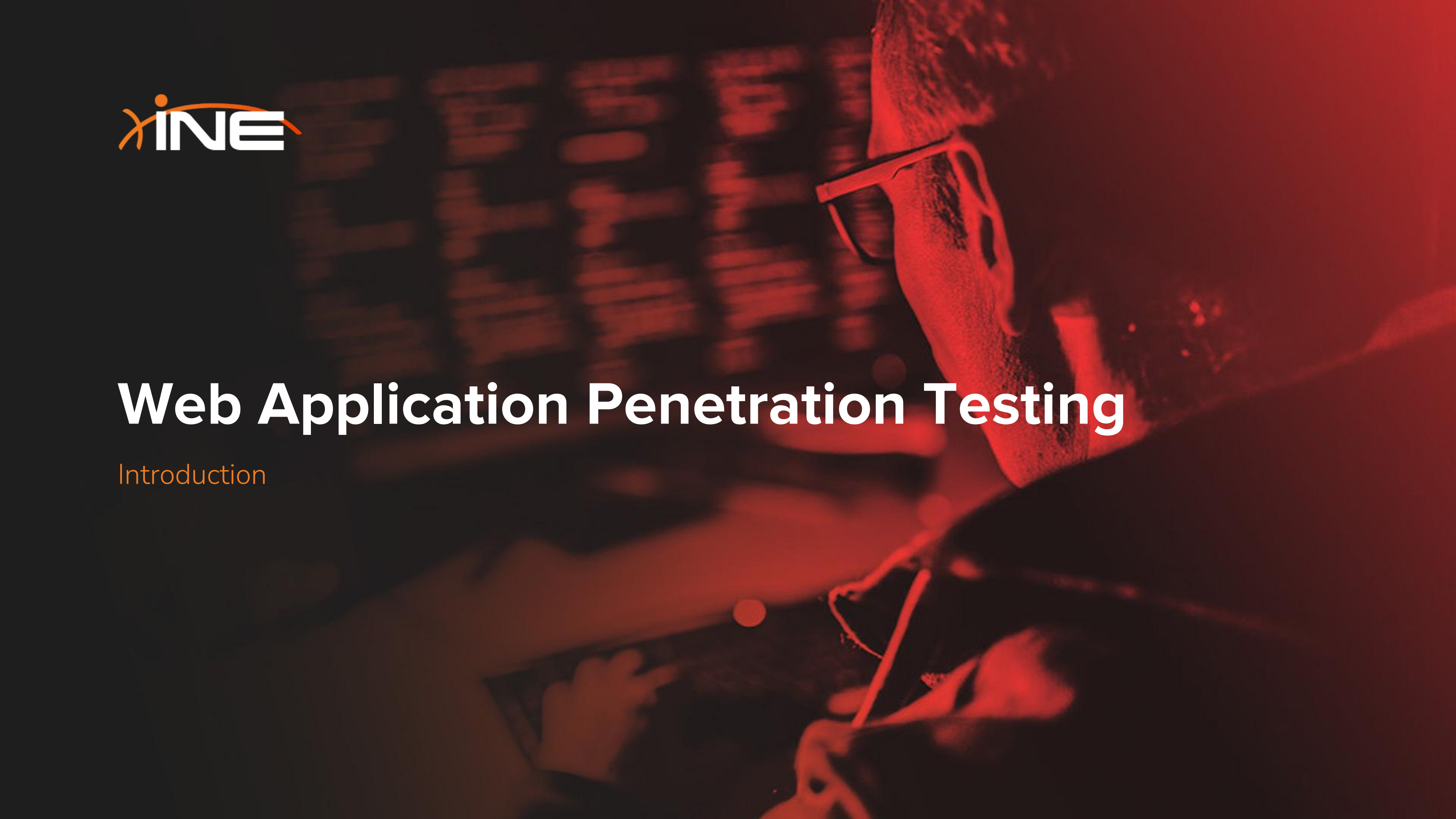


Web Application Penetration Testing

Introduction

Alexis Ahmed

Senior Penetration Tester & Red Teamer @HackerSploit

Red Team Instructor @INE

✉ aahmed@ine.com

@HackerSploit

in @alexisahmed

Course Topic Overview

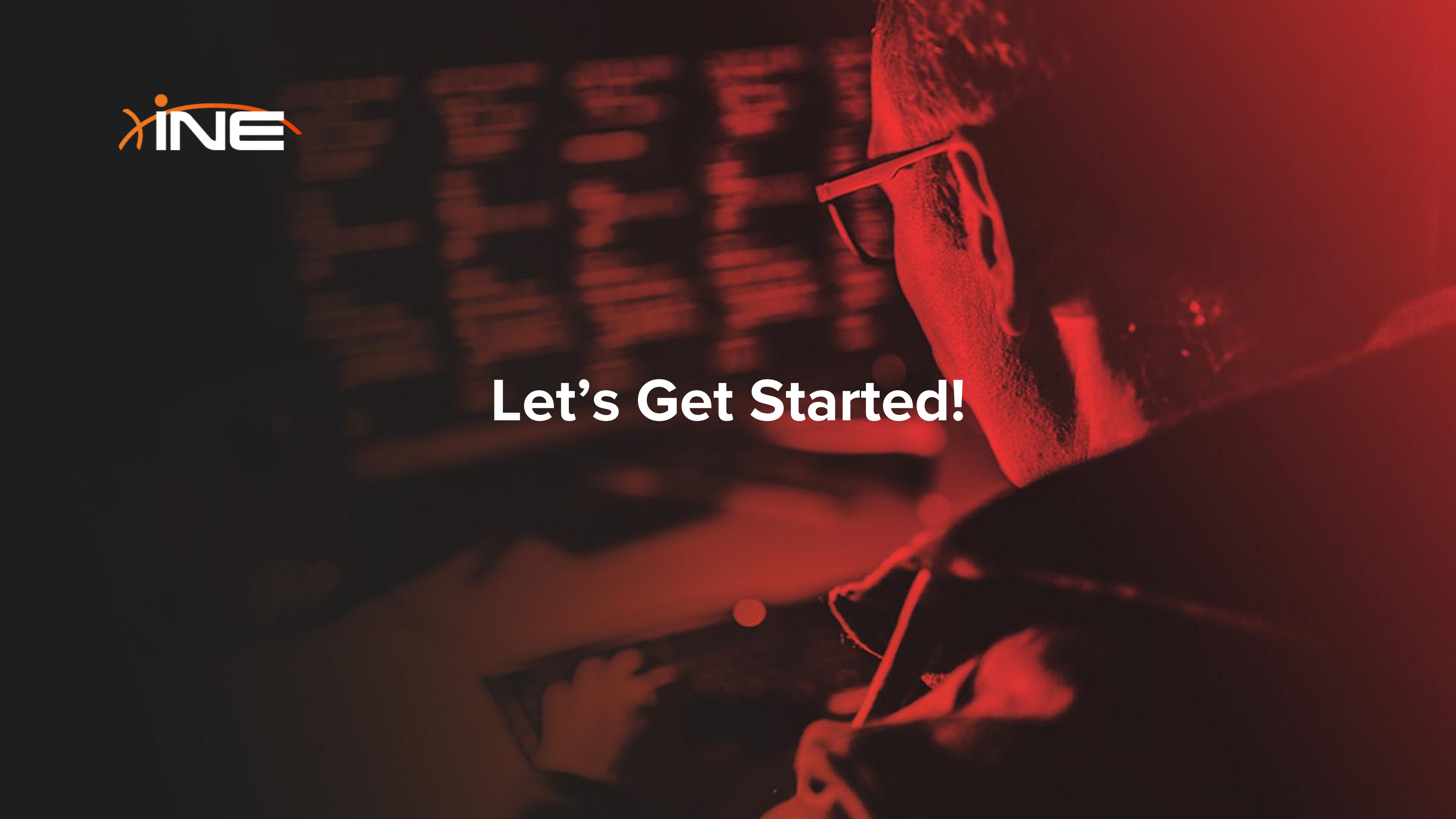
- Passive Information Gathering
- Active Information Gathering
- Web App Enumeration & Fingerprinting
- Web Server Fingerprinting & Vulnerability Scanning
- File & Directory Brute Force Attacks
- Identifying & Exploiting Stored, Reflected & DOM-Based XSS Vulnerabilities
- Identifying & Exploiting Error-Based and Union-Based SQL Injection Vulnerabilities
- Automated Recon Frameworks

Prerequisites

- Basic familiarity with the web (TCP/IP, UDP & HTTP)
- Familiarity with Windows & Linux

Learning Objectives:

- You will learn how to use the OWASP Web Security Testing Guide as a methodology for web app pentesting engagements.
- You will be able to perform passive web app information gathering.
- You will learn how to to perform passive and active DNS enumeration.
- You will learn how to detect web application firewalls (WAF).
- You will learn how to perform spidering and crawling to identify the content structure of websites.
- You will learn how to perform subdomain enumeration through publicly available sources and through subdomain brute-force attacks..
- You will learn how to perform file & directory enumeration.
- You will learn how to utilize automated tools and web proxies to identify and exploit XSS vulnerabilities in web applications.
- You will be able to identify and exploit In-Band SQL Injection vulnerabilities (Error-Based SQLi & UNION-Based SQLi).



Let’s Get Started!

Introduction To Web Enumeration & Information Gathering

What is Information Gathering?

