Investigate Web Attack – DFIR Challenge Report

Author: Harikumar Nandakumar
Platform: LetsDefend – Blue Team Challenge
Challenge: Investigate Web Attack
Date: July 2025

---

1. Executive Summary

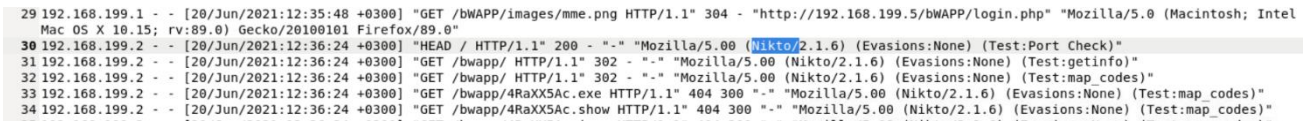
This report documents a real-world DFIR challenge completed on LetsDefend titled "Investigate Web Attack." The challenge required analyzing Apache access logs to trace and understand the phases of a multi-step web attack. The investigation revealed an attacker using automated tools to perform reconnaissance, brute force authentication, code injection, and persistence mechanisms on a vulnerable web application.

The goal was to determine the tools and techniques used, validate whether each phase was successful, and suggest prevention strategies based on observed patterns.

Phase 1: Reconnaissance – Nikto Scanner Detected

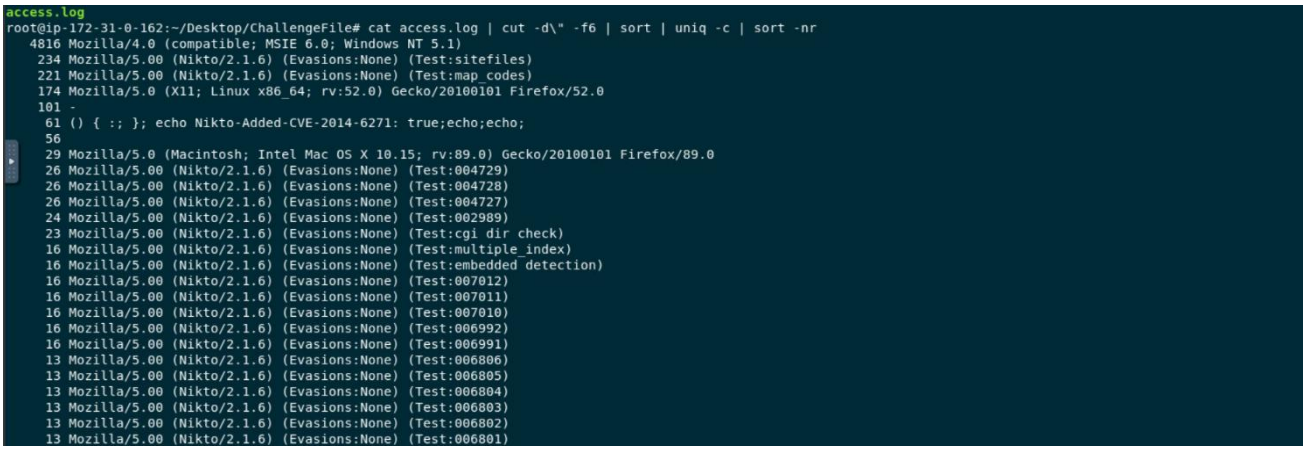
Objective:
Identify whether any automated scanning tools were used by the attacker to enumerate the web server.

Evidence from Log:
The investigation began by inspecting the Apache access log file.In the Apache access log, the following entry was identified:



Confirmation via CLI:
To further verify and quantify the use of Nikto, the following Linux command was used:

cat access.log | cut -d\" -f6 | sort | uniq -c | sort -nr



This string clearly indicates that the attacker used **Nikto**, a well-known open-source web vulnerability scanner. The tool is frequently used to identify outdated software, misconfigurations, directory structures, and potential injection points.
This command extracts and counts all User-Agent strings from the access log. The result revealed:
234 Mozilla/5.00 (Nikto/2.1.6)
This confirms extensive scanning activity using Nikto, validating that the initial phase of the attack was automated reconnaissance.

TTPs Observed:

- **Tactic:** Reconnaissance
- **Technique:** T1595.002 – Active Scanning: Vulnerability Scanning
- **Tool Identified:** Nikto v2.1.6

Takeaway:
Early detection of automated scanning tools like Nikto can help defenders block malicious actors before exploitation. Monitoring for Nikto in access logs is a simple but effective defensive measure.

