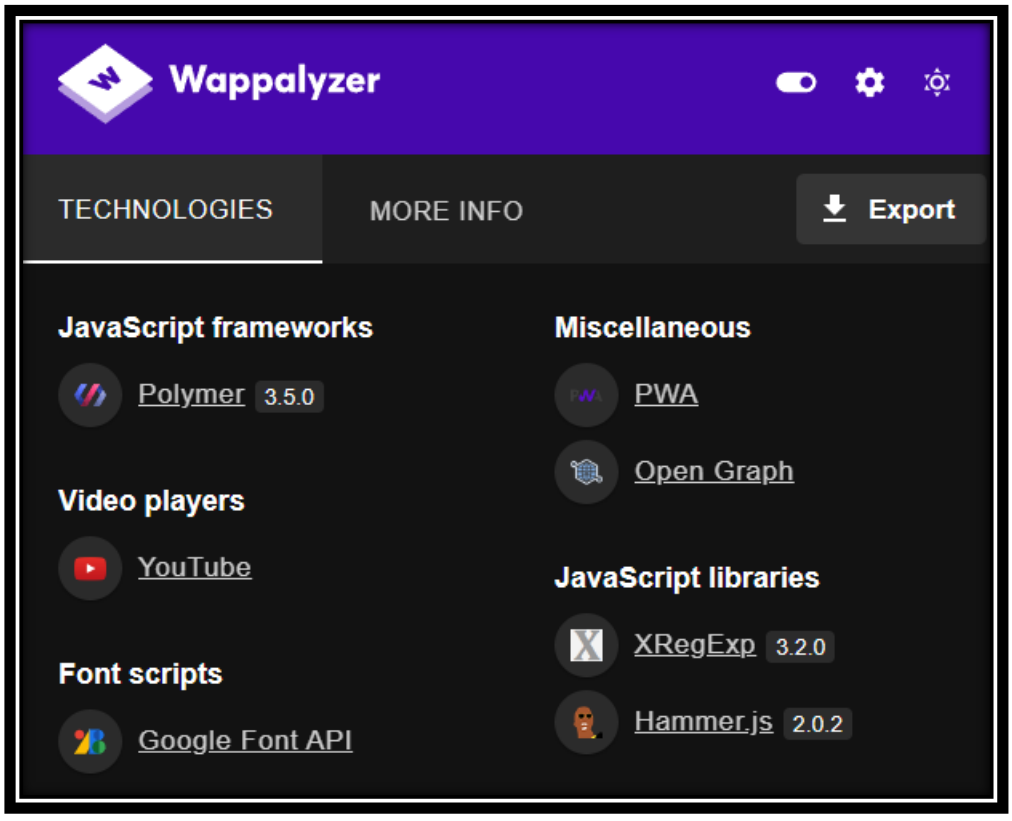


Figure 18 Wappalyzer UI

2.10 WIRESHARK

Wireshark is a powerful network protocol analyzer used for troubleshooting, analysis, and development of communication protocols. It captures and displays network packets in real-time, allowing users to inspect the contents of data being transmitted over a network. Wireshark supports a wide range of protocols and provides detailed packet-level information, making it invaluable for network administrators and security professionals. It offers features like packet filtering, packet decryption, and protocol dissection, enabling deep inspection of network traffic. Widely regarded as the standard tool for network analysis, Wireshark is essential for diagnosing network issues and investigating security incidents.

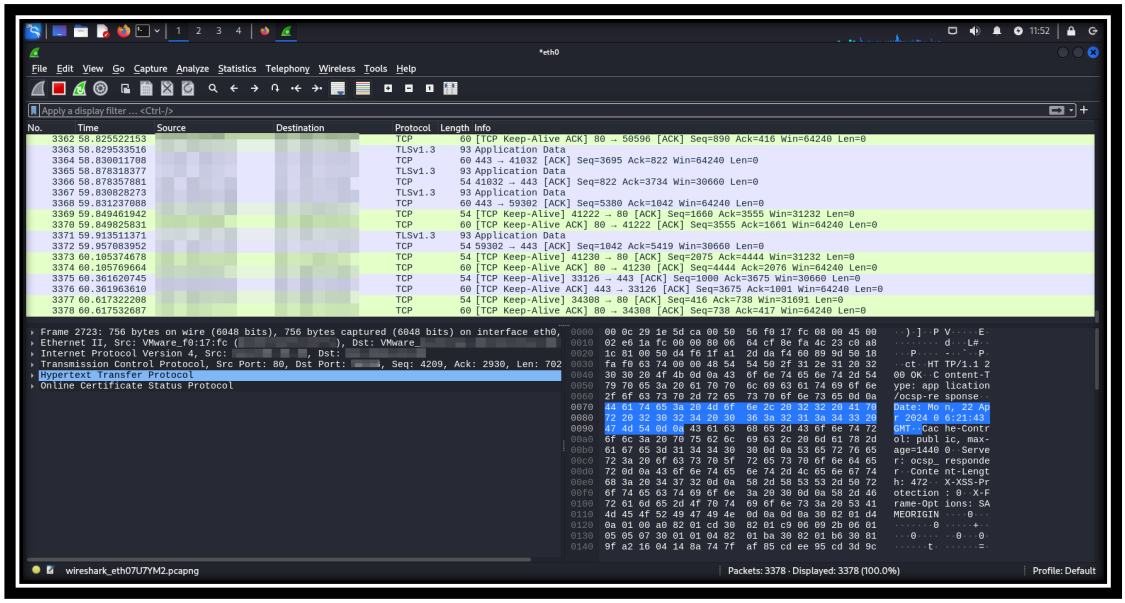


Figure 19 Wireshark Packet Capture Interface

2.11 METASPLOIT

Metasploit is an open-source penetration testing framework used for developing and executing exploit code against remote targets. It provides a comprehensive suite of tools for vulnerability assessment, exploitation, and post-exploitation activities. Metasploit simplifies the process of identifying and exploiting security vulnerabilities in networks, systems, and applications. It offers a vast collection of pre-built exploits, payloads, and auxiliary modules, making it accessible to both beginners and advanced users. Widely adopted by security professionals and ethical hackers, Metasploit is an essential tool for assessing and improving the security posture of organizations.

Figure 20 Metasploit Main Menu

Figure 21 Metasploit Search and Use Section

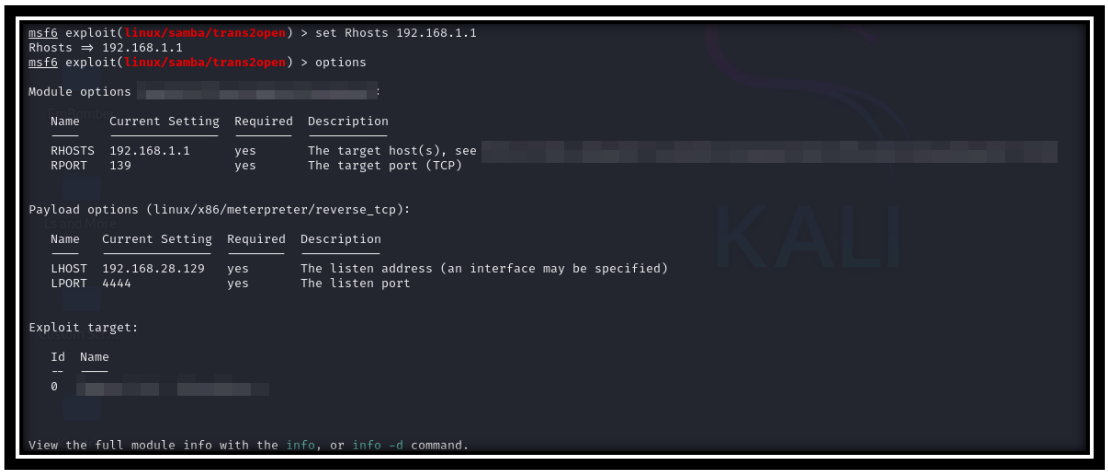


Figure 22 Metasploit Options Menu

2.12 HASHCAT

Hashcat is a powerful password recovery tool used for cracking hashed passwords through brute-force and dictionary attacks. It supports various hashing algorithms, including MD5, SHA-1, SHA-256, and bcrypt, among others. Hashcat utilizes the GPU to accelerate the password cracking process, enabling high-speed and efficient password recovery. It's commonly used by security professionals, penetration testers, and forensic analysts to assess password security and recover lost passwords. With its flexibility, speed, and wide range of supported hash types, Hashcat is a popular choice for password cracking tasks.

Figure 23 Hashcat Help Menu

2.13 JOHN THE RIPPER

John the Ripper is a widely-used password cracking tool designed to find weak passwords through brute-force and dictionary attacks. It supports various encryption algorithms and hash formats, making it versatile for cracking passwords stored in different systems and applications. John the Ripper is known for its speed and efficiency in recovering passwords, especially when combined with powerful hardware configurations. It's a popular choice for security professionals and penetration testers for assessing password security and performing security audits. With its command-line interface and extensive features, John the Ripper remains a staple tool in the field of password security.