- Information gathering is the first step of any penetration test and involves gathering or collecting information about an individual, company, website or system that you are targeting.
- The more information you have on your target, the more successful you will be during the latter stages of a penetration test.
- Information gathering is typically broken down into two types:
  - Passive information gathering - Involves gathering as much information as possible without actively engaging with the target.
  - Active information gathering/Enumeration - Involves gathering as much information as possible by actively engaging with the target system. (You will require authorization in order to perform active information gathering)

What is Information Gathering?

- Gathering information about the target server/web app is the initial phase of any penetration test, and is arguable the most important phase of the entire engagement.
- One of the nuances of this phase is that there is no unnecessary information, everything you collect should be recorded/saved for future use.
- In the context of web application penetration testing, the information collected in this phase will become extremely useful in understanding the website/web application logic and structure during the initial access/exploitation phase.

What Information Are We Looking For?

- Website & domain ownership.
- IP addresses, domains and subdomains.
- Hidden files & directories.
- Hosting infrastructure (web server, CMS, Database etc).
- Presence of defensive solutions like a web application firewall (WAF).

Passive Information Gathering

Passive Information Gathering

- Identifying domain names and domain ownership information.
- Discovering hidden/disallowed files and directories.
- Identifying web server IP addresses & DNS records.
- Identifying web technologies being used on target sites.
- WAF detection.
- Identifying subdomains.
- Identify website content structure.