Phase 2: Directory Brute Forcing – Enumeration in Action

Objective:
Identify if the attacker attempted to discover hidden directories or sensitive configuration files after reconnaissance.

Observations from Log Review:
Immediately following the final Nikto scan request (/bwapp/sixcms/admin/login), the attacker initiated a series of GET requests targeting various directory and file names. These requests were sent in alphabetical order, which is a strong indicator of an automated wordlist-based attack.Some of the attempted paths include.

Following the tail end of the Nikto scan (log line 7567), multiple suspicious GET requests were made to paths like:

```
7564 - - [20/Jun/2021:12:36:57 +0300] "GET /bwapp/wiki/VERSION HTTP/1.1" 404 300 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:007022)"
7565 - - [20/Jun/2021:12:36:57 +0300] "GET /bwapp/dokuwiki/VERSION HTTP/1.1" 404 300 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:007022)"
7566 - - [20/Jun/2021:12:36:57 +0300] "GET /bwapp/solr/#/ HTTP/1.1" 400 326 "-" "-"
7567 - - [20/Jun/2021:12:36:57 +0300] "GET /bwapp/sixcms/admin/login/ HTTP/1.1" 404 300 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:007024)"
7568 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/randomfile1 HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7569 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/frand2 HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7570 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.bash_history HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7571 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.bashrc HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

/bwapp/.bash_history

/bwapp/.git/HEAD

/bwapp/.htaccess

/bwapp/.ssh

/bwapp/.mysql_history

/bwapp/.htpasswd

HTTP Response Patterns:

- Status codes such as 404 (Not Found) and 403 (Forbidden) were frequently returned.
- The consistent use of paths like .git/HEAD, .htaccess, and .bashrc suggests the attacker was looking for configuration leaks or sensitive artifacts.

```
7579 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.hta HTTP/1.1" 403 303 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7580 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.hta_ HTTP/1.1" 403 303 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7581 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.htaccess HTTP/1.1" 403 303 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7582 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.htaccess_ HTTP/1.1" 403 303 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7583 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.htpasswd HTTP/1.1" 403 303 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7584 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.htpasswd_ HTTP/1.1" 403 303 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7585 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.listing HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7586 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.listings HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7587 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.mysql_history HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7588 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.passwd HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7589 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.perf HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7590 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.profile HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7591 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.rhosts HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7592 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.sh_history HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7593 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.ssh HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7594 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.subversion HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7595 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.svn HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7596 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.svn/entries HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7597 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.swf HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7598 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/.web HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7599 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/@ HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7600 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_ HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7601 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_adm HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7602 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_admin HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7603 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_ajax HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7604 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_archive HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7605 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_assets HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7606 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_backup HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7607 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_baks HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7608 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_borders HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7609 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_cache HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7610 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_catalogs HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7611 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_code HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7612 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_common HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7613 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_conf HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7614 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_config HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
7615 - - [20/Jun/2021:12:37:50 +0300] "GET /bwapp/_css HTTP/1.1" 404 300 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

Command Pattern Reference:

```
root@ip-172-31-0-162:~/Desktop/ChallengeFile# grep -n "Nikto" access.log | tail -n 1
7567:192.168.199.2 - - [20/Jun/2021:12:36:57 +0300] "GET /bwapp/sixcms/admin/login/ HTTP/1.1" 404 300 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:007024)"
```

This command revealed a clear transition from vulnerability scanning to targeted directory enumeration.

TTPs Identified:

- **Tactic:** Discovery
- **Technique:** T1083 – File and Directory Discovery
- **Behavioral Indicators:** Sequential path attempts, known sensitive files, and status feedback analysis.

Analysis:
This pattern confirms the use of a **directory brute-forcing tool**, possibly Gobuster, DirBuster, or even a Nikto module configured for deeper enumeration. The attacker is attempting to uncover default, hidden, or misconfigured files and folders that might expose credentials, version control, or shell history.

Implications:
If even one of these paths were left unprotected or exposed, it could result in:

- Leaked source code or configuration.
- Enumeration of usernames/passwords.
- Access to credential storage (.mysql_history, .htpasswd).
- Information leading to privilege escalation.