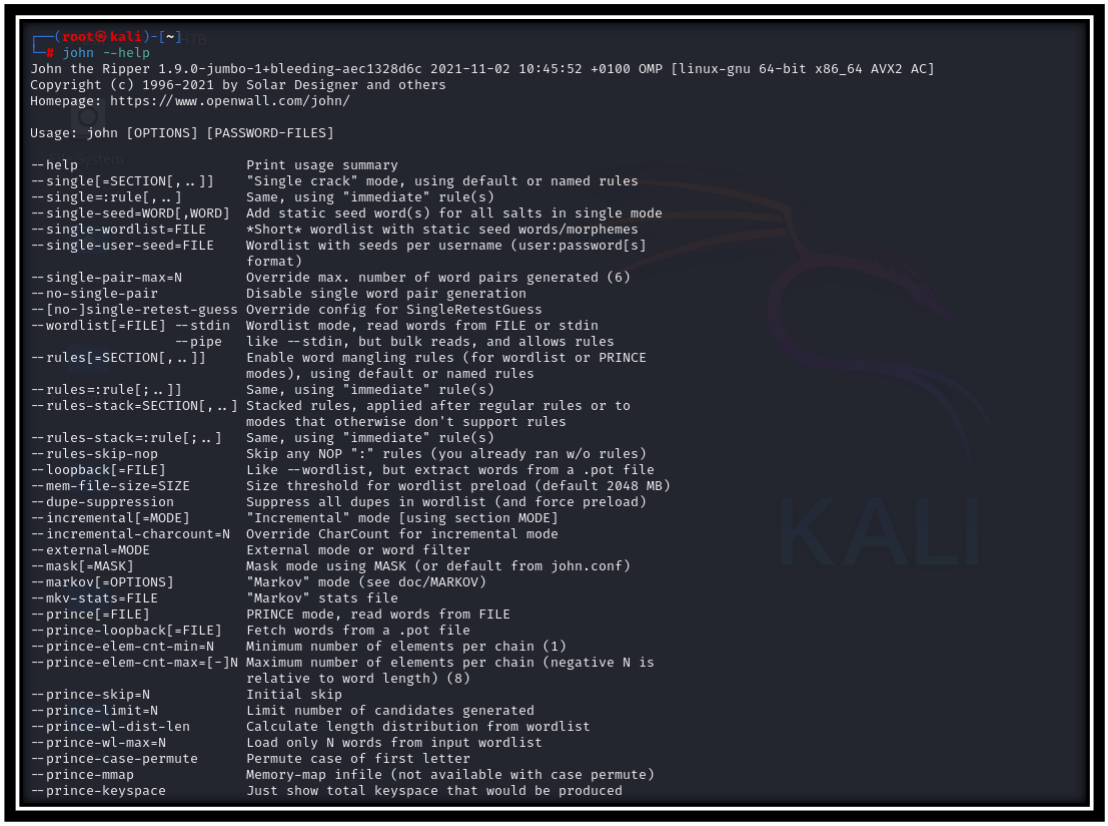


Figure 24 John the Ripper Help Menu

3. OWASP TOP 10

3.1 INTRODUCTION

The OWASP (Open Web Application Security Project) Top 10 is a regularly updated list of the most critical web application security risks. It's essentially a consensus-driven compilation of the most prevalent and severe security concerns facing web applications today. The OWASP community, composed of security professionals and experts from around the world, collaborates to identify, prioritize, and mitigate these risks.

The list serves as a guide for developers, security professionals, and organizations to understand and address common vulnerabilities that attackers frequently exploit. By focusing on these top 10 risks, developers can prioritize their efforts in securing their web applications effectively.

The OWASP Top 10 typically covers a range of issues, including injection flaws, broken authentication, sensitive data exposure, XML External Entities (XXE), broken access control, security misconfigurations, cross-site scripting (XSS), insecure deserialization, using components with known vulnerabilities, and insufficient logging and monitoring.

It's important to note that while the OWASP Top 10 provides valuable insights, it's not exhaustive, and developers should consider other security risks specific to their applications and environments. Additionally, the list is regularly updated to reflect emerging threats and changes in the cybersecurity landscape.

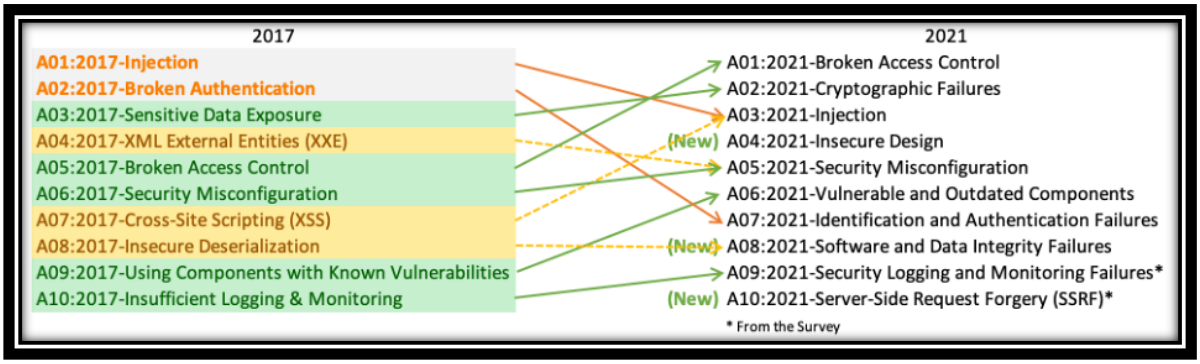


Figure 25 OWASP Top 10 2017 vs 2021

3.2 A01:2021 – Broken Access Control

Overview

Moving up from the fifth position, 94% of applications were tested for some form of broken access control with the average incidence rate of 3.81%, and has the most occurrences in the contributed dataset with over 318k. Notable Common Weakness Enumerations (CWEs) included are CWE-200: Exposure of Sensitive Information to an Unauthorized Actor, CWE-201: Insertion of Sensitive Information Into Sent Data, and CWE-352: Cross-Site Request Forgery.

Description

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits. Common access control vulnerabilities include:

- Violation of the principle of least privilege or deny by default, where access should only be granted for particular capabilities, roles, or users, but is available to anyone.
- Bypassing access control checks by modifying the URL (parameter tampering or force browsing), internal application state, or the HTML page, or by using an attack tool modifying API requests.
- Permitting viewing or editing someone else's account, by providing its unique identifier (insecure direct object references)
- Accessing API with missing access controls for POST, PUT and DELETE.
- Elevation of privilege. Acting as a user without being logged in or acting as an admin when logged in as a user.
- Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT) access control token, or a cookie or hidden field manipulated to elevate privileges or abusing JWT invalidation.

- CORS misconfiguration allows API access from unauthorized/untrusted origins.
- Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user.

3.3 A02:2021-Cryptographic Failures

Overview

Shifting up one position to #2, previously known as Sensitive Data Exposure, which is more of a broad symptom rather than a root cause, the focus is on failures related to cryptography (or lack thereof). Which often lead to exposure of sensitive data. Notable Common Weakness Enumerations (CWEs) included are CWE-259: Use of Hard-coded Password, CWE-327: Broken or Risky Crypto Algorithm, and CWE-331 Insufficient Entropy.

Description

The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information, and business secrets require extra protection, mainly if that data falls under privacy laws, e.g., EU's General Data Protection Regulation (GDPR), or regulations, e.g., financial data protection such as PCI Data Security Standard (PCI DSS). For all such data:

Is any data transmitted in clear text? This concerns protocols such as HTTP, SMTP, FTP also using TLS upgrades like STARTTLS. External internet traffic is hazardous. Verify all internal traffic, e.g., between load balancers, web servers, or back-end systems.

- Are any old or weak cryptographic algorithms or protocols used either by default or in older code?
- Are default crypto keys in use, weak crypto keys generated or re-used, or is proper key management or rotation missing? Are crypto keys checked into source code repositories?
- Is encryption not enforced, e.g., are any HTTP headers (browser) security directives or headers missing?
- Is the received server certificate and the trust chain properly validated?
- Are initialization vectors ignored, reused, or not generated sufficiently secure for the cryptographic mode of operation? Is an insecure mode of operation such as ECB in use? Is encryption used when authenticated encryption is more appropriate?

- Are passwords being used as cryptographic keys in absence of a password base key derivation function?
- Is randomness used for cryptographic purposes that was not designed to meet cryptographic requirements? Even if the correct function is chosen, does it need to be seeded by the developer, and if not, has the developer over-written the strong seeding functionality built into it with a seed that lacks sufficient entropy/unpredictability?
- Are deprecated hash functions such as MD5 or SHA1 in use, or are non-cryptographic hash functions used when cryptographic hash functions are needed?
- Are deprecated cryptographic padding methods such as PKCS number 1 v1.5 in use?
- Are cryptographic error messages or side channel information exploitable, for example in the form of padding oracle attacks?