Active Information Gathering/Enumeration

Active Information Gathering

- Downloading & analyzing website/web app source code.
- Port scanning & service discovery.
- Web server fingerprinting.
- Web application scanning.
- DNS Zone Transfers.
- Subdomain enumeration via Brute-Force.

OWASP Web Security Testing Guide

Demo: Using The OWASP Web Security Testing Guide

WHOIS

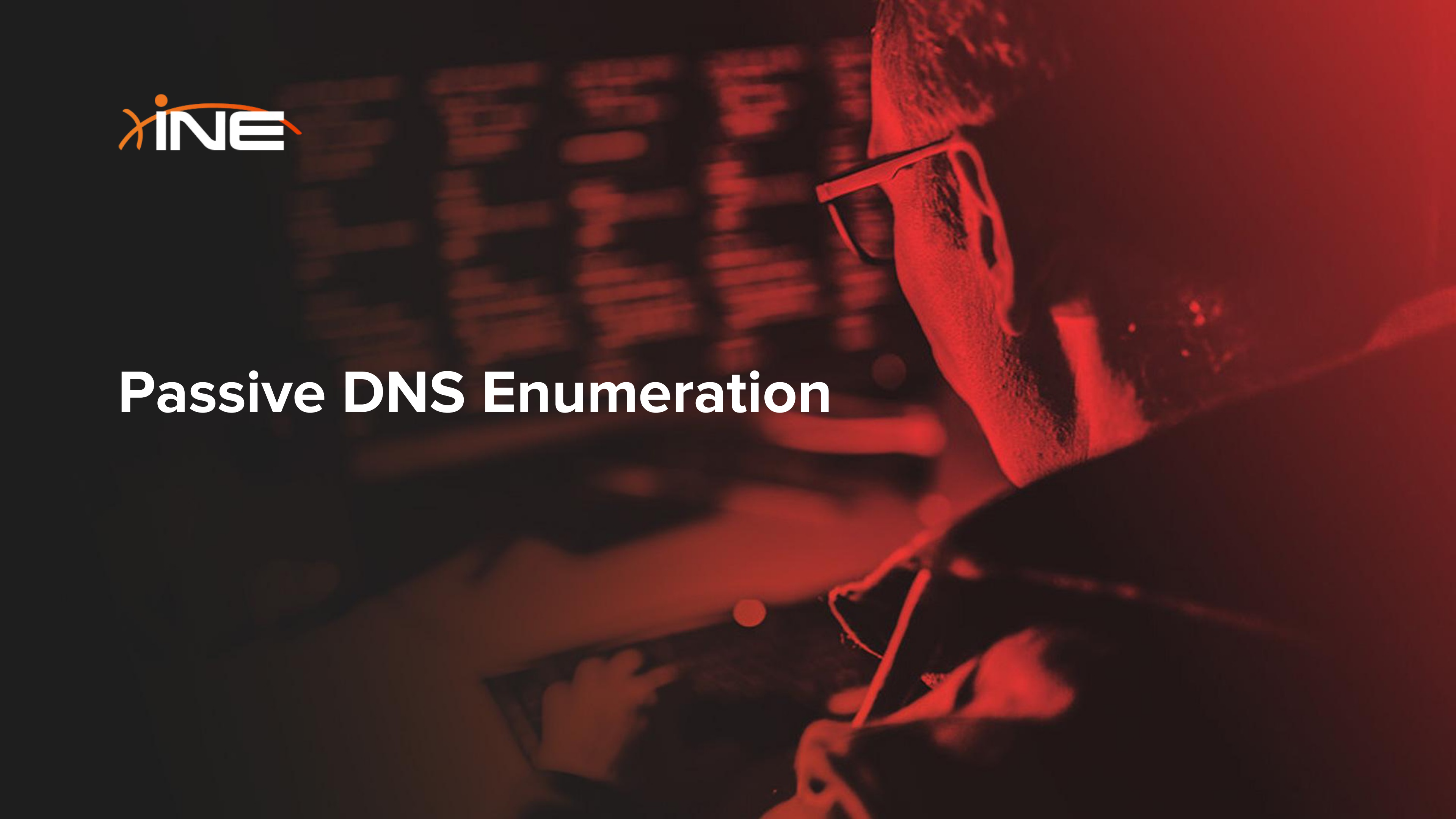
WHOIS

- WHOIS is a query and response protocol that is used to query databases that store the registered users or organizations of an internet resource like a domain name or an IP address block.
- WHOIS lookups can be performed through the command line interface via the whois client or through some third party web-based tools to lookup the domain ownership details from different databases.

Demo: Identifying Ownership Information With WHOIS

Website Fingerprinting With Netcraft

Practical Demo



Passive DNS Enumeration

DNS Enumeration

- Now that we have gathered some valuable information about our target, we can start digging deeper into the data we found to build a map/topology of the target site and it’s underlying infrastructure.
- A valuable resource for this information is the Domain Name System (DNS).
- We can query DNS to identify the DNS records associated with a particular domain or IP address.

DNS

- Domain Name System (DNS) is a protocol that is used to resolve domain names/hostnames to IP addresses.
- During the early days of the internet, users would have to remember the IP addresses of the sites that they wanted to visit, DNS resolves this issue by mapping domain names (easier to recall) to their respective IP addresses.
- A DNS server (nameserver) is like a telephone directory that contains domain names and their corresponding IP addresses.
- A plethora of public DNS servers have been set up by companies like Cloudflare (1.1.1.1) and Google (8.8.8.8). These DNS servers contain the records of almost all domains on the internet.