Defensive Recommendations:

- Deny public access to sensitive files via .htaccess or server config.
- Implement WAF rules to detect path fuzzing and block directory brute force attempts.
- Use file integrity monitoring to watch for unexpected access to sensitive files.
- Limit response verbosity (avoid helpful 403/404 metadata in production).

Phase 3: Brute Force Login – Credential Stuffing Attempt

Objective:
Determine whether the attacker attempted to gain access to the web application through brute-force login methods.

Observations from Log Review:
Upon analyzing the access logs post-directory enumeration, a pattern of repetitive POST requests targeting the /bWAPP/login.php endpoint was observed.

Example log entries:

- The request method remained constant (POST).
- Status codes were initially **200** OK, indicating failed login attempts.
- Each request had a static payload size (4086 bytes), suggesting that the attacker used the same parameters repeatedly.

Pattern Indicators:

- No changes in POST parameters = likely credential reuse.
- Repeated requests = automation or brute-force tool.
- Time intervals = near-identical, indicating script-driven behavior.

Brute Force Login - Attack Success Confirmed

Confirmation of Success:

After a series of unsuccessful attempts returning HTTP 200 with a constant response size (4086 bytes), the server responded with a **302 Found** status code. This change indicates a successful login and redirection, likely to a post-authentication landing page.

Log Evidence:

```
20100101 Firefox/52.0"
192.168.199.2 - - [20/Jun/2021:12:47:50 +0300] "POST /bWAPP/login.php HTTP/1.1" 200 4086
"http://192.168.199.5/bWAPP/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/-
20100101 Firefox/52.0"
192.168.199.2 - - [20/Jun/2021:12:47:52 +0300] "POST /bWAPP/login.php HTTP/1.1" 200 4086
"http://192.168.199.5/bWAPP/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/-
20100101 Firefox/52.0"
192.168.199.2 - - [20/Jun/2021:12:49:35 +0300] "POST /bWAPP/login.php HTTP/1.1" 200 4086
"http://192.168.199.5/bWAPP/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/-
20100101 Firefox/52.0"
192.168.199.2 - - [20/Jun/2021:12:49:35 +0300] "POST /bWAPP/login.php HTTP/1.1" 302 -
"http://192.168.199.5/bWAPP/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/-
20100101 Firefox/52.0"
192.168.199.2 - - [20/Jun/2021:12:50:10 +0300] "POST /bWAPP/login.php HTTP/1.1" 302 -
"http://192.168.199.5/bWAPP/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/-
20100101 Firefox/52.0"
```

What HTTP 302 Means:

A 302 Found response is used by web servers to redirect the user after a successful authentication. Combined with an increase in response size or subsequent access to protected resources, it confirms the attacker gained valid credentials.

Conclusion:

Yes, the brute-force login was successful.

The attacker likely accessed the authenticated section of the application and could proceed to the next phase: post-authentication exploitation.

TTPs Identified:

- **Tactic:** Credential Access
- **Technique:** T1110.001 – Brute Force: Password Guessing
- **Behavioral Pattern:** Static POST size, rapid repeated requests, no CAPTCHA or rate-limiting in place.

Tools Potentially Used:

- Hydra, Burp Intruder, Curl scripts, or custom brute-force scripts.

Impact:
Successful brute-force attacks could grant unauthorized access to the web application backend, leading to lateral movement or privilege escalation.

Defensive Recommendations:

- Implement CAPTCHA and brute-force lockouts after N failed login attempts.
- Use MFA (Multi-Factor Authentication).
- Monitor POST activity frequency and variance.
- Enable application-level logging of failed login attempts and IP correlation.

Phase 4: Code Injection – Command Execution via URL

Objective:
Determine if the attacker exploited a web parameter to execute arbitrary system commands, post-authentication.