3.4 A03:2021-Injection

Overview

Injection slides down to the third position. 94% of the applications were tested for some form of injection with a max incidence rate of 19%, an average incidence rate of 3%, and 274k occurrences. Notable Common Weakness Enumerations (CWEs) included are CWE-79: Cross-site Scripting, CWE-89: SQL Injection, and CWE-73: External Control of File Name or Path.

Description

- An application is vulnerable to attack when:
- User-supplied data is not validated, filtered, or sanitized by the application.
- Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
- Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.
- Hostile data is directly used or concatenated. The SQL or command contains the structure and malicious data in dynamic queries, commands, or stored procedures.

Some of the more common injections are SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection. The concept is identical among all interpreters. Source code review is the best method of detecting if applications are vulnerable to injections. Automated testing of all parameters, headers, URL, cookies, JSON, SOAP, and XML data inputs is strongly encouraged. Organizations can include static (SAST), dynamic (DAST), and interactive (IAST) application security testing tools into the CI/CD pipeline to identify introduced injection flaws before production deployment.

3.5 A04:2021-Insecure Design

Overview

A new category for 2021 focuses on risks related to design and architectural flaws, with a call for more use of threat modeling, secure design patterns, and reference architectures. As a community we need to move beyond "shift-left" in the coding space to pre-code activities that are critical for the principles of Secure by Design. Notable Common Weakness Enumerations (CWEs) include CWE-209: Generation of Error Message Containing Sensitive Information, CWE-256: Unprotected Storage of Credentials, CWE-501: Trust Boundary Violation, and CWE-522: Insufficiently Protected Credentials.

Description

Insecure design is a broad category representing different weaknesses, expressed as “missing or ineffective control design.” Insecure design is not the source for all other Top 10 risk categories. There is a difference between insecure design and insecure implementation. We differentiate between design flaws and implementation defects for a reason, they have different root causes and remediation. A secure design can still have implementation defects leading to vulnerabilities that may be exploited. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks. One of the factors that contribute to insecure design is the lack of business risk profiling inherent in the software or system being developed, and thus the failure to determine what level of security design is required.

Requirements and Resource Management

Collect and negotiate the business requirements for an application with the business, including the protection requirements concerning confidentiality, integrity, availability, and authenticity of all data assets and the expected business logic. Take into account how exposed your application will be and if you need segregation of tenants (additionally to access control). Compile the technical requirements, including functional and non-functional security requirements. Plan and negotiate the budget covering all design, build, testing, and operation, including security activities.

Secure Design

Secure design is a culture and methodology that constantly evaluates threats and ensures that code is robustly designed and tested to prevent known attack methods. Threat modeling should be integrated into refinement sessions (or similar activities); look for changes in data flows and access control or other security controls. In the user story development determine the correct flow and failure states, ensure they are well understood and agreed upon by responsible and impacted parties. Analyze assumptions and conditions for expected and failure flows, ensure they are still accurate and desirable. Determine how to validate the assumptions and enforce conditions needed for proper behaviors. Ensure the results are documented in the user story. Learn from mistakes and offer positive incentives to promote improvements. Secure design is neither an add-on nor a tool that you can add to software.

Secure Development Lifecycle

Secure software requires a secure development lifecycle, some form of secure design pattern, paved road methodology, secured component library, tooling, and threat modeling. Reach out for your security specialists at the beginning of a software project throughout the whole project and maintenance of your software. Consider leveraging the OWASP Software Assurance Maturity Model (SAMM) to help structure your secure software development efforts.

3.6 A05:2021-Security Misconfiguration

Overview

Moving up from #6 in the previous edition, 90% of applications were tested for some form of misconfiguration, with an average incidence rate of 4.%, and over 208k occurrences of a Common Weakness Enumeration (CWE) in this risk category. With more shifts into highly configurable software, it's not surprising to see this category move up. Notable CWEs included are CWE-16 Configuration and CWE-611 Improper Restriction of XML External Entity Reference.

Description

The application might be vulnerable if the application is:

- Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed (e.g., unnecessary ports, services, pages, accounts, or privileges).
- Default accounts and their passwords are still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, the latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks (e.g., Struts, Spring, ASP.NET), libraries, databases, etc., are not set to secure values.
- The server does not send security headers or directives, or they are not set to secure values.
- The software is out of date or vulnerable (see A06:2021-Vulnerable and Outdated Components).Without a concerted, repeatable application security configuration process, systems are at a higher risk**.**

3.7 A06:2021-Vulnerable and Outdated Components

Overview

It was #2 from the Top 10 community survey but also had enough data to make the Top 10 via data. Vulnerable Components are a known issue that we struggle to test and assess risk and is the only category to not have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploits/impact weight of 5.0 is used. Notable CWEs included are CWE-1104: Use of Unmaintained Third-Party Components and the two CWEs from Top 10 2013 and 2017.

Description

- You are likely vulnerable:
- If you do not know the versions of all components you use (both client-side and server-side). This includes components you directly use as well as nested dependencies.
- If the software is vulnerable, unsupported, or out of date. This includes the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries.
- If you do not scan for vulnerabilities regularly and subscribe to security bulletins related to the components you use.
- If you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion. This commonly happens in environments when patching is a monthly or quarterly task under change control, leaving organizations open to days or months of unnecessary exposure to fixed vulnerabilities.
- If software developers do not test the compatibility of updated, upgraded, or patched libraries.
- If you do not secure the components' configurations (see A05:2021-Security Misconfiguration).

3.8 A07:2021-Identification and Authentication Failures

Overview

Previously known as Broken Authentication, this category slid down from the second position and now includes Common Weakness Enumerations (CWEs) related to identification failures. Notable CWEs included are CWE-297: Improper Validation of Certificate with Host Mismatch, CWE-287: Improper Authentication, and CWE-384: Session Fixation.

Description

Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks. There may be authentication weaknesses if the application:

- Permits automated attacks such as credential stuffing, where the attacker has a list of valid usernames and passwords.
- Permits brute force or other automated attacks.
- Permits default, weak, or well-known passwords, such as "Password1" or "admin/admin".
- Uses weak or ineffective credential recovery and forgot-password processes, such as "knowledge-based answers," which cannot be made safe.
- Uses plain text, encrypted, or weakly hashed passwords data stores (see A02:2021-Cryptographic Failures).
- Has missing or ineffective multi-factor authentication.
- Exposes session identifier in the URL.
- Reuse session identifier after successful login.
- Does not correctly invalidate Session IDs. User sessions or authentication tokens (mainly single sign-on (SSO) tokens) aren't properly invalidated during logout or a period of inactivity.

3.9 A08:2021-Software and Data Integrity Failures

Overview

A new category for 2021 focuses on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. One of the highest weighted impacts from Common Vulnerability and Exposures/Common Vulnerability Scoring System (CVE/CVSS) data. Notable Common Weakness Enumerations (CWEs) include CWE-829: Inclusion of Functionality from Untrusted Control Sphere, CWE-494: Download of Code Without Integrity Check, and CWE-502: Deserialization of Untrusted Data.

Description

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise. Lastly, many applications now include auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application. Attackers could potentially upload their own updates to be distributed and run on all installations. Another example is where objects or data are encoded or serialized into a structure that an attacker can see and modify is vulnerable to insecure deserialization.

3.10 A09:2021-Security Logging and Monitoring Failures

Overview

Security logging and monitoring came from the Top 10 community survey (#3), up slightly from the tenth position in the OWASP Top 10 2017. Logging and monitoring can be challenging to test, often involving interviews or asking if attacks were detected during a penetration test. There isn't much CVE/CVSS data for this category, but detecting and responding to breaches is critical. Still, it can be very impactful for accountability, visibility, incident alerting, and forensics. This category expands beyond CWE-778 Insufficient Logging to include CWE-117 Improper Output Neutralization for Logs, CWE-223 Omission of Security-relevant Information, and CWE-532 Insertion of Sensitive Information into Log File.

Description

Returning to the OWASP Top 10 2021, this category is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time:

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools (such as OWASP ZAP) do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time. You are vulnerable to information leakage by making logging and alerting events visible to a user or an attacker.

3.11 A10:2021-Server-Side Request Forgery

Overview

This category is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage and above-average Exploit and Impact potential ratings. As new entries are likely to be a single or small cluster of Common Weakness Enumerations (CWEs) for attention and awareness, the hope is that they are subject to focus and can be rolled into a larger category in a future edition.

Description

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing. Also, the severity of SSRF is becoming higher due to cloud services and the complexity of architectures.

4. TYPES OF ATTACKS

4.1 CROSS SITE SCRIPTING (XSS)

Overview

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. For more details on the different types of XSS flaws.

Description

Cross-Site Scripting (XSS) attacks occur when:

- Data enters a Web application through an untrusted source, most frequently a web request.
- The data is included in dynamic content that is sent to a web user without being validated for malicious content.
- The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash, or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data, like cookies or other session information, to the attacker, redirecting the victim to web content controlled by the attacker, or

performing other malicious operations on the user's machine under the guise of the vulnerable site.

Reflected and Stored XSS Attacks

XSS attacks can generally be categorized into two categories: reflected and stored. There is a third, much less well-known type of XSS attack called DOM Based XSS that is discussed separately here.