DNS Records

- A - Resolves a hostname or domain to an IPv4 address.
- AAAA - Resolves a hostname or domain to an IPv6 address.
- NS - Reference to the domains nameserver.
- MX - Resolves a domain to a mail server.
- CNAME - Used for domain aliases.
- TXT - Text record.
- HINFO - Host information.
- SOA - Domain authority.
- SRV - Service records.
- PTR - Resolves an IP address to a hostname

Demo: Passive DNS Enumeration

Thank You!

Reviewing Webserver Metafiles

Practical Demo

Web App Technology Fingerprinting

Practical Demo

Passive Crawling & Spidering With Burp Suite & OWASP ZAP

Crawling

- Crawling is the process of navigating around the web application, following links, submitting forms and logging in (where possible) with the objective of mapping out and cataloging the web application and the navigational paths within it.
- Crawling is typically passive as engagement with the target is done via what is publicly accessible, we can utilize Burp Suite's passive crawler to help us map out the web application to better understand how it is setup and how it works.

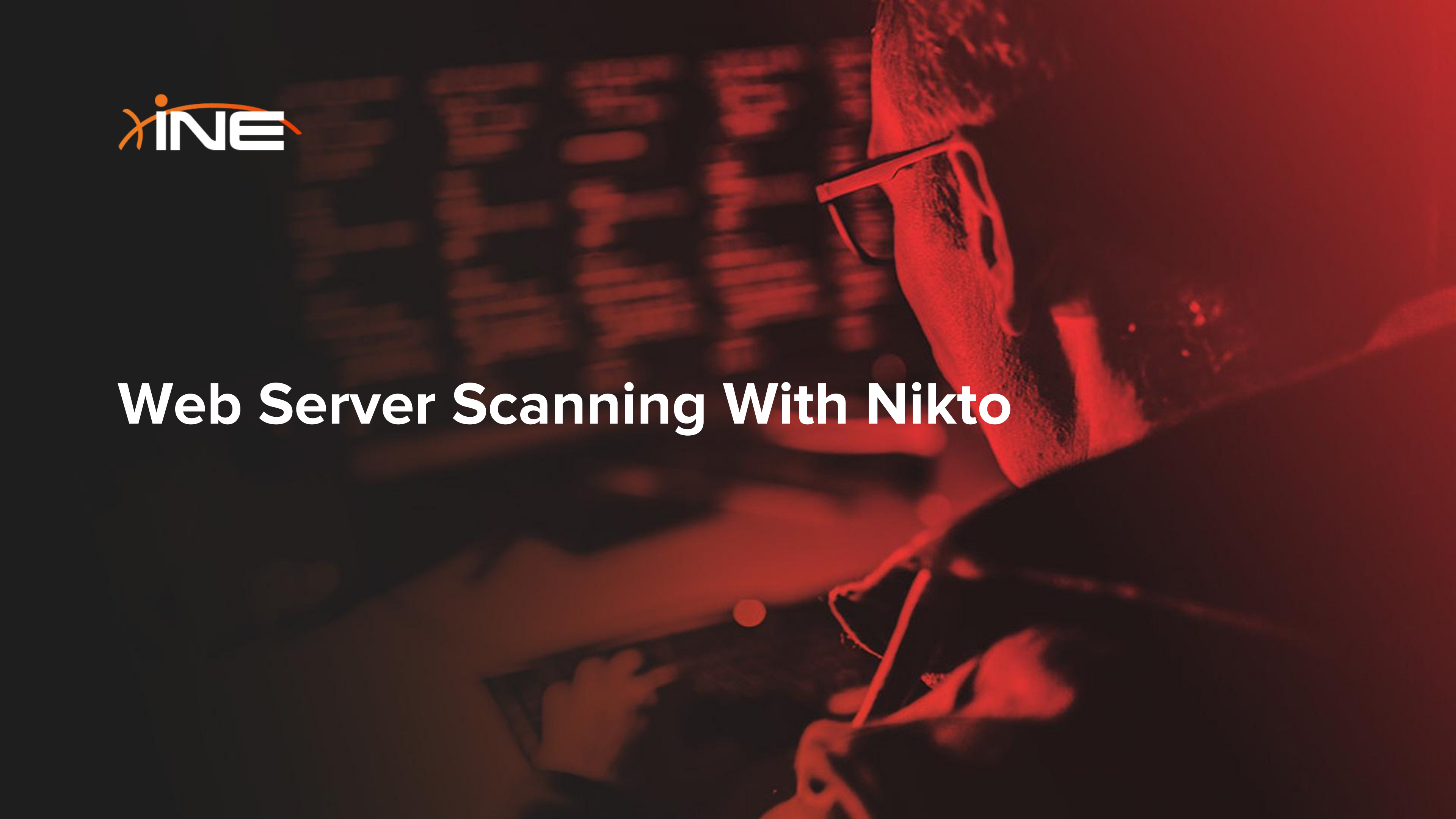
Spidering

- Spidering is the process of automatically discovering new resources (URLs) on a web application/site.
- It typically begins with a list of target URLs called seeds, after which the Spider will visit the URLs and identified hyperlinks in the page and adds them to the list of URLs to visit and repeats the process recursively.
- Spidering can be quite loud and as a result, it is typically considered to be an active information gathering technique.
- We can utilize OWASP ZAP's Spider to automate the process of spidering a web application to map out the web application and learn more about how the site is laid out and how it works.