Evidence from Access Logs:
After successfully logging into the system, the attacker targeted a vulnerable script. They began by sending benign test inputs:

```
12551 192.168.199.2 - - [20/Jun/2021:12:50:17 +0300] "GET /bWAPP/phpi.php?message=test HTTP/1.1" 200 12759 "http://192.168.199.5/bWAPP/phpi.php" "Mozilla/5.0 (X11;
      Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12552 192.168.199.2 - - [20/Jun/2021:12:51:37 +0300] "GET /bWAPP/phpi.php?message=test HTTP/1.1" 200 12759 "http://192.168.199.5/bWAPP/phpi.php" "Mozilla/5.0 (X11;
      Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12553 192.168.199.2 - - [20/Jun/2021:12:52:36 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27whoami%27) HTTP/1.1" 200 12778 "-" "Mozilla/5.0 (X11; Linux
      x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12554 192.168.199.2 - - [20/Jun/2021:12:52:46 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27net%20user%27) HTTP/1.1" 200 13045 "-" "Mozilla/5.0 (X11; Linux
      x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12555 192.168.199.2 - - [20/Jun/2021:12:52:56 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27net%20share%27) HTTP/1.1" 200 13175 "-" "Mozilla/5.0 (X11; Linux
```

Confirmation from Logs:
Shortly after, this evolved into command injection attempts:

```
12551 192.168.199.2 - - [20/Jun/2021:12:50:17 +0300] "GET /bWAPP/phpi.php?message=test HTTP/1.1" 200 12759 "http://192.168.199.5/bWAPP/phpi.php" "Mozilla/5.0 (X11;
      Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12552 192.168.199.2 - - [20/Jun/2021:12:51:37 +0300] "GET /bWAPP/phpi.php?message=test HTTP/1.1" 200 12759 "http://192.168.199.5/bWAPP/phpi.php" "Mozilla/5.0 (X11;
      Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12553 192.168.199.2 - - [20/Jun/2021:12:52:36 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27whoami%27) HTTP/1.1" 200 12778 "-" "Mozilla/5.0 (X11; Linux
      x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12554 192.168.199.2 - - [20/Jun/2021:12:52:46 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27net%20user%27) HTTP/1.1" 200 13045 "-" "Mozilla/5.0 (X11; Linux
      x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12555 192.168.199.2 - - [20/Jun/2021:12:52:56 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27net%20share%27) HTTP/1.1" 200 13175 "-" "Mozilla/5.0 (X11; Linux
      x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12556 192.168.199.2 - - [20/Jun/2021:12:53:13 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27net%20user%20hacker%20Asd123!!%20/add%27) HTTP/1.1" 200 12755 "-"
```

Purpose of Payload:

- The whoami command is commonly used in post-exploitation to determine the user context in which the application is running.
- It is often the first command used in enumeration to understand permission levels.

Commands Executed:

- whoami: To identify the current system user.
- net user: To enumerate user accounts.
- net share: To list shared resources.

Behavioral Analysis:

- The payload was encoded to bypass filtering mechanisms.
- The vulnerable application likely passed the input directly to a system shell via the system() function without proper sanitization.
- This clearly illustrates a command injection vulnerability.

TTPs Identified:

- **Tactic:** Execution
- **Technique:** T1059.001 – Command and Scripting Interpreter: PowerShell / System Shell

- **Attack Name:** Code Injection

Impact:
Successful command injection grants the attacker the ability to execute arbitrary OS commands, which can lead to:

- Full system compromise.
- Privilege escalation.
- Lateral movement.
- Persistence establishment.

Defensive Recommendations:

- Implement strict input validation and reject or sanitize special characters.
- Avoid using system()-like functions for handling user input.
- Use parameterized functions or safe language APIs.
- Apply runtime application self-protection (RASP) or **Web Application Firewalls** (WAFs) to detect and block injection attempts.

