ATTACKFORGE

PENETRATION TEST

ACME CORP. BAT PORTAL PENTEST

WEBAPP1001
Saturday, May 24, 2025

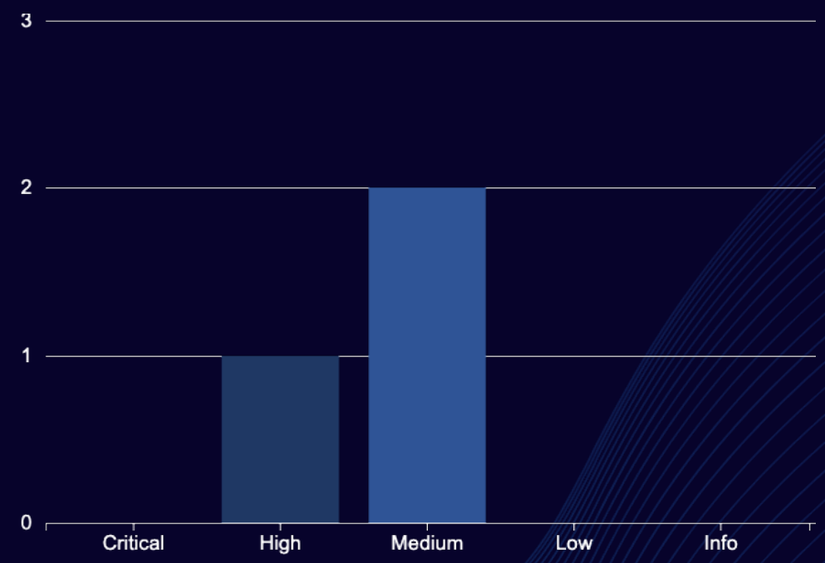
ATTACKFORGE

TESTING SUMMARY

Start	Jan 14 2024	Tested	100.00%
End	Jan 24 2024	In Progress	0%
Completed	100%	Not Tested	0%
Total Testcases	127	Not Applicable	0%

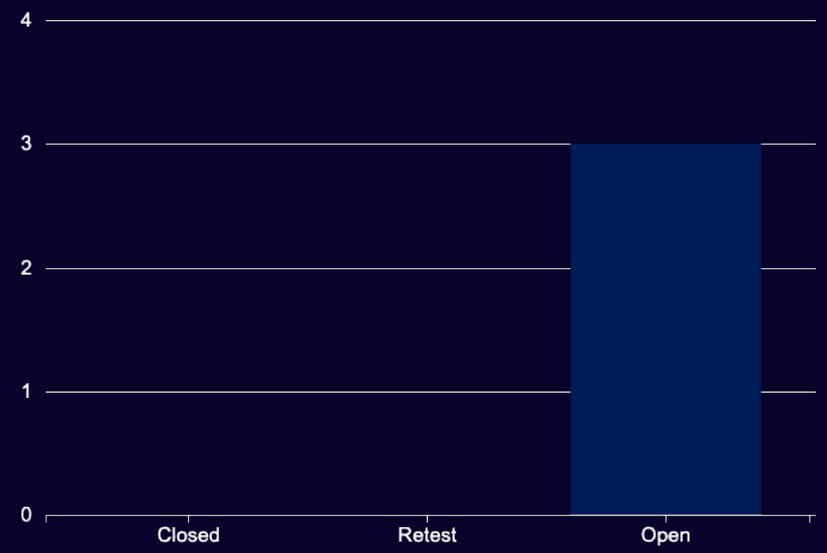
UNIQUE VULNERABILITIES

Total	3
Critical	0
High	1
Medium	2
Low	0
Info	0



REMEDIATION PROGRESS

Open	3
Retest	0
Closed	0



SCOPE

In	bat-api.attackforge.com, bat-portal.attackforge.com	Out	The following items considered out-of-scope for this assessment: • Any APIs following /integrations/... • Testing from the Admin user role • ...

ATTACKFORGE

EXECUTIVE SUMMARY

Objective

The objective of testing was to assess the security posture for the new AttackForge Bat Portal, from the perspective of an external self-registered user.

Approach

The following scenario's were assessed:

- Attacker has self-registered an account on the AttackForge Bat Portal.
- Attacker has access to compromised staff user credentials for an account on the AttackForge Bat Portal.

Summary

AttackForge was able to compromise the application, and subsequently gain access to the internal corporate network. From here, AttackForge was able to gain access to credit card data within the Secure Payments Area (SPA) within Cardholder Data Environment (CDE).