Practical Demo

Web Server Fingerprinting

Practical Demo



Web Server Scanning With Nikto

Practical Demo

File & Directory Brute-Force

Practical Demo

Automated Web Recon With OWASP Amass

Practical Demo

Exploiting Reflected XSS Vulnerabilities in WordPress

Lab Demo: Exploiting Reflected XSS Vulnerabilities in WordPress

Exploiting Stored XSS Vulnerabilities in MyBB Forum

Lab Demo: Exploiting Stored XSS Vulnerabilities in MyBB Forum

Exploiting DOM-Based XSS Vulnerabilities

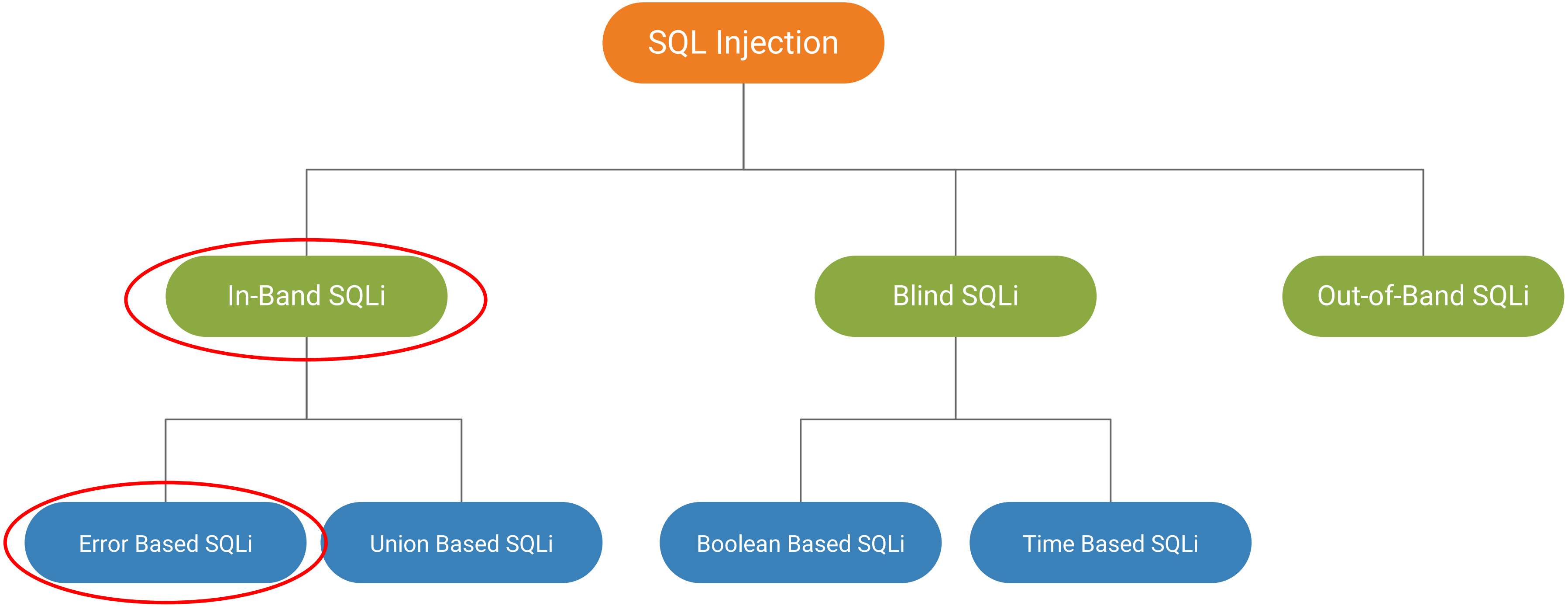
Lab Demo: Exploiting DOM-Based XSS Vulnerabilities