Reflected XSS Attacks

Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other website. When a user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site, the injected code travels to the vulnerable web site, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server. Reflected XSS is also sometimes referred to as Non-Persistent or Type-I XSS (the attack is carried out through a single request / response cycle).

Stored XSS Attacks

Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information. Stored XSS is also sometimes referred to as Persistent or Type-II XSS.

Blind Cross-site Scripting

Blind Cross-site Scripting is a form of persistent XSS. It generally occurs when the attacker's payload saved on the server and reflected back to the victim from the backend application. For example in feedback forms, an attacker can submit the malicious payload using the form, and once the backend user/admin of the application will open the attacker's submitted form via the backend application, the attacker's payload will get executed. Blind Cross-site Scripting is hard to confirm in the real-world scenario but one of the best tools for this is XSS Hunter.

Other Types of XSS Vulnerabilities

In addition to Stored and Reflected XSS, another type of XSS, DOM Based XSS was identified by Amit Klein in 2005. OWASP recommends the XSS categorization as described in the OWASP Article: Types of Cross-Site Scripting, which covers all these XSS terms, organizing them into a matrix of Stored vs. Reflected XSS and Server vs. Client XSS, where DOM Based XSS is a subset of Client XSS.

XSS Attack Consequences

The consequence of an XSS attack is the same regardless of whether it is stored or reflected (or DOM Based). The difference is in how the payload arrives at the server. Do not be fooled into thinking that a "read-only" or "brochureware" site is not vulnerable to serious reflected XSS attacks. XSS can cause a variety of problems for the end user that range in severity from an annoyance to complete account compromise. The most severe XSS attacks involve disclosure of the user's session cookie, allowing an attacker to hijack the user's session and take over the account. Other damaging attacks include the disclosure of end user files, installation of Trojan horse programs, redirecting the user to some other page or site, or modifying presentation of content. An XSS vulnerability allowing an attacker to modify a press release or news item could affect a company's stock price or lessen consumer confidence. An XSS vulnerability on a pharmaceutical site could allow an attacker to modify dosage information resulting in an overdose.

How to Determine If You Are Vulnerable:

XSS flaws can be difficult to identify and remove from a web application. The best way to find flaws is to perform a security review of the code and search for all places where input from an HTTP request could possibly make its way into the HTML output. Note that a variety of different HTML tags can be used to transmit a malicious JavaScript. Nessus, Nikto, and some other available tools can help scan a website for these flaws, but can only scratch the surface. If one part of a website is vulnerable, there is a high likelihood that there are other problems as well.

How to Protect Yourself

The primary defenses against XSS are described in the OWASP XSS Prevention Cheat Sheet.

Also, it's crucial that you turn off HTTP TRACE support on all web servers. An attacker can steal cookie data via Javascript even when document.cookie is disabled or not supported by the client. This attack is mounted when a user posts a malicious script to a forum so when another user clicks the link, an asynchronous HTTP Trace call is triggered which collects the user's cookie information from the server, and then sends it over to another malicious server that collects the cookie information so the attacker can mount a session hijack attack. This is easily mitigated by removing support for HTTP TRACE on all web servers.

The OWASP ESAPI project has produced a set of reusable security components in several languages, including validation and escaping routines to prevent parameter tampering and the injection of XSS attacks. In addition, the OWASP WebGoat Project training application has lessons on Cross-Site Scripting and data encoding.

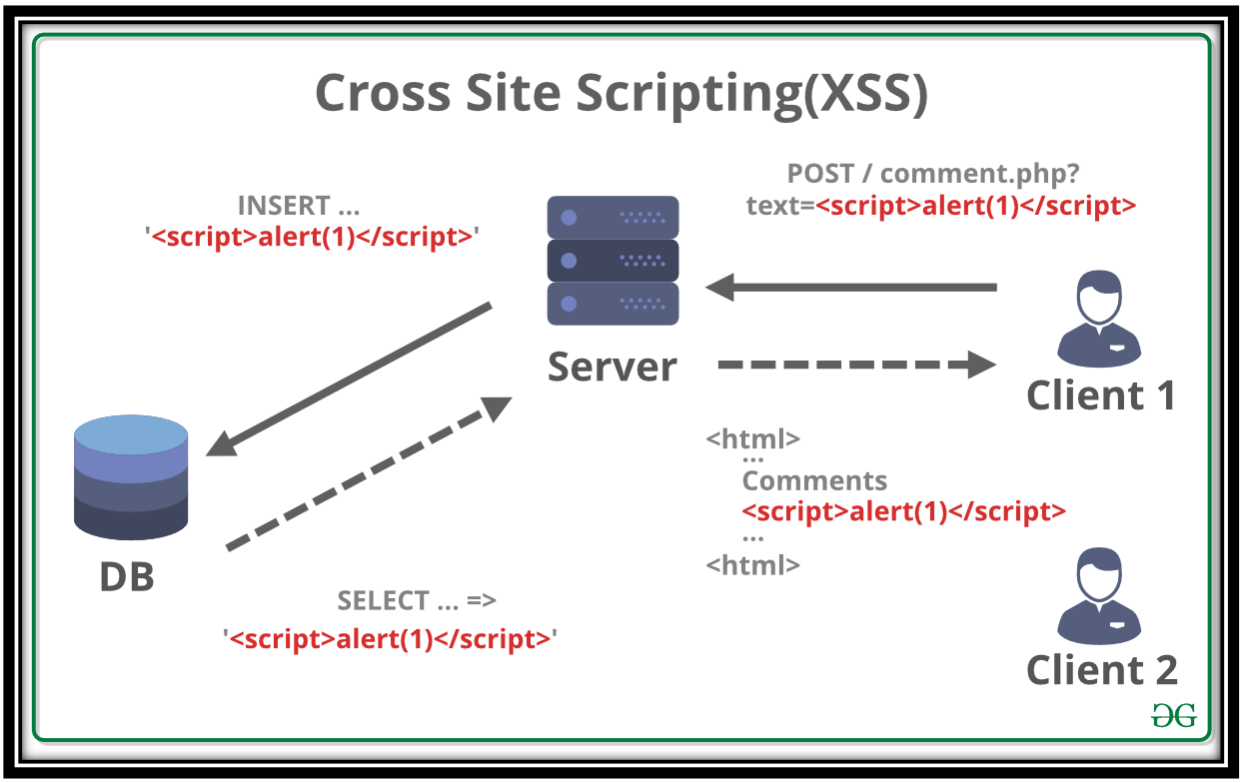


Figure 26 Cross Site Scripting Overview

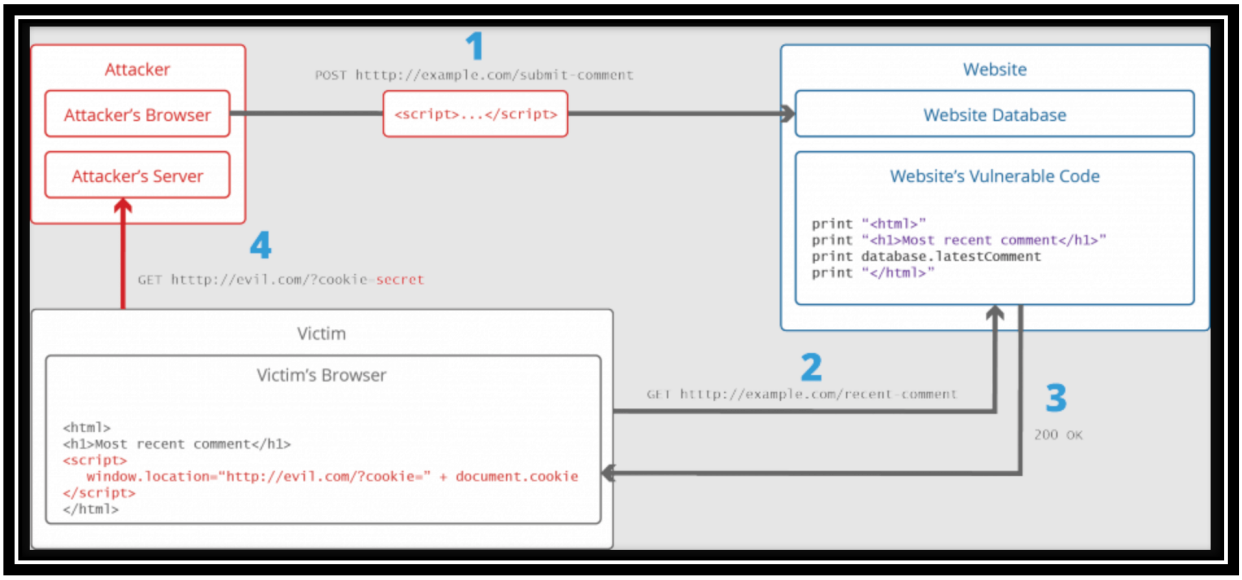


Figure 27 Cross Site Scripting Attack Flow

4.2 SQL INJECTION

Overview

A SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

Threat Modeling

- SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.
- SQL Injection is very common with PHP and ASP applications due to the prevalence of older functional interfaces. Due to the nature of programmatic interfaces available, J2EE and ASP.NET applications are less likely to have easily exploited SQL injections.
- The severity of SQL Injection attacks is limited by the attacker’s skill and imagination, and to a lesser extent, defense in depth countermeasures, such as low privilege connections to the database server and so on. In general, consider SQL Injection a high impact severity.