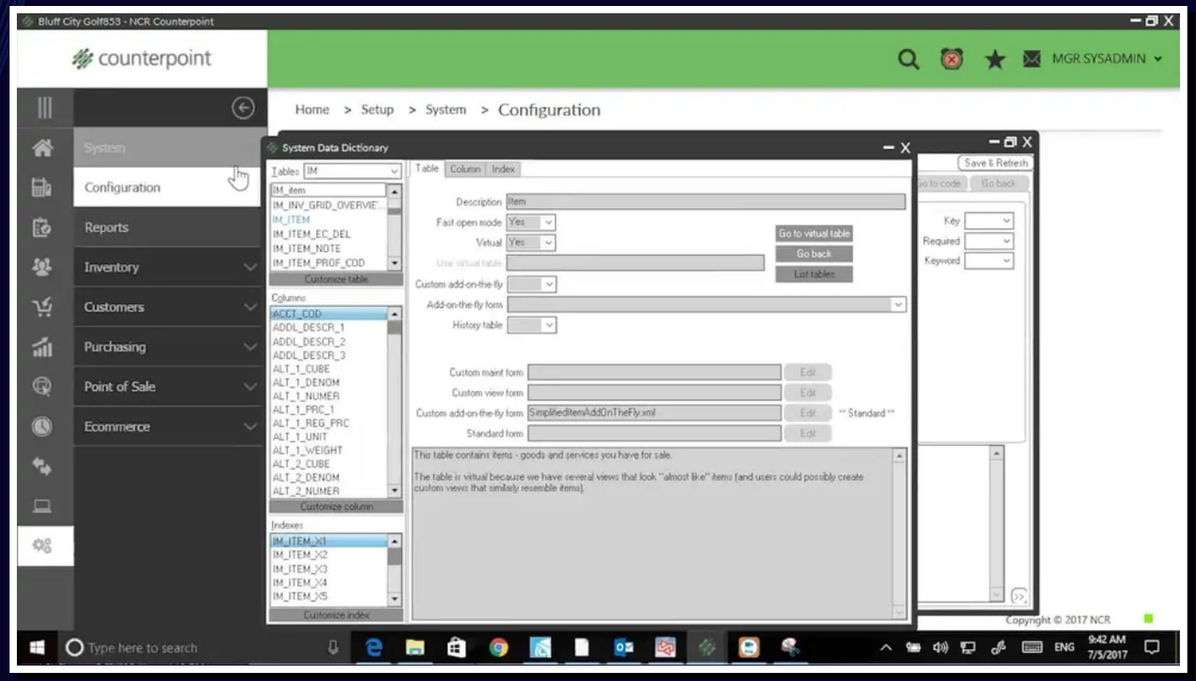


Figure 1: Access to Credit Card Data in SPA

ATTACKFORGE

SUMMARY OF RECOMMENDATIONS

The following recommendations are made to <CUSTOMER> as a result of this assessment:

Recommendation 1: Do this thing..

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ipsum augue, finibus id pretium sit amet, semper nec lorem. Nam bibendum arcu sed odio iaculis, a pretium eros ornare. Suspendisse in imperdiet ipsum, sed congue nisi. In ullamcorper feugiat bibendum. Vestibulum eu diam sed diam ultrices cursus eu vel nisl. Phasellus vestibulum est eu faucibus tempor. Integer id elementum enim, quis tristique est.

Recommendation 2: Do that thing..

Integer neque urna, elementum at nibh ut, ultrices pulvinar lacus. Donec eget turpis porttitor lectus laoreet euismod. Vivamus suscipit gravida metus vitae pellentesque. Ut maximus dictum mi, ut accumsan nulla maximus et. Vestibulum auctor quis nulla pulvinar eleifend. Aliquam aliquam iaculis est blandit dapibus. Sed posuere ipsum sed consectetur ultrices. Vestibulum feugiat vulputate magna eget commodo.

ATTACKFORGE

POSITIVE SECURITY OBSERVATIONS

The following positive security observations were observed during this assessment:

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Nullam ipsum augue, finibus id pretium sit amet, semper nec lorem.
- Nam bibendum arcu sed odio iaculis, a pretium eros ornare.
- Suspendisse in imperdiet ipsum, sed congue nisi.

ATTACKFORGE

SUMMARY FINDINGS

PRIORITY	VULNERABILITY	REMEDIATION STATUS
HIGH	Unrestricted Upload of File with Dangerous Type	
MEDIUM	Inconsistent Access Control	
MEDIUM	Relative Path Traversal	

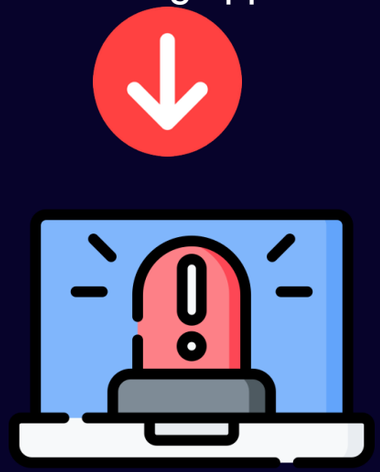
ATTACKFORGE

ATTACKCHAINS

1. ***Gain control of core web server to further pivot attack into ACME Corp. internal network.***

External Attacker

Attacker who has self-registered account on ACME Corp. Bat Portal Internet-facing application.