Finding SQL Injection Vulnerabilities With OWASP ZAP

Demo: Finding SQL Injection Vulnerabilities With OWASP ZAP

Exploiting Error-Based SQL Injection Vulnerabilities

SQL Injection Types & Subtypes



SQL Injection

In-Band SQLi

Blind SQLi

Out-of-Band SQLi

Error Based SQLi

Union Based SQLi

Boolean Based SQLi

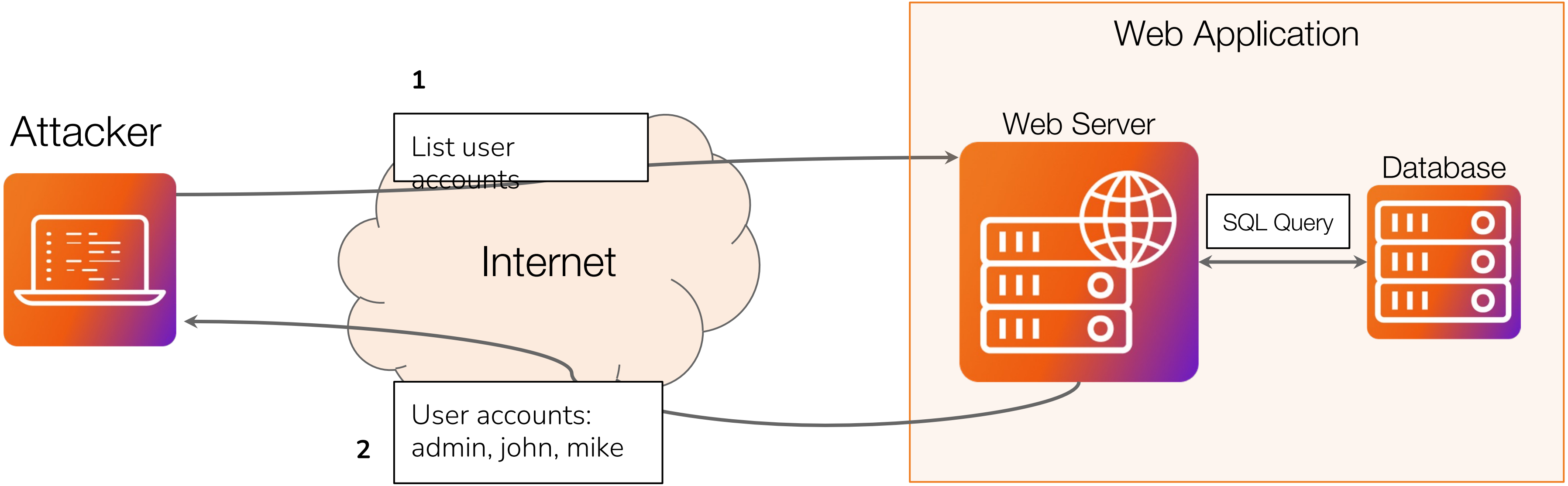
Time Based SQLi

In-Band SQL Injection

- In-band SQL injection is the most common type of SQL injection attack. It occurs when an attacker uses the same communication channel to send the attack and receive the results.
- In other words, the attacker injects malicious SQL code into the web application and receives the results of the attack through the same channel used to submit the code.
- In-band SQL injection attacks are dangerous because they can be used to steal sensitive information, modify or delete data, or take over the entire web application or even the entire server.

In-Band SQL Injection

During an in-band SQLi attack the penetration tester finds a way to ask the the web application for desired information.



Error-Based SQL Injection

- Error-based SQL injection is a technique used by attackers to exploit SQL injection vulnerabilities in web applications.
- It relies on intentionally causing database errors and using the error messages returned by the database to extract information or gain unauthorized access to the application's database.
- The error message can contain valuable information about the database schema or the contents of the database itself, which the attacker can use to further exploit the vulnerability.
- Identifying error-based SQL injection vulnerabilities involves testing the web application to determine if it is susceptible to this type of attack.

Error-Based SQL Injection Methodology

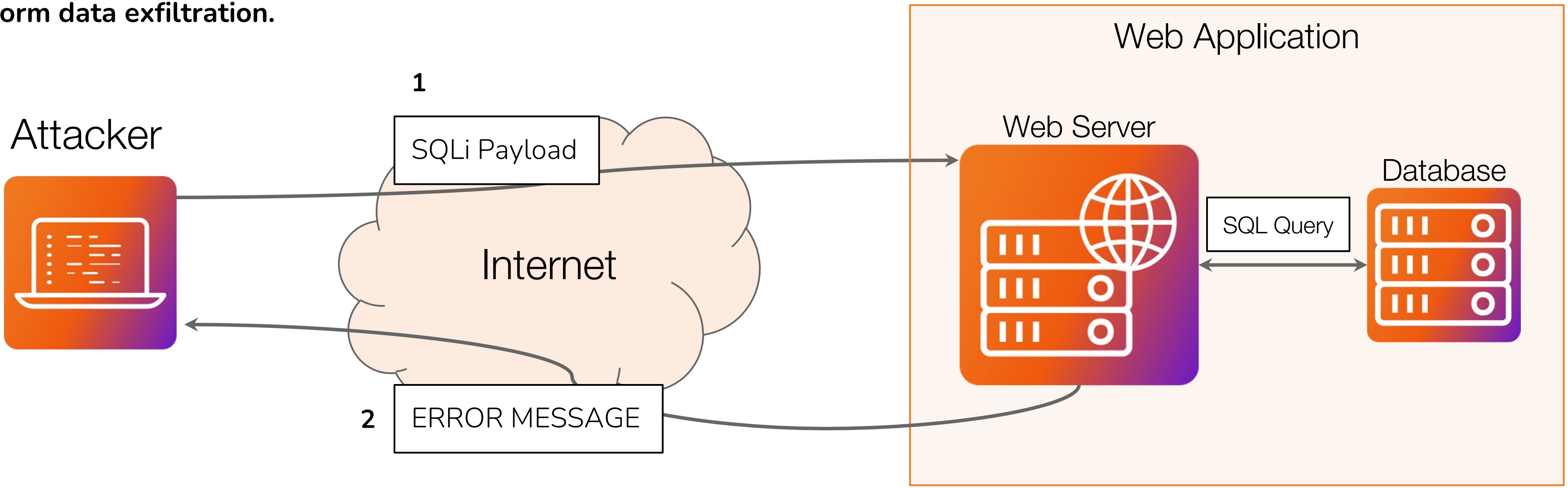
- Identify a vulnerable parameter: Find a parameter in the web application that is vulnerable to SQL injection, typically through user input fields, URL parameters, or form inputs.
- Inject malicious SQL code: Craft a payload that includes SQL statements designed to trigger a database error. This can involve appending invalid SQL syntax or manipulating existing queries.
- Observe error messages: Submit the payload to the vulnerable parameter and observe the error message returned by the database. The error message can provide valuable information about the structure and content of the database.

Error-Based SQL Injection Methodology

- Extract data: Modify the payload to extract specific information from the database by leveraging the error messages. This can include retrieving usernames, passwords, or other sensitive data stored in the database.
- Exploit the vulnerability: Exploit the information gathered through error-based SQL injection to further exploit the application, gain unauthorized access, or perform other malicious actions.