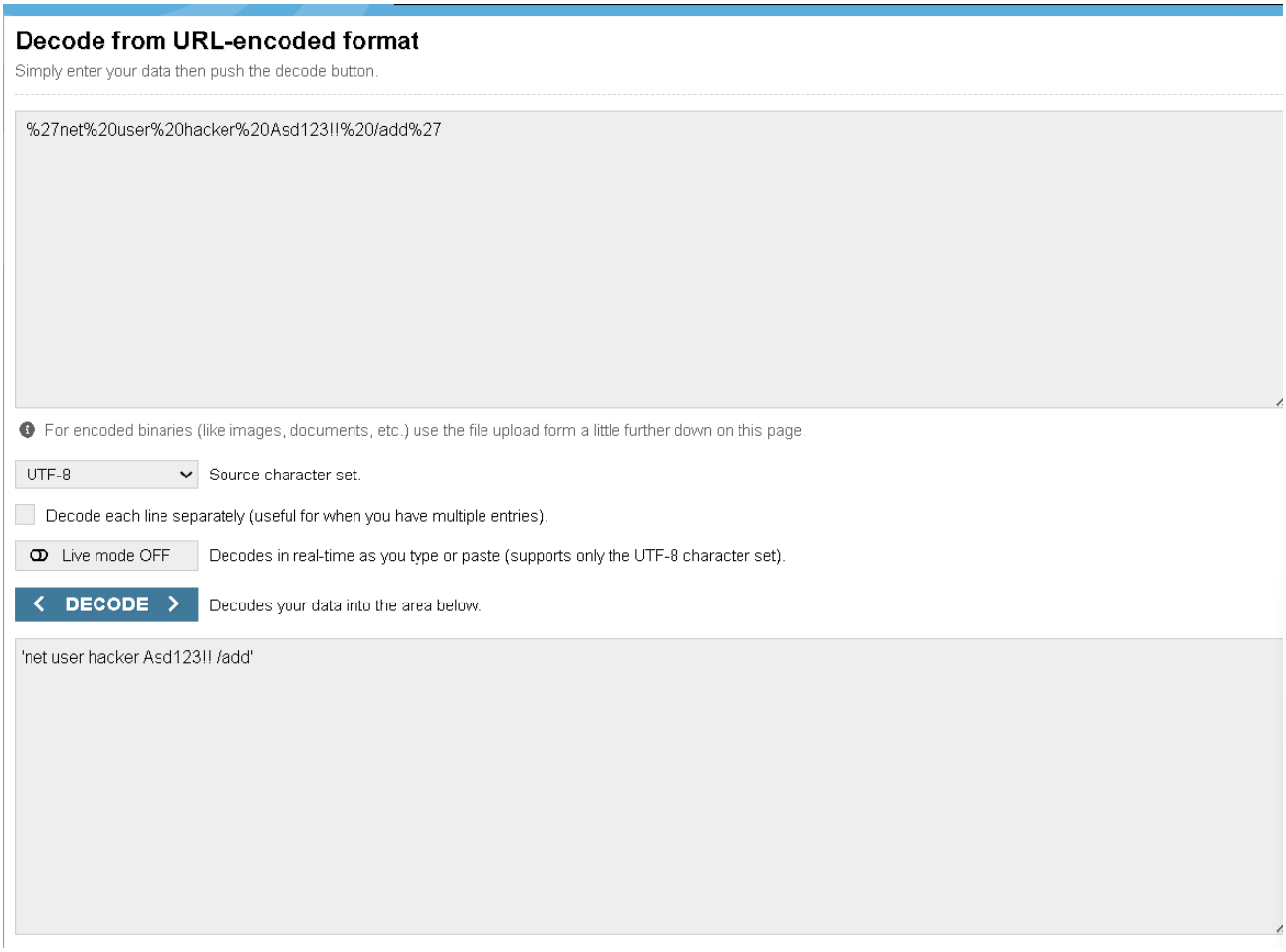
Phase 5: Persistence – User Account Creation

Objective:
Investigate whether the attacker attempted to maintain long-term access to the victim system after successful code execution.

Log Evidence:
Shortly after executing reconnaissance and injection commands, the attacker submitted this payload via a GET request to the vulnerable phpi.php endpoint:

```
12554 192.168.199.2 - - [20/Jun/2021:12:52:46 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27net%20user%27) HTTP/1.1" 200 13045 "-" "Mozilla/5.0 (X11; Linux
      x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12555 192.168.199.2 - - [20/Jun/2021:12:52:56 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27net%20share%27) HTTP/1.1" 200 13175 "-" "Mozilla/5.0 (X11; Linux
      x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12556 192.168.199.2 - - [20/Jun/2021:12:53:13 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27net%20user%20hacker%20Asd123!!%20/add%27) HTTP/1.1" 200 12755 "-"
      "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
12557 192.168.199.2 - - [20/Jun/2021:12:53:23 +0300] "GET /bWAPP/phpi.php?message=%22%22;%20system(%27net%20user%20hacker%20Asd123!!%20/add%27) HTTP/1.1" 200 12755 "-"
```

Decoded Payload:



This command instructs the Windows operating system to create a new local user named hacker with the password Asd123!!. This account could then be used for re-entry, lateral movement, or privilege escalation.

What's the Goal of This Command?
To establish persistence — ensuring the attacker can return later even after losing access to the session or injected script.