Action

Log into application and enumerate vulnerable file-upload functionality within the application.

Exploit High Vulnerability

Attacker identifies vulnerable upload functionality in MyProfile and uploads web shell.

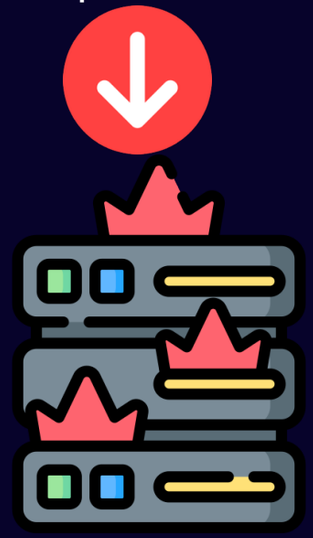
ATTACKFORGE

Action

Search for ways to trigger web shell.

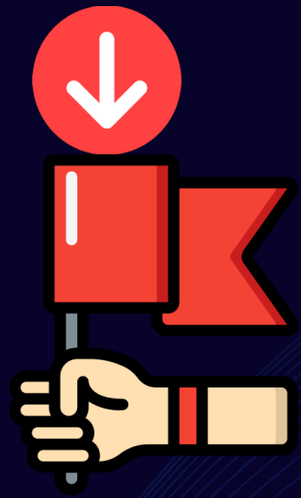
Exploit Medium Vulnerability

Attacker enumerates server directory structure to navigate directly to uploaded web shell.



Target Server

Attacker triggers web shell, elevates to full shell, then creates back door in web server for persistent remote access.



Captured Flag

Operating-System access to compromised ACME Corp. Bat Portal web server allowing further attack into ACME Corp. internal network.

ATTACKFORGE

VULNERABILITIES

1. Unrestricted Upload of File with Dangerous Type

CVSSv3 SCORE	
Base	8.5
Temporal	8.5
Environmental	8.6
Vector	CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:H

DESCRIPTION

An unrestricted upload of files with dangerous type often occurs in applications that explicitly trust application users will submit files of specific type and content only. Uploaded files are then either passed to other internal components for further processing or remain stored for future use within the application in an easily guessable location.

In a typical attack scenario, an attacker discovers the upload functionality in the application and submits a specifically crafted file that embeds malicious content, e.g. virus, exploit or shell code. From this point the attacker either passively waits until the malicious content is accessed and executed by an internal component, other system or user, or if the file is stored in a discoverable location the attacker tries triggering file execution within application by leveraging application mapping of known file types to specific execution routines. An insider or an attacker who can get administrative access to application can upload a web shell, then upload a normal shell and escalate privileges. The attack can be further propagated to other hosts on same network segment.

ATTACK SCENARIO

Arbitrary code execution is possible if an uploaded file is interpreted and executed as code by the recipient. This is especially true for .asp and .php extensions uploaded to web servers because these file types are often treated as automatically executable, even when file system permissions do not specify execution.