Error Based SQL Injection

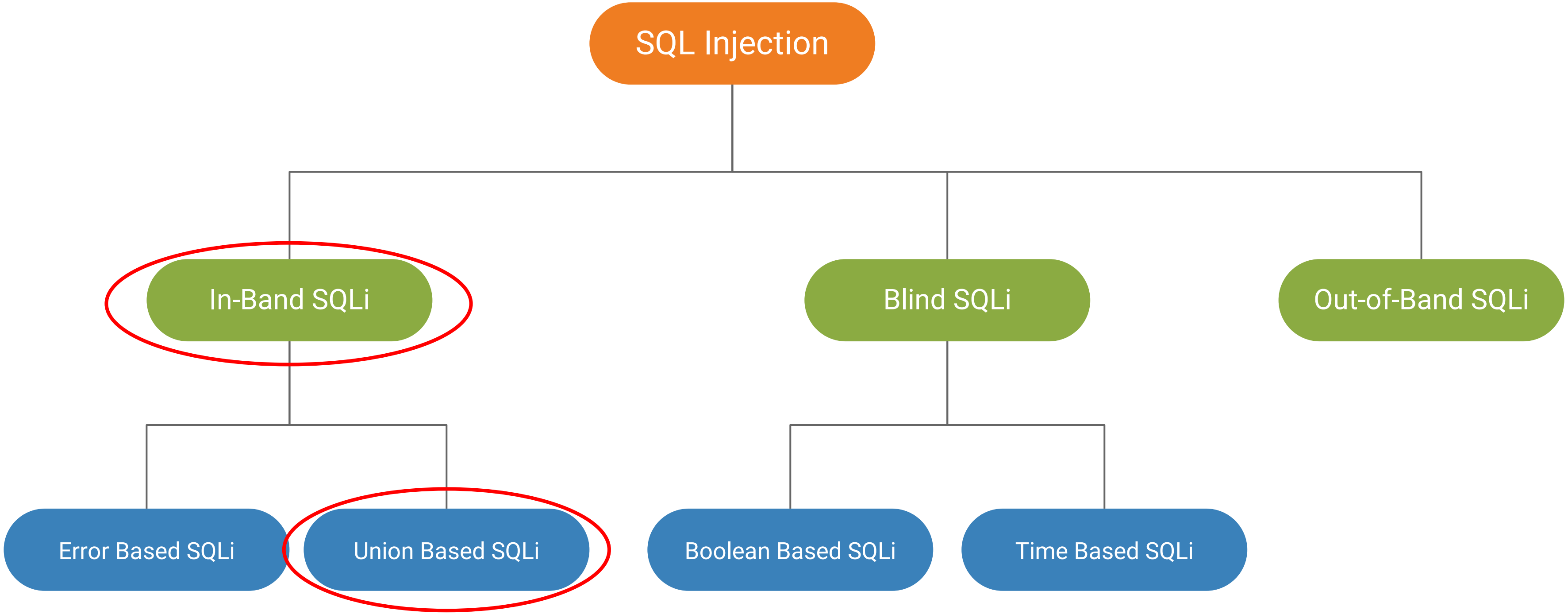
During an Error-Based SQL injection attack, the penetration tester tries to force the DBMS to output an error message and then uses that information to perform data exfiltration.



Demo: Exploiting Error-Based SQL Injection Vulnerabilities

Exploiting Union-Based SQL Injection Vulnerabilities

SQL Injection Types & Subtypes



Union-Based SQL Injection

- Union-based SQL injection is a type of SQL injection attack that exploits the ability to use the UNION operator in SQL queries.
- It occurs when an application fails to properly validate or sanitize user input and allows an attacker to inject malicious SQL code into the query.
- The UNION operator is used in SQL to combine the results of two or more SELECT statements into a single result set.
- It requires that the number of columns and their data types match in the SELECT statements being combined.
- In a union-based SQL injection attack, the attacker injects additional SELECT statements through the vulnerable input to retrieve data from other database tables or to extract sensitive information.

Union-Based SQL Injection

Here's an example to illustrate the concept. Consider the following vulnerable code snippet:

```
SELECT id, name FROM users WHERE id = '<user_input>'
```

An attacker can exploit this vulnerability by injecting a UNION-based attack payload into the <user_input> parameter. They could inject a statement like:

```
' UNION SELECT credit_card_number, 'hack' FROM credit_cards --
```

The injected payload modifies the original query to retrieve the credit card numbers along with a custom value ('hack') from the credit_cards table. The double dash at the end is used to comment out the remaining part of the original query.

Union-Based SQL Injection

If the application is vulnerable to union-based SQL injection, the modified query would become:

```
SELECT id, name FROM users WHERE id = '' UNION SELECT
credit_card_number, 'hack' FROM credit_cards --
```

The database would then execute this modified query, and the result would include the credit card numbers alongside the original user data. The attacker can subsequently extract this sensitive information.

Union-Based SQL Injection Methodology

- Identify user inputs: Determine the inputs on the application that are used in database queries. These inputs can include URL parameters, form fields, cookies, or any other user-controllable data.
- Test inputs for vulnerability: Inject a simple payload, such as a single quote (') or a double quote ("). If the application produces an error or exhibits unexpected behavior, it might indicate a potential SQL injection vulnerability.
- Identify vulnerable injection points: Manipulate the injected payload to check if the application responds differently based on the injected data. You can try injecting various payloads like UNION SELECT statements or boolean conditions (e.g., ' OR '1'='1) to see if the application behaves differently based on the response.