Description

SQL injection attack occurs when:

- An unintended data enters a program from an untrusted source.
- The data is used to dynamically construct a SQL query

The main consequences are:

- Confidentiality: Since SQL databases generally hold sensitive data, loss of confidentiality is a frequent problem with SQL Injection vulnerabilities.
- Authentication: If poor SQL commands are used to check user names and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password.
- Authorization: If authorization information is held in a SQL database, it may be possible to change this information through the successful exploitation of a SQL Injection vulnerability.
- Integrity: Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL Injection attack.

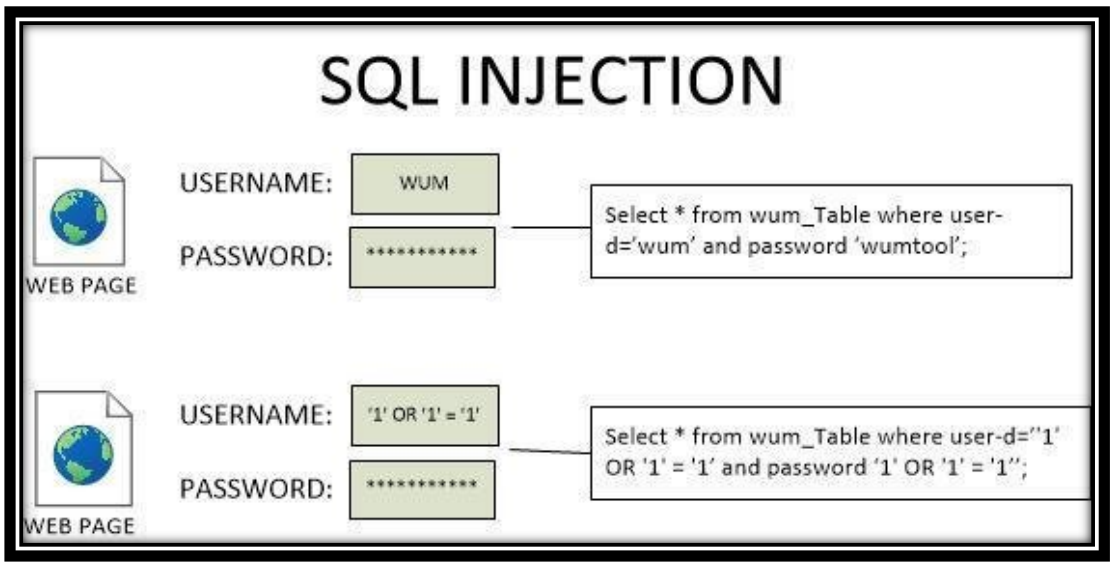


Figure 28 SQL Injection Attack

4.3 HOST HEADER INJECTION

Overview

Host Header Injection is an attack that exploits the way web servers and applications handle the Host header in HTTP requests. The Host header is part of the HTTP/1.1 protocol and is used to specify the domain name of the server (virtual host) that the client wants to connect to.When a client sends an HTTP request to a web server, it includes a Host header in the request that specifies the domain name of the server that it wants to connect to. For example, if a user types "www.example.com" into their browser, the browser will send an HTTP request with the Host header set to "www.example.com".

Description

There are several ways that an attacker can perform a Host Header Injection attack. Here are a few common methods:

- Malformed Host header value: An attacker can inject a malformed Host header value in the HTTP request. For example, they can include a newline character in the header value to create a new header field.
- Multiple Host header values: An attacker can inject multiple Host header values in the HTTP request. The web server will usually only read the first Host header value, but some servers may read subsequent values as well.
- Spoofed Host header: An attacker can spoof the Host header in the HTTP request to make it look like the request is coming from a different domain. For example, they can set the Host header to a subdomain of the target domain.

Once the attacker has successfully injected a malicious Host header value, they can perform a range of attacks, including:

- Cross-site scripting (XSS) attacks
- Cross-site request forgery (CSRF) attacks
- Session hijacking attacks

- Password reset attacks
- Access control bypass attacks

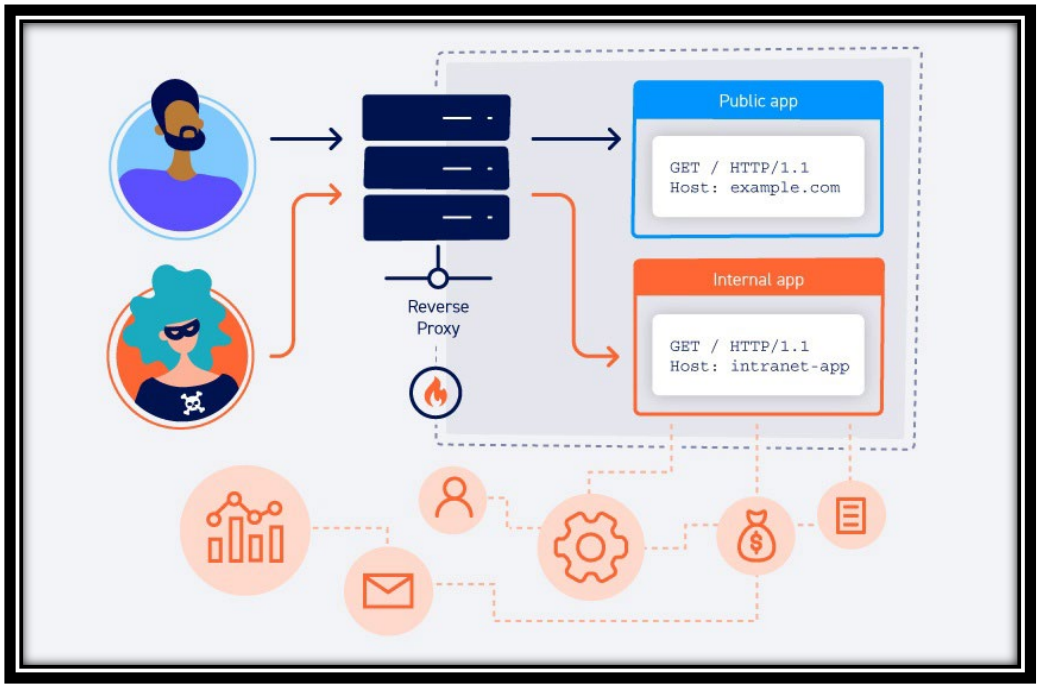


Figure 29 Host Header Injection Attack Flow

4.4 NO RATE LIMITING

Overview

Broadly, rate limiting is a method of preventing a user (human or bot) from repeating an action in quick succession too many times (sometimes with the intent of causing damage). It sees a broad range of applications, from preventing DoS attacks at the proxy level to locking accounts to prevent brute-force attacks. While it can be admittedly annoying at times, an application without any form of rate limiting is begging to be targeted. And if you really want to avoid annoyance, you should be using a password manager to prevent typos.

Description

Based on this principle, we believe that any web application will, sooner or later, be confronted with a traffic-generating attack. These can take several forms, but the main ones are the following:

- Mass DoS (denial of service) attacks, which consist of generating a huge number of requests in order to overload the server and thus prevent it from processing legitimate requests.
- More targeted denial of service attacks, consisting of repeatedly calling resource-intensive endpoints and functions of the application with the same goal as the previous attack.
- Brute force attacks, consisting of attempting numerous login/password combinations in order to obtain a session on the platform and thus steal an account.
- Enumeration attacks, consisting, as the name suggests, of enumerating the platform's resources by attempting to iterate over them. This can be a list of IDs, file names, usernames, etc.

All these attacks can be mitigated by implementing an essential process that limits the amount of requests a client can send to the server: rate limiting.

4.5 FILE INCLUSION

Overview

File inclusion in cybersecurity refers to a vulnerability where an application dynamically includes files based on user input, leading to potential security risks. It commonly occurs in web applications and can manifest as either local file inclusion (LFI) or remote file inclusion (RFI). Attackers exploit file inclusion vulnerabilities to execute malicious code, disclose sensitive information, or gain unauthorized access to the system. Proper input validation and sanitization are crucial for mitigating file inclusion vulnerabilities. Web application firewalls (WAFs) and security testing tools can help detect and prevent file inclusion attacks, enhancing overall cybersecurity posture.