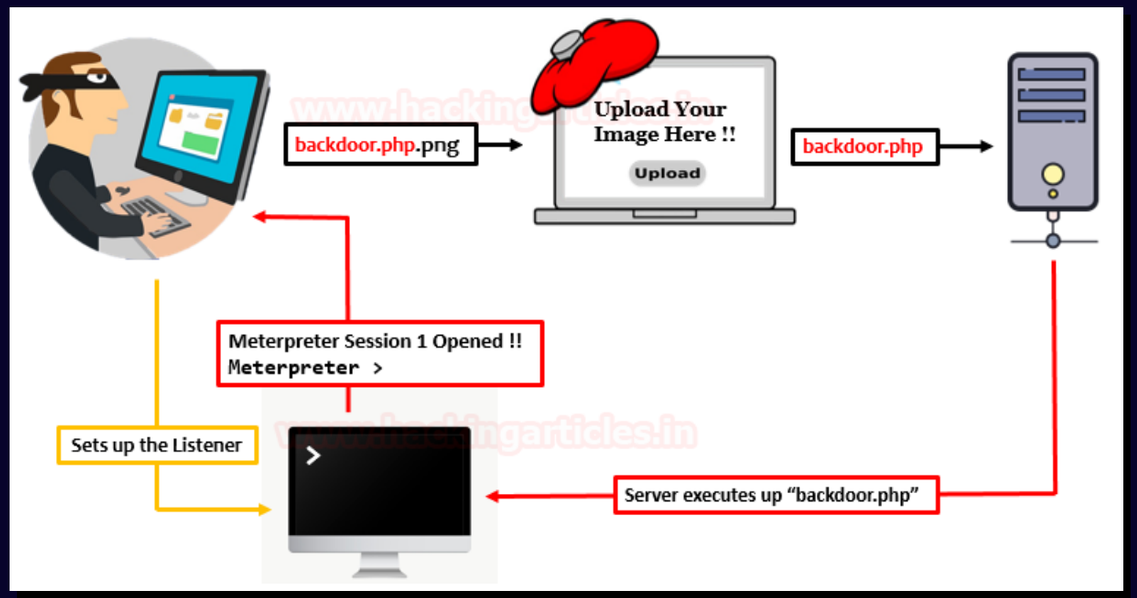


Figure 2

REMEDIATION RECOMMENDATION

1. Assume all input is malicious. Use an 'accept known good' input validation strategy, i.e. use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does.
2. Generate a new, unique filename for an uploaded file instead of using the user-supplied filename, so that no external input is used at all.

3. Define a very limited set of allowable extensions and only generate filenames that end in these extensions.
4. Consider the possibility of XSS (CWE-79) before allowing .html or .htm file types.
5. Ensure that only one extension is used in the filename. Some web servers, including some versions of Apache, may process files based on inner extensions so that 'filename.php.gif' is fed to the PHP interpreter.
6. For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.
7. Do not rely exclusively on the MIME content type or filename attribute when determining how to render a file. Validating the MIME content type and ensuring that it matches the extension is only a partial solution.
8. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). A blacklist is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.
9. For example, limiting filenames to alphanumeric characters can help to restrict the introduction of unintended file extensions.

ATTACKFORGE

AFFECTED ASSET	
Open	bat-portal.attackforge.com
POC	1. Do this... 2. Do that... <some script>...<do something>...</some script>

