Get Event Log

Digital Investigation

PEN & SQA

Cyber Hygiene Assessment

64MM

3/22/2021

Draft 0.1

Get Event Log

Email: chris@geteventlog.com – Web: www.Get Event Log.com

Table of Contents

Document Control

Version	Date	Author	Description
Draft 1.0	3/12/2021	CSW	Initial Draft
Final 1.0	3/14/2021	CSW	Final Doc

What to Do With This Report

Welcome to your Cyber Hygiene report. This document aims to be a comprehensive weekly/monthly/yearly snapshot of known vulnerabilities detected on Internet-facing hosts for 64MM.

While it is not our intent to prescribe to you a particular process for remediating vulnerabilities, we hope you will use this report to strengthen your security posture. Here is the basic flow:

1. Review the Cyber Hygiene Report Card for a high-level overview. This section gives a quick comparison of the problems we find on the initial report or from report to report. If this is your first report, you should note that the Report Card will initially lack historical data to make comparisons against, though that data will exist in your next report.

2. See Appendix A: Vulnerability Summary for a list of unique vulnerabilities across all the systems we detect problems with. Appendix C: Detailed Findings and Recommended Mitigations by Vulnerability provides more information about each vulnerability and all the hosts that we detect are susceptible to a given vulnerability. You should focus on those vulnerabilities rated with the greatest severity, as well as those that impact your high-value assets, but do not ignore the medium or low vulnerabilities. Recognize that a vulnerability's rating tends to get worse with time.

3. If this report is not your first, review Appendix B: Vulnerability Changes Since Last Report for a breakdown of all the changes we detected in your scope in the last week.

4. If you have patched a vulnerability since your last report, verify it is listed here. If it is not present, there may still be an issue. It may also be possible that the issue was fixed after our latest scan, which was on [date of last scan].

As you review the report, you may have additional questions. Check out the answers we provide in the Frequently Asked Questions section. If you have any additional questions, email us here.

64MM Contacts

Name	Position	Email	Phone
Surge Protector	CEO	sp@64mm.com	15551234
Monty Burns	CTO	Mb3@64mm.com	15551515
Marge Simpson	Security Manager	marge@64mm.com	15552112

Cyber Hygiene Report Card

High Level Findings

Addresses Scan Dates(s)		
Vulnerabilities Scan Date(s)	03-08-2021 to 03-12-2021	
Addresses Owned	1	72.167.241.134
Addresses Scanned	1	72.167.241.134
Hosts	1	https://64mm.com
Vulnerable Hosts	1	https://64mm.com
Vulnerabilities	22	

Vulnerabilities

Type	**Number**	**Resolved**	**New**
HIGH	17		17
MEDIUM			
LOW	5		5

Vulnerability Response Time (since last report if applicable)

	HIGH		MEDIUM		LOW	
	Median	Maximum	Median	Maximum	Median	Maximum
Days to Mitigate						
Days Currently Active						

Executive Summary

This report provides the results of a Get Event Log assessment of 64MM conducted from March 8, 2021 at 15:54 UTC through March 12, 2021 at 03:53 UTC. The Cyber Hygiene assessment includes network mapping and vulnerability scanning for Internet-accessible 64MM hosts. This report is intended to provide 64MM with enhanced understanding of their cyber posture and to promote a secure and resilient Information Technology (IT) infrastructure across 64MM's Internet accessible networks and hosts.