Description

Types of file inclusion vulnerabilities

File inclusion vulnerabilities come in two types, depending on the origin of the included file:

- Local File Inclusion (LFI)

```
<?php
$user_profile = $_GET['profile'];
include($user_profile . '.html');
?>
```

Figure 30 LFI Syntax Example

- Remote File Inclusion (RFI)

```
<?php
$templateUrl = $_POST['template_url'];
// Fetch and display the custom template
include($templateUrl);
?>
```

Figure 31 RFI Syntax Example

Local File Inclusion (LFI)

- A Local File Inclusion attack is used to trick the application into exposing or running files on the server. They allow attackers to execute arbitrary commands or, if the server is misconfigured and running with high privileges, to gain access to sensitive data.
- These attacks typically occur when an application uses the path to a file as input. If the application treats that input as trusted, an attacker can use the local file in an include statement.
- While Local File Inclusion and Remote File Inclusion are very similar, an attacker using LFI may include only local files.

Remote File Inclusion (RFI)

- An attacker who uses Remote File Inclusion targets web applications that dynamically reference external scripts. The goal of the attacker is to exploit the referencing function in the target application and to upload malware from a remote URL, located on a different domain.
- The results of a successful RFI attack can be information theft, a compromised server and a site takeover, resulting in content modification.

5. INTERNSHIP WORK

5.1 TASK #1 Scanning A Wordpress Website

1. Preparation:

- Obtain explicit permission from the website owner or responsible party to conduct the penetration test, ensuring compliance with legal and ethical considerations.
- Set up a testing environment, ideally a local or isolated environment, to prevent any accidental damage to the live website during the testing process.
- Ensure that you have all the necessary tools installed and configured, including web vulnerability scanners like WPScan, to effectively assess the security of the WordPress installation.
- Familiarize yourself with the scope of the penetration test, understanding which components of the WordPress site are included and any specific areas of focus.
- Prepare a plan or checklist outlining the steps you'll take during the testing process, including how to handle any identified vulnerabilities.

2. Install WPScan:

- WPScan can be installed on various operating systems, including Linux, macOS, and Windows, typically via the command line.
- Visit the official WPScan GitHub repository to access the installation instructions and download the necessary files.
- Follow the step-by-step instructions provided to install WPScan on your system, ensuring that any dependencies are also met.
- Verify the installation by running a simple command to check if WPScan is successfully installed and accessible from the command line.
- Once installed, update WPScan regularly to ensure that you have the latest version with the most up-to-date vulnerability information.

3. Perform Basic Scanning:

- Initiate a basic scan using WPScan by specifying the target WordPress website's URL as the command-line argument.
- WPScan will start scanning the target site, attempting to identify common vulnerabilities and configuration issues.
- Monitor the scan progress to ensure that it completes successfully without any errors or interruptions.
- After the scan is complete, review the basic scan results to get an initial overview of the WordPress site's security posture.
- Note any significant findings or anomalies that may require further investigation or analysis.

4. Enumerate Users:

- Utilize WPScan's user enumeration feature to gather information about the WordPress users registered on the target site.
- This process involves systematically querying the site's login page and analyzing the server responses to identify valid user accounts.
- WPScan will attempt various techniques, such as brute force and username enumeration, to enumerate existing users.
- The output will include a list of discovered usernames, which can be useful for further analysis, such as password cracking or targeted attacks.
- Use this information responsibly and ethically, avoiding any unauthorized access or misuse of user accounts.

5. Enumerate Plugins and Themes:

- Execute a plugin and theme enumeration scan using WPScan to identify the installed plugins and themes on the WordPress site.
- WPScan will analyze the site's HTML and CSS code to extract information about active plugins and themes, including their versions.
- This step is crucial for identifying outdated or vulnerable plugins and themes that may pose security risks to the site.
- The scan results will provide a comprehensive list of detected plugins and themes, along with their respective versions, aiding in vulnerability assessment.
- Prioritize further investigation of plugins and themes with known vulnerabilities or outdated versions that may require immediate attention.

6. Scan for Vulnerabilities:

- Conduct a thorough vulnerability scan using WPScan to identify potential security weaknesses within the WordPress installation.
- WPScan utilizes its extensive database of known vulnerabilities to check the WordPress core, plugins, and themes for any reported security issues.
- The scan results will categorize vulnerabilities based on severity levels, helping prioritize remediation efforts.
- Pay close attention to critical vulnerabilities that could lead to unauthorized access, data breaches, or site compromise.
- Note that while WPScan automates the vulnerability scanning process, manual verification may be necessary to confirm the existence and impact of certain vulnerabilities.

7. Manual Inspection:

- Perform manual inspection of the WordPress site's codebase, configuration files, and custom scripts to uncover potential security flaws.
- Look for sensitive information exposure, such as hardcoded credentials or configuration settings, which could be exploited by attackers.
- Review file permissions to ensure that sensitive files and directories are adequately protected from unauthorized access.
- Investigate any custom functionalities or third-party integrations for vulnerabilities that may not be detected by automated scanning tools.
- Document any findings from the manual inspection process for inclusion in the penetration testing report.

8. Analyze Results:

- Analyze the results of the vulnerability scans, manual inspection, and any other testing performed during the penetration test.
- Prioritize vulnerabilities based on their severity, potential impact on the WordPress site's security, and likelihood of exploitation.
- Consider the context of the website and its intended use when assessing the significance of identified vulnerabilities.
- Consult with relevant stakeholders, such as website owners, administrators, or security teams, to discuss the findings and determine appropriate remediation steps.
- Document the analysis process and findings thoroughly to provide clear and actionable recommendations in the penetration testing report.

9. Exploitation (if allowed):

- If permitted by the rules of engagement and ethical considerations, attempt to exploit identified vulnerabilities in a controlled environment.
- Exercise caution to avoid causing harm to the target system or exposing sensitive data during exploitation attempts.
- Document the steps taken to exploit each vulnerability, including any successful or unsuccessful attempts.
- Use exploitation as a validation technique to confirm the severity and impact of identified vulnerabilities.
- Ensure that all exploitation activities are conducted responsibly and in accordance with ethical hacking guidelines.

10. Report Findings:

- Prepare a comprehensive penetration testing report detailing all findings from the assessment, including vulnerabilities, their severity ratings, and potential impact.
- Clearly communicate the risks associated with each identified vulnerability and provide recommendations for remediation.
- Include detailed information on the testing methodology, tools used, and any limitations or constraints encountered during the assessment.
- Present the report to the website owner or responsible parties, ensuring that it is easily understandable and actionable.
- Collaborate with stakeholders to address any questions or concerns raised by the findings and facilitate the implementation of recommended remediation measures.

Figure 32 WPScan Console

Figure 33 WPScan Console

5.2 TASK #2 Cracking PDF Password

1. Preparation:

- Before beginning the process, ensure that you have explicit permission to attempt to crack the password of the PDF file. Unauthorized access to someone else's files is illegal and unethical.
- Download and install Hashcat and John the Ripper on your system. These are powerful password cracking tools available for various operating systems.
- Obtain the password-protected PDF file that you intend to crack. It's important to emphasize that you should only proceed with cracking passwords on files you have legitimate access to.
- Make sure to have a clear understanding of the legal implications and ethical considerations surrounding password cracking.
- Always operate within the boundaries of the law and adhere to ethical hacking guidelines.

2. Extract Hash:

- Use a tool like `pdf2john`, which is part of John the Ripper, to extract the hash from the PDF file. The hash represents the encrypted form of the password.
- Run the `pdf2john` command followed by the path to the PDF file. This tool converts the hash into a format that Hashcat or John the Ripper can understand and process.
- The extracted hash is saved into a file, typically named `hash.txt`, for later use in the password cracking process.
- This step is crucial as it provides the necessary input for Hashcat or John the Ripper to attempt to crack the password.
- Always verify that you have successfully extracted the hash before proceeding to the next steps.

3. Choose Attack Mode:

- Hashcat and John the Ripper offer various attack modes to crack passwords, each with its own strengths and weaknesses.
- Consider factors such as the complexity of the password, available computational resources, and time constraints when choosing the attack mode.
- Common attack modes include dictionary attack, brute-force attack, and hybrid attack, among others.
- Research the characteristics of each attack mode and select the one that is most likely to succeed based on your specific scenario.
- The choice of attack mode significantly influences the effectiveness and efficiency of the password cracking process.

4. Create Wordlist:

- For dictionary and hybrid attacks, you'll need a wordlist containing potential passwords to try during the cracking process.
- Wordlists can be generated manually, downloaded from reputable sources, or obtained from previous data breaches (if legally and ethically permissible).
- Ensure that the wordlist encompasses a wide range of possibilities, including common passwords, dictionary words, and variations of alphanumeric combinations.
- The quality and comprehensiveness of the wordlist directly impact the success rate of the password cracking attempt.
- Continuously update and refine the wordlist to incorporate new password patterns and trends.

5. Run Hashcat or John the Ripper:

- Once you've chosen the attack mode and prepared the necessary resources, initiate the password cracking process by running Hashcat or John the Ripper.
- Specify the appropriate command-line options, including the attack mode, hash file (generated in the previous step), and wordlist (if applicable).
- Both tools will systematically try each password from the wordlist against the hashed password until a match is found or all possibilities are exhausted.
- Monitor the progress of the cracking process, paying attention to metrics such as speed, estimated time to completion, and number of password attempts made.
- Adjust the cracking parameters or switch to alternative attack modes if necessary based on the observed progress and results.