ATTACKFORGE

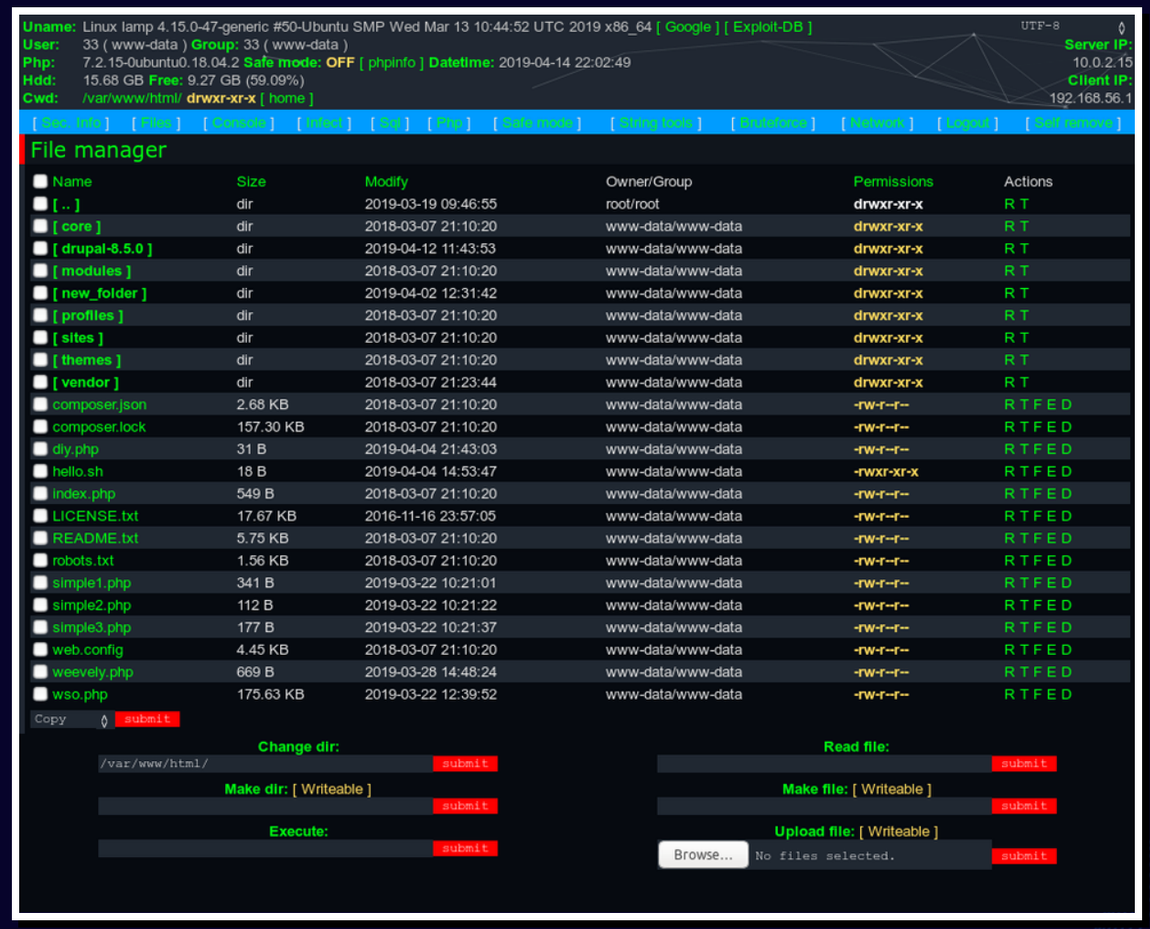


Figure 3

ATTACKFORGE

2.Inconsistent Access Control

CVSSv3 SCORE	
Base	6.5
Temporal	6.5
Environmental	6.5
Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:L

DESCRIPTION

It appears that the application does not consistently apply access controls to all its resources. The lack of access protection for some sensitive resources can be leveraged by an non-authorised attacker to either gather important information for a consequent attack against other application users, or to access and modify directly unprotected application data.

Assuming a user with a given identity, authorisation is the process of determining whether that user can access a given resource, based on the user's privileges and any permissions or other access-control specifications that apply to the resource.

When access control checks are not applied consistently - or not at all - users are able to access data or perform actions that they should not be allowed to perform. This can lead to a wide range of problems, including information exposures, denial of service, and arbitrary code execution.

ATTACK SCENARIO

The page can be identified quick and easily through application fingerprinting and crawling. An attacker could read sensitive data, either by reading the data directly from a data store that is not properly restricted, or by accessing insufficiently-protected, privileged functionality to read the data.

REMEDIATION RECOMMENDATION

This issue should be fixed by applying proper authorisation permission to the affected resources unless it is not an intended business feature.

- For web applications, make sure that the access control mechanism is enforced correctly at the server side on every page. Users should not be able to access any unauthorised functionality or information by simply requesting direct access to that page. One way to do this is to ensure that all pages containing sensitive information are not cached, and that all such pages restrict access to requests that are accompanied by an active and authenticated session token associated with a user who has the required permissions to access that page.
- Ensure that you perform access control checks related to your business logic. These checks may be different than the access control checks that you apply to more generic resources such as files, connections, processes, memory, and database records. For example, a database may restrict access for medical records to a specific database user, but each record might only be intended to be accessible to the patient and the patient's doctor.
- Reduce the attack surface by carefully mapping roles with data and functionality. Use role-based access control (RBAC) to enforce the roles at the appropriate boundaries. Note that this

approach may not protect against horizontal authorisation, i.e., it will not protect a user from attacking others with the same role.

AFFECTED ASSET	
Open	bat-api.attackforge.com, bat-portal.attackforge.com
NOTES	During testing, it was possible to iterate over 100k users in a short amount of time using scripts. These scripts were successful in scraping user details such as: • First name • Last name • Email address • Home address • Work address
POC	1. Open a web browser in private/incognito mode 2. Navigate to https://bat-portal.attackforge.com/api/users/1 3. Notice that you are able to view all user information. Increase '1' to view another user.