Additional Risk:
In real-world cases, the attacker might follow up with:

system('net localgroup administrators hacker /add').

TTPs Identified:

- **Tactic:** Persistence
- **Technique:** T1136.001 – Create Account: Local Account
- **Attack Name:** Local User Creation via Command Injection.

Impact:

Persistence through local account creation allows the attacker to:

- Return at will without repeating the attack chain.

- Operate under a legitimate-looking user context.
- Maintain long-term access even after system reboots.
- Bypass detection tools monitoring temporary processes or malware signatures.

Defensive Recommendations:

- Log and alert on suspicious use of net user and net localgroup commands in web contexts.
- Restrict or disable system-level execution (e.g., system(), exec()) in web application code.
- Enable system auditing (e.g., Windows Event ID 4720 – Account Creation) and forward logs to SIEM.
- Apply principle of least privilege to the application service account.
- Use endpoint protection tools to flag privilege changes or unauthorized user creations.
- Implement User Behavior Analytics (UBA) to detect anomalous access attempts from newly created accounts.

Summary Insight:

This final stage of the attack chain illustrates how attackers systematically blend Initial Access, Execution, and Persistence to achieve full system compromise. By leveraging simple but powerful commands like net user, the attacker transitions from temporary exploitation to sustained control.

What makes this phase especially dangerous is its stealth when user creation is done via trusted binaries and logged under legitimate system behavior, it often evades detection. Without centralized logging, alerting, or privilege monitoring, such activity can silently succeed.

Establishing persistence allows an attacker to:

- Re-enter the system at any time, bypassing the need to repeat initial exploitation.
- Operate under the radar using newly created local accounts.
- Lay the groundwork for privilege escalation, lateral movement, or long-term surveillance.

This reinforces the importance of:

- Implementing strong access control policies.

- Enabling detailed system event logging.
- Correlating suspicious user creation with previous attack phases.
- Integrating log analysis and behavior-based detection into blue team workflows.

In real-world DFIR scenarios, missing these indicators early can result in delayed detection and higher remediation costs. Persistence is the attacker's foothold and defenders must treat it as a red flag for deeper compromise.

Threat Techniques Glossary

This section serves as a quick-reference guide to key techniques observed during the web attack simulation.

1. Nikto

- What It Is:
  Nikto is an open-source web server scanner used to detect known vulnerabilities, outdated software, and common misconfigurations.
- Use Case in Attack:
  Used during the Reconnaissance phase to identify exploitable endpoints and directories.
- Detection Tip:
  Monitor logs for User-Agent strings like Nikto/2.1.6.

2. Directory Brute Force

- What It Is:
  Automated enumeration of hidden directories or files on a web server using wordlists or tools like DirBuster or Gobuster.
- Use Case in Attack:
  Helps attackers locate admin panels, configuration files, or backups that are not publicly linked.
- Detection Tip:
  Repeated 404/403 HTTP responses in logs indicate brute-force probing.

3. Brute Force Login

- What It Is:
  A technique where an attacker tries many username/password combinations in rapid succession to guess valid credentials.
- Use Case in Attack:
  Used to gain access to protected areas such as login portals (/login.php).

- Detection Tip:
  Look for high-volume POST requests with similar size and no parameter variation, followed by a sudden shift from HTTP 200 to HTTP 302.

4. Code Injection

- What It Is:
  Insertion of system-level commands or code into a vulnerable parameter, typically in URL or form inputs.
- Use Case in Attack:
  The attacker used system('whoami') to execute OS-level commands via the phpi.php parameter.
- Detection Tip:
  Watch for unexpected parameters containing OS commands like whoami, net user, etc.

5. whoami

- What It Is:
  A basic OS command that returns the username of the current session.
- Use Case in Attack:
  Used to confirm the attacker's level of access and current privilege context after injection.
- Detection Tip:
  Execution of whoami via web input is a strong sign of successful code execution.

6. Persistence (via Local Account Creation)

- What It Is:
  A tactic to maintain access over time by creating a new user account on the system.
- Use Case in Attack:
  The attacker used net user hacker Asd123!! /add to add a backdoor account for later use.
- Detection Tip:
  Log and alert on execution of net user or net localgroup commands, especially via web paths.

📌 **Why This Matters:**
Understanding these techniques allows defenders to **map attacks using the MITRE ATT&CK framework**, improve detection logic, and design layered controls against real-world adversary behavior.