6. Monitor Progress:

- As the password cracking process proceeds, monitor its progress closely to gauge its effectiveness.
- Keep an eye on metrics such as the number of password attempts per second, estimated time to completion, and any errors encountered during the process.
- Depending on the complexity of the password and available computational resources, the cracking process may take some time to complete.
- Periodically check the status of the cracking process to ensure it's running smoothly and making progress towards finding the correct password.
- Be prepared to adjust the cracking parameters or strategy if the progress is slower than expected or if unforeseen challenges arise.

7. Review Results:

- Once the password cracking process completes, review the results to determine if the password has been successfully cracked.
- If successful, the cracked password will be displayed in the output alongside the corresponding hash.
- Analyze the cracked passwords to identify patterns or weaknesses in password choices, which can inform future security practices.
- If the cracking process fails to find the correct password, carefully evaluate the reasons for the failure and consider alternative approaches or adjustments to improve the chances of success.
- Keep in mind that some passwords may be more resistant to cracking due to their complexity or randomness, requiring additional time and computational resources.

8. Post-Processing:

- After obtaining the cracked password, handle the information responsibly and ethically.
- If the password grants access to sensitive information, use it only for authorized purposes and avoid sharing it with unauthorized parties.
- Consider strengthening security measures to prevent similar vulnerabilities in the future, such as using stronger passwords or implementing multi-factor authentication.
- Document the password cracking process, including the tools used, parameters configured, and results obtained, for future reference or auditing purposes.
- Continuously evaluate and improve password security practices to mitigate the risk of unauthorized access and data breaches.

```
root@attackdefense:~# tools/JohnTheRipper/pdf2john.pl secret.pdf > hash
root@attackdefense:~# cat hash
secret.pdf:$pdf$2*3*128*-3904*1*16*400fe7ec4c2ecbba360a1e4707085fd5*32*924ea8d9badcbd235a750efb0891f4830000000000000000000000000000000000*32*f340
ec3846be71e419c10ae059c0b3507a21440ae0cf17189c92755918e90561
root@attackdefense:~#
```

Figure 34 Extracting Hash Using John the Ripper

```
root@attackdefense:~# john --wordlist=/root/wordlists/1000000-password-seclists.txt hash
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
sopranos         (secret.pdf)
1g 0:00:00:00 DONE (2018-12-06 16:19) 3.846g/s 44415p/s 44415c/s 44415C/s sopranos
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@attackdefense:~#
```

Figure 35 Cracking Password using a Wordlist and John the Ripper

```
root@attackdefense:~# vim hash
root@attackdefense:~# cat hash
$pdf$2*3*128*-3904*1*16*400fe7ec4c2ecbba360a1e4707085fd5*32*924ea8d9badcbd235a750efb0891f4830000000000000000000000000000000000*32*f340ec3846be71e
419c10ae059c0b3507a21440ae0cf17189c92755918e90561
root@attackdefense:~#
```

Figure 36 Saving the Hash in a file named hash

```
$pdf$2*3*128*-3904*1*16*400fe7ec4c2ecbba360a1e4707085fd5*32*924ea8d9badcbd235a750efb0891f4830000000000000000000000000000000000*32*f340ec3846be71e
419c10ae059c0b3507a21440ae0cf17189c92755918e90561:sopranos

Session..........: hashcat
Status...........: Cracked
Hash.Type........: PDF 1.4 - 1.6 (Acrobat 5 - 8)
Hash.Target......: $pdf$2*3*128*-3904*1*16*400fe7ec4c2ecbba360a1e47070...e90561
Time.Started.....: Thu Dec  6 16:20:22 2018 (1 sec)
Time.Estimated...: Thu Dec  6 16:20:23 2018 (0 secs)
Guess.Base.......: File (wordlists/1000000-password-seclists.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    35352 H/s (121.43ms) @ Accel:80 Loops:8 Thr:64 Vec:8
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 40960/1000003 (4.10%)
Rejected.........: 0/40960 (0.00%)
Restore.Point....: 0/1000003 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:64-70
Candidates.#1....: 123456 -> taint
```

Figure 37 Cracking password using Hashcat

```
root@attackdefense:~# pdftotext -upw sopranos secret.pdf
root@attackdefense:~# ls -l
total 44
-rw-r--r-- 1 root root   293 Nov 25 15:54 README
-rw-r--r-- 1 root root   193 Dec  7 06:22 hash
-rw-r--r-- 1 root root 24319 Dec  6 12:14 secret.pdf
-rw-r--r-- 1 root root    41 Dec  7 06:23 secret.txt
drwxr-xr-x 1 root root  4096 Dec  6 12:12 tools
drwxr-xr-x 1 root root  4096 Dec  6 12:34 wordlists
root@attackdefense:~# cat secret.txt
Flag: 49cdf40206c7c06c49f8284f83dfd7a3


root@attackdefense:~#
```