ATTACKFORGE

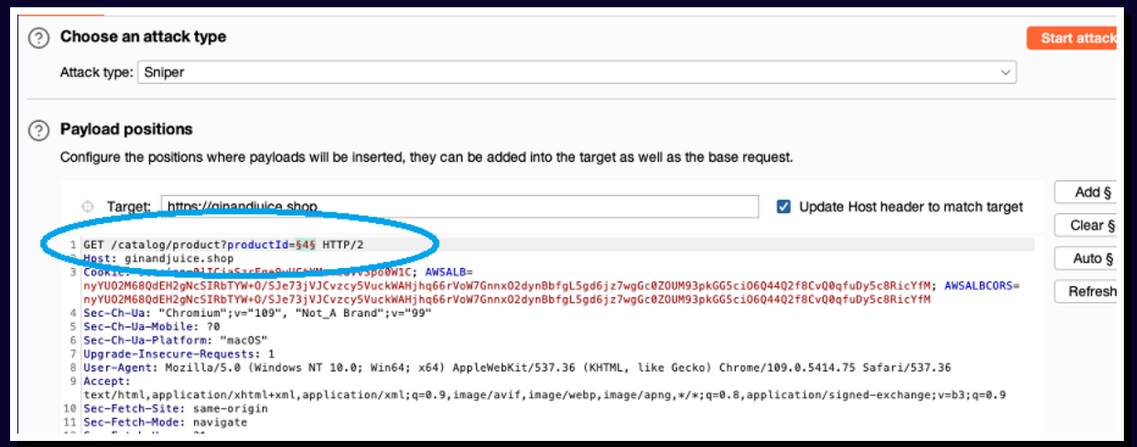


Figure 4: Discover endpoint to enumerate users

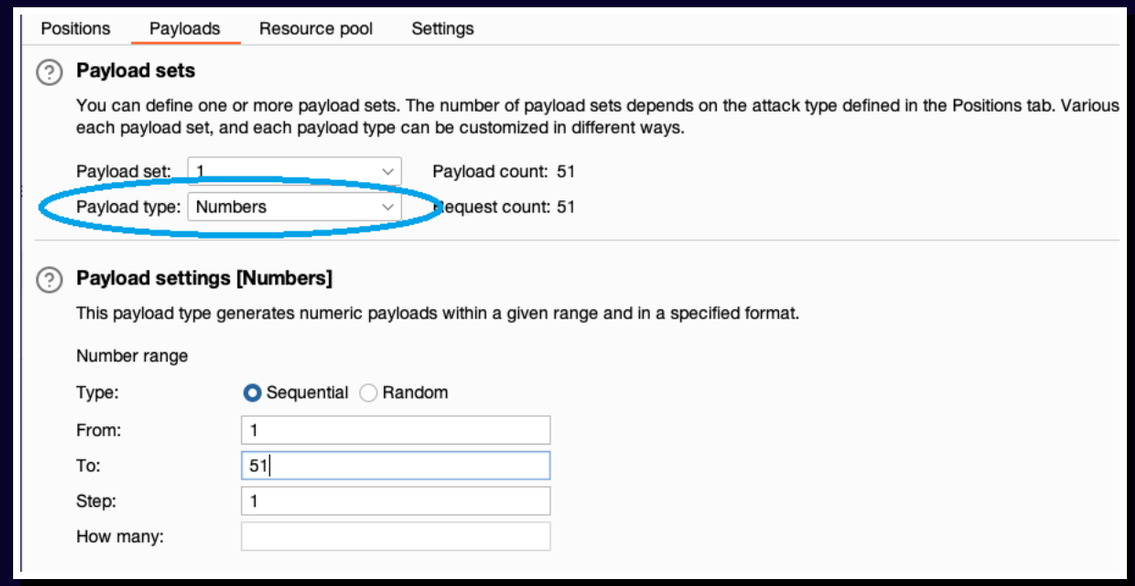


Figure 5: Set up automation parameters

ATTACKFORGE

Results | Positions | Payloads | Resource pool | Settings

Filter: Showing all items

Request	Payload	Status	Error	Redirects followed	Timeout
13	13	404	☐	0	☐
0		200	☐	0	☐
1	1	200	☐	0	☐
2	2	200	☐	0	☐
3	3	200	☐	0	☐
4	4	200	☐	0	☐
5	5	200	☐	0	☐
6	6	200	☐	0	☐
7	7	200	☐	0	☐
8	8	200	☐	0	☐
9	9	200	☐	0	☐
10	10	200	☐	0	☐
11	11	200	☐	0	☐
12	12	200	☐	0	☐

Figure 6: Notice 200 responses are successful

ATTACKFORGE

3.Relative Path Traversal

CVSSv3 SCORE	
Base	4.3
Temporal	4.3
Environmental	4.3
Vector	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

DESCRIPTION

The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize sequences such as '..' that can resolve to a location that is outside of that directory.
This allows attackers to traverse the file system to access files or directories that are outside of the restricted directory.

ATTACK SCENARIO

The attacker may be able to create or overwrite critical files that are used to execute code, such as programs or libraries.
The attacker may be able to overwrite or create critical files, such as programs, libraries, or important data. If the targeted file is used for a security mechanism, then the attacker may be able to bypass

that mechanism. For example, appending a new account at the end of a password file may allow an attacker to bypass authentication. The attacker may be able read the contents of unexpected files and expose sensitive data. If the targeted file is used for a security mechanism, then the attacker may be able to bypass that mechanism.
For example, by reading a password file, the attacker could conduct brute force password guessing attacks in order to break into an account on the system. The attacker may be able to overwrite, delete, or corrupt unexpected critical files such as programs, libraries, or important data. This may prevent the software from working at all and in the case of a protection mechanisms such as authentication, it has the potential to lockout every user of the software.