Union-Based SQL Injection Methodology

- Confirm the presence of a vulnerability: Once you have identified a potential injection point, you need to confirm if it is vulnerable to Union-based SQL injection. To do this, you can inject a UNION SELECT statement and observe the application's response. If the response includes additional columns or unexpected data, it is likely vulnerable to Union-based SQL injection.
- Enumerate the database: Exploit the Union-based SQL injection vulnerability to enumerate the database structure. Inject UNION SELECT statements with appropriate column names and table names to retrieve information about the database schema, tables, and columns. You can use techniques like ORDER BY or LIMIT clauses to retrieve specific information.

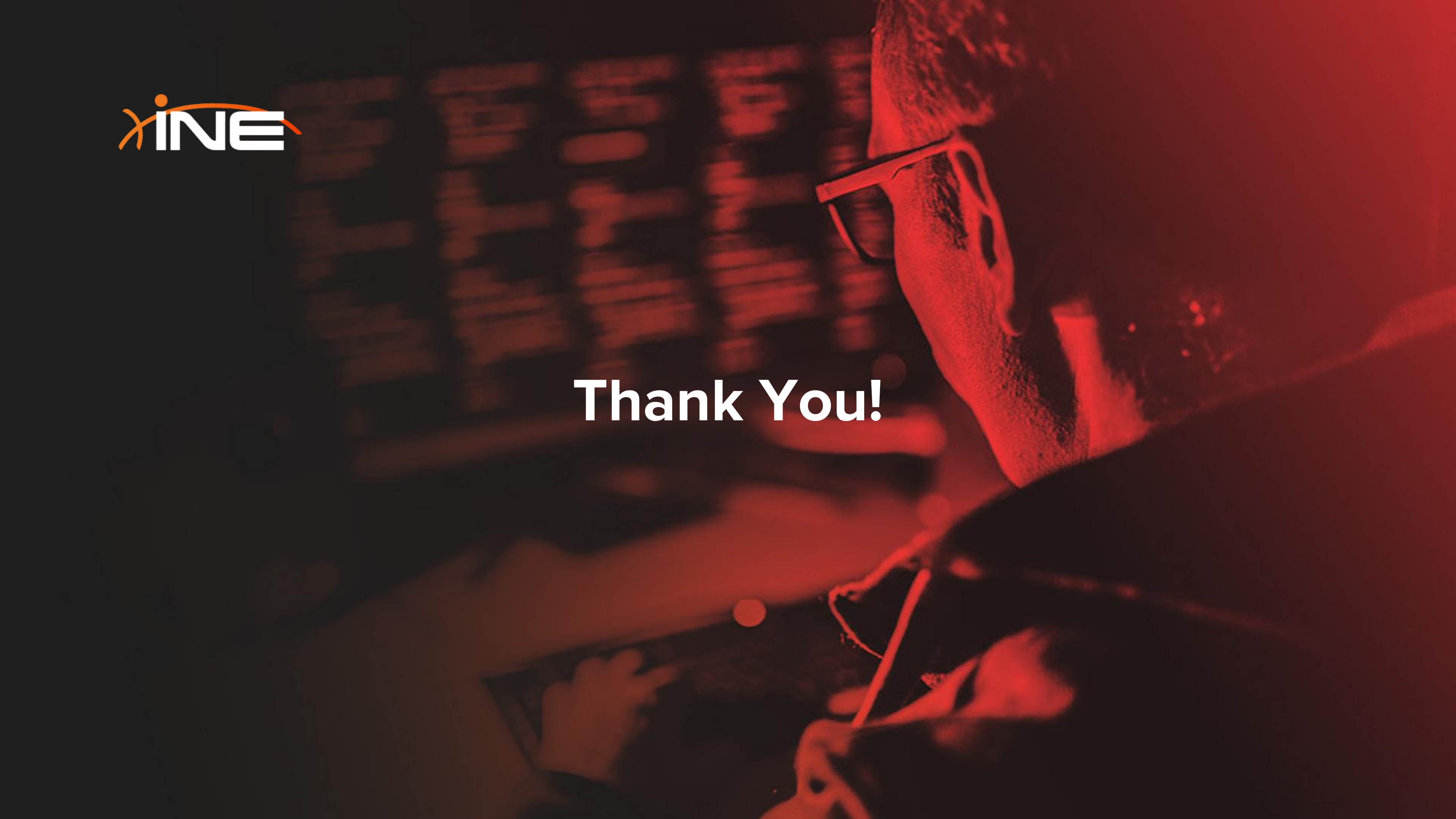
Demo: Exploiting Union-Based SQL Injection Vulnerabilities

Web Application Penetration Testing

Conclusion

Learning Objectives:

- You will learn how to use the OWASP Web Security Testing Guide as a methodology for web app pentesting engagements.
- You will be able to perform passive web app information gathering.
- You will learn how to to perform passive and active DNS enumeration.
- You will learn how to detect web application firewalls (WAF).
- You will learn how to perform spidering and crawling to identify the content structure of websites.
- You will learn how to perform subdomain enumeration through publicly available sources and through subdomain brute-force attacks..
- You will learn how to perform file & directory enumeration.
- You will learn how to utilize automated tools and web proxies to identify and exploit XSS vulnerabilities in web applications.
- You will be able to identify and exploit In-Band SQL Injection vulnerabilities (Error-Based SQLi & UNION-Based SQLi).



Thank You!

EXPERTS AT MAKING YOU AN EXPERT

INE