Figure 38 Reading the Password from secret.txt file

6. Flowcharts

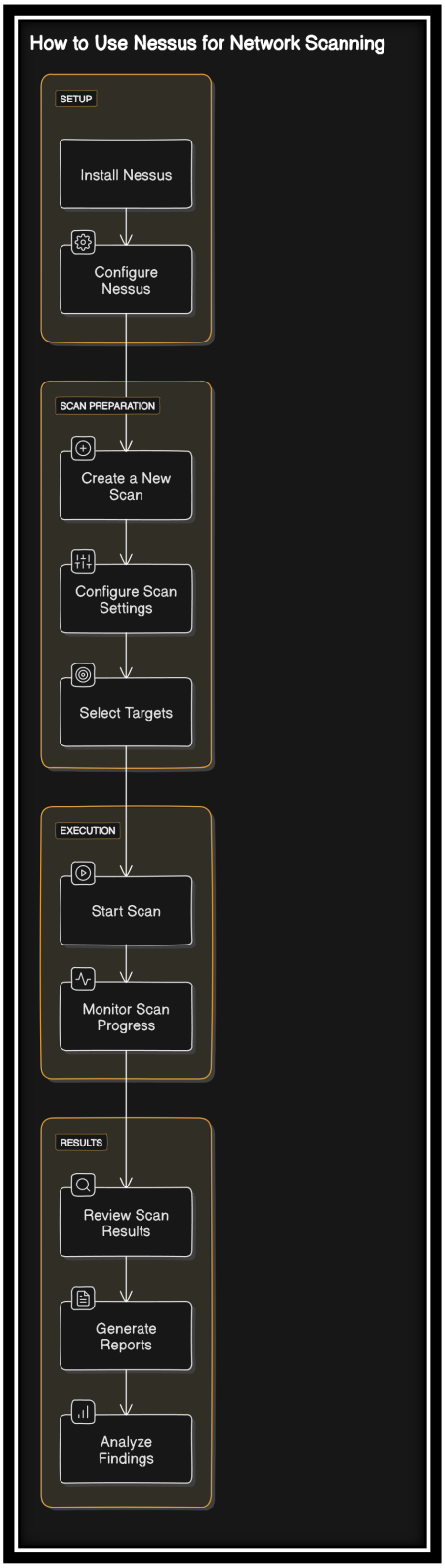


Figure 39 Nessus Application Flow

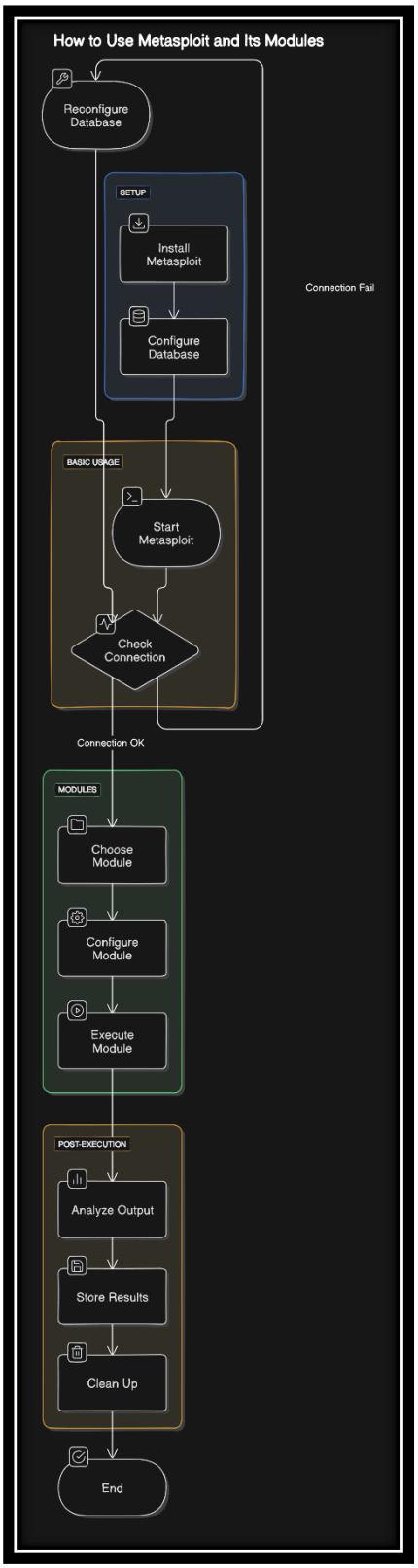


Figure 40 Metasploit Application Flow

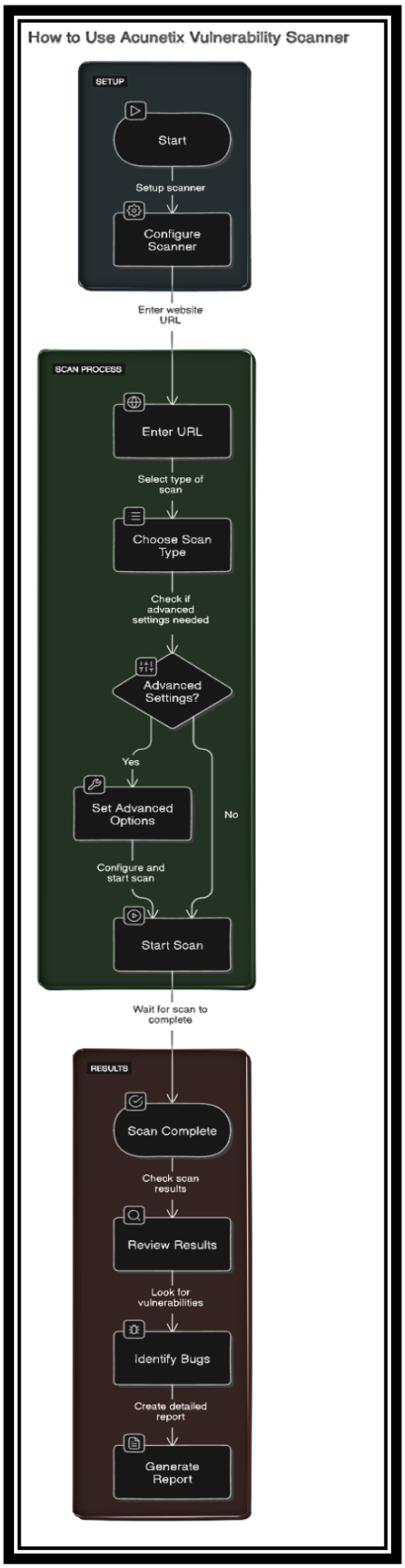


Figure 41 Acunetix Application Flow

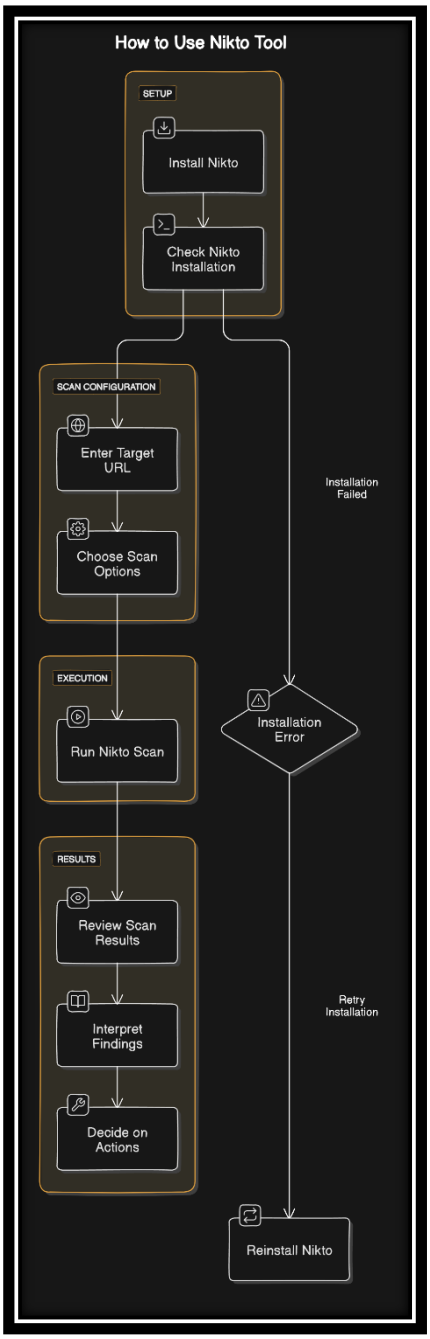


Figure 42 Nikto Application Flow

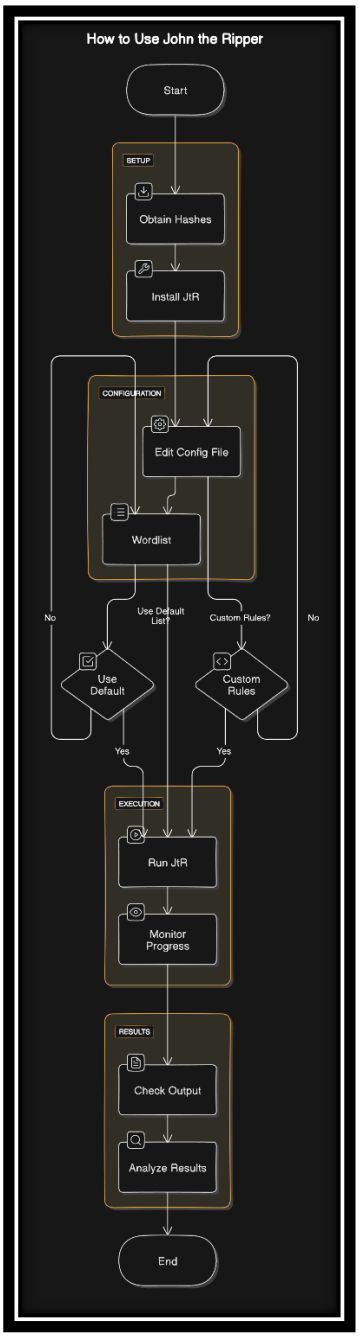


Figure 43 John the Ripper Application Flow

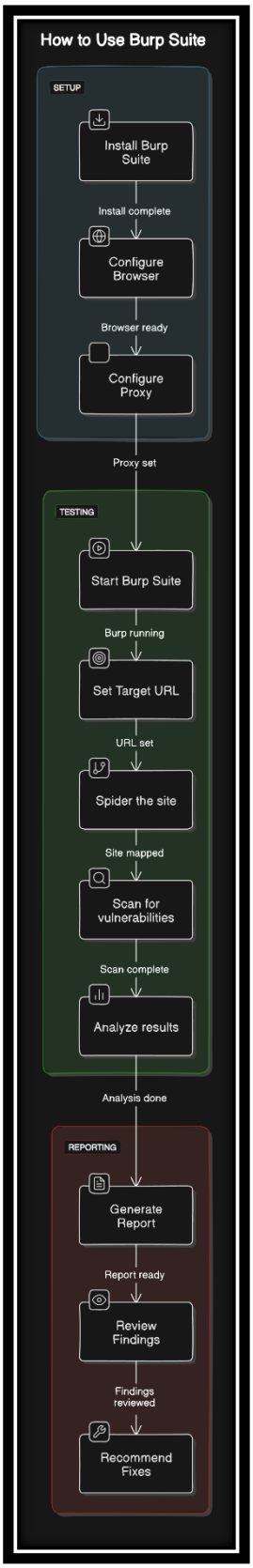


Figure 44 Burp Suite Application Flow

7. References & Bibliography

References:

1. OWASP Top 10 Web Application Security Risks. (2021). Retrieved from https://owasp.org/www-project-top-ten/
2. Nmap: Network Mapper. (n.d.). Retrieved from https://nmap.org/
3. WPScan Documentation. (n.d.). Retrieved from https://wpscan.com/documentation
4. Acunetix Web Vulnerability Scanner. (n.d.). Retrieved from https://www.acunetix.com/
5. Nessus Vulnerability Scanner. (n.d.). Retrieved from https://www.tenable.com/products/nessus
6. Burp Suite Professional. (n.d.). Retrieved from https://portswigger.net/burp
7. Hashcat: Advanced Password Recovery. (n.d.). Retrieved from https://hashcat.net/hashcat/
8. John the Ripper password cracker. (n.d.). Retrieved from https://www.openwall.com/john/

Bibliography:

1. Stuttard, D., & Pinto, M. (2011). The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws (2nd ed.). Wiley.
2. Engebretson, P. (2013). The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy (2nd ed.). Syngress.
3. Weidman, G. (2014). Penetration Testing: A Hands-On Introduction to Hacking. No Starch Press.
4. Andreu, A. (2006). Professional Penetration Testing: Creating and Learning in a Hacking Lab. Syngress.
5. Vashishtha, N. (2014). Web Penetration Testing with Kali Linux. Packt Publishing.
6. Ferriman, J. (2014). WordPress Security. Packt Publishing.
7. Zeltser, L. (2019). Mastering Kali Linux for Advanced Penetration Testing (3rd ed.). Packt Publishing.
8. OWASP Foundation. (2020). OWASP Web Application Security Testing Checklist. OWASP.