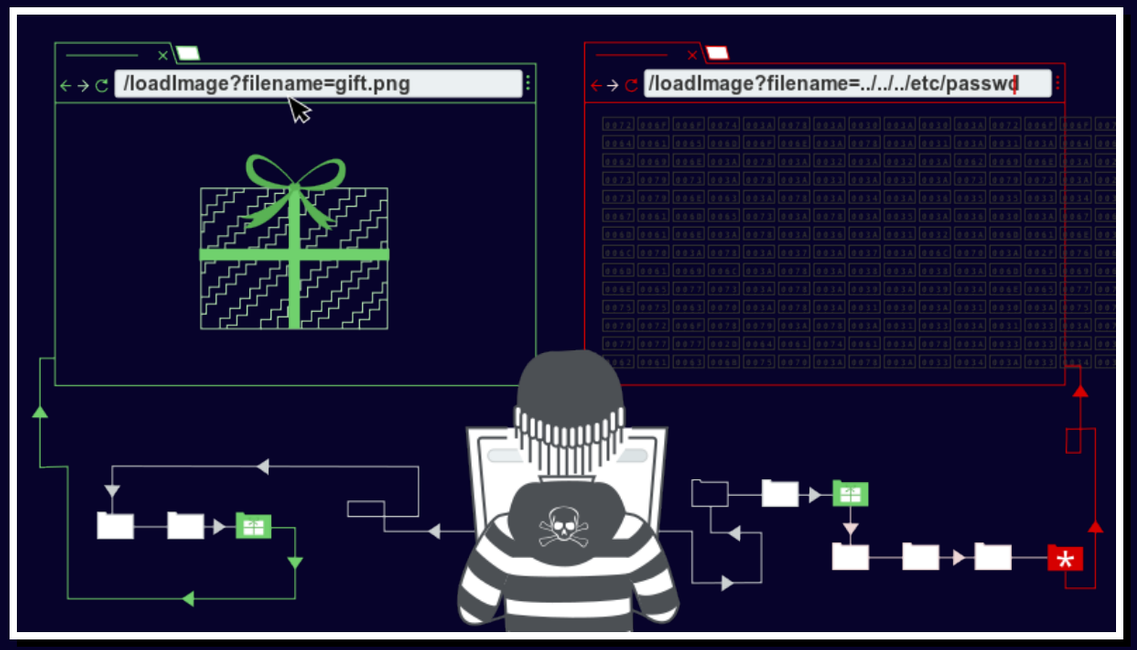


Figure 7

REMEDIATION RECOMMENDATION

Assume all input is malicious. Use an 'accept known good' input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does.

When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, 'boat' may be

syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colours such as 'red' or 'blue.'
Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). A blacklist is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When validating filenames, use stringent whitelists that limit the character set to be used. If feasible, only allow a single '.' character in the filename to avoid weaknesses such as CWE-23, and exclude directory separators such as '/' to avoid CWE-36. Use a whitelist of allowable file extensions, which will help to avoid CWE-434. Do not rely exclusively on a filtering mechanism that removes potentially dangerous characters. This is equivalent to a blacklist, which may be incomplete (CWE-184).
For example, filtering '/' is insufficient protection if the filesystem also supports the use of '' as a directory separator. Another possible error could occur when the filtering is applied in a way that still produces dangerous data (CWE-182). For example, if '../' sequences are removed from the '.../.../' string in a sequential fashion, two instances of '../' would be removed from the original string, but the remaining characters would still form the '../' string.

AFFECTED ASSET	
Open	bat-portal.attackforge.com
POC	Authenticate to the portal as administrator user and browse to the following URL:

https://globexcorp.com.au/test/cmsedit.jsp?file=../../../../../../../../etc/hostname

Note that the resulting page will contain the hostname of the underlying system.

Affected

- URL: https://globexcorp.com.au/test/cmsedit.jsp?file=../../../../../../../../etc/hostname
- Parameter: file

ATTACKFORGE

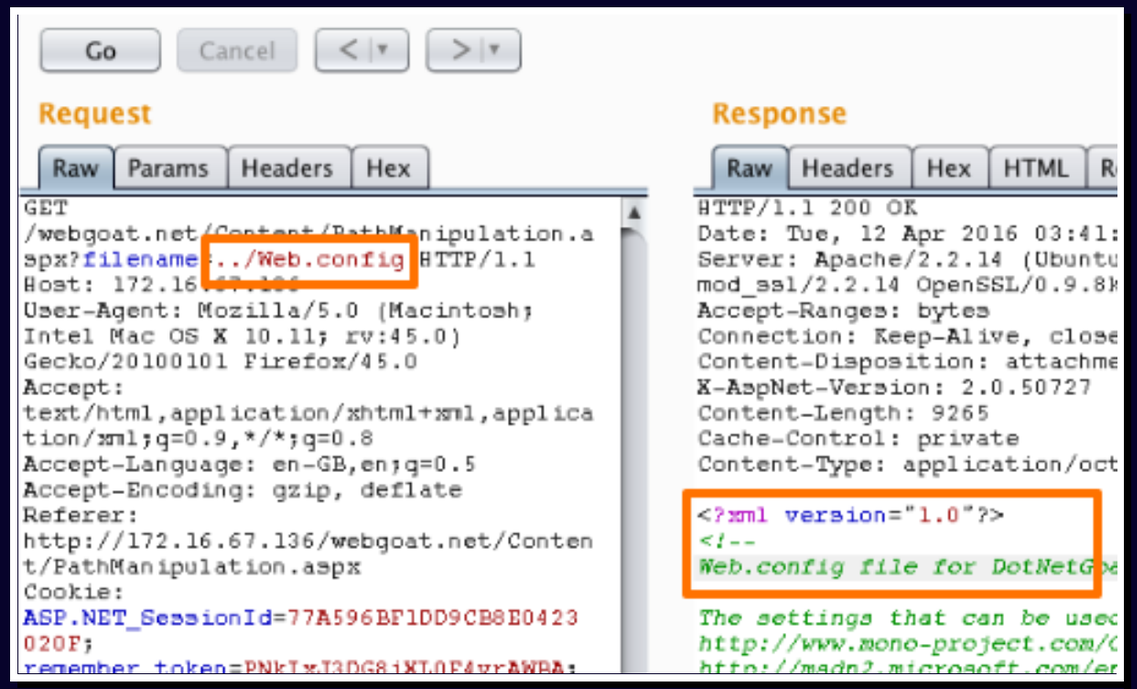


Figure 8: Identified path traversal

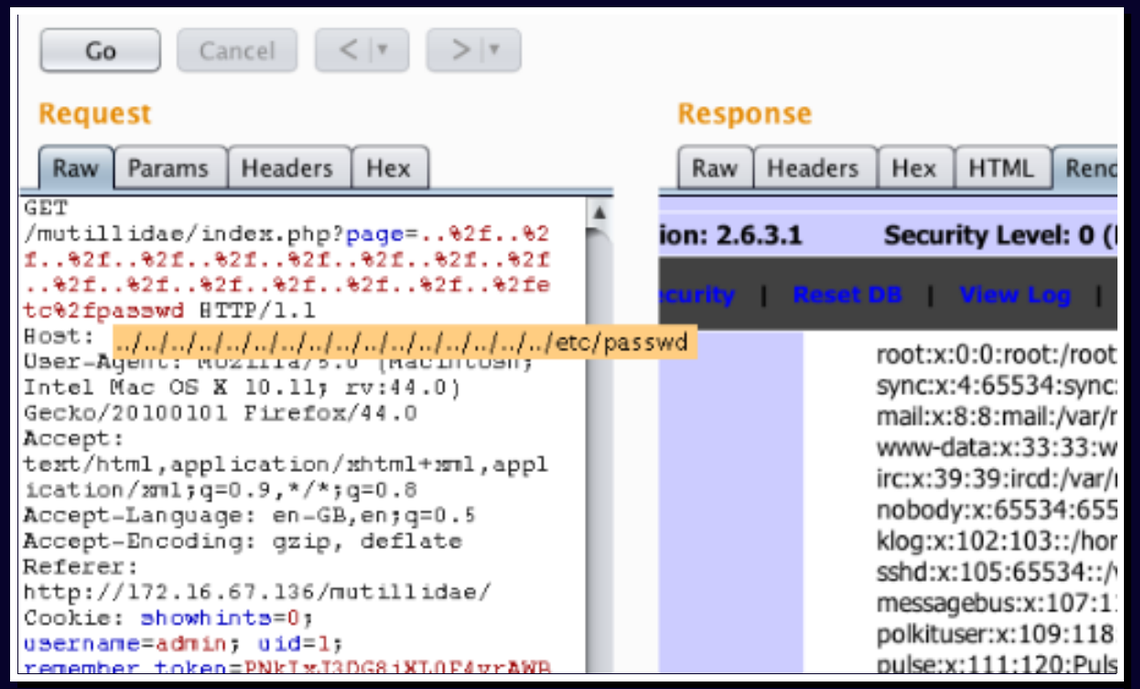


Figure 9: Accessing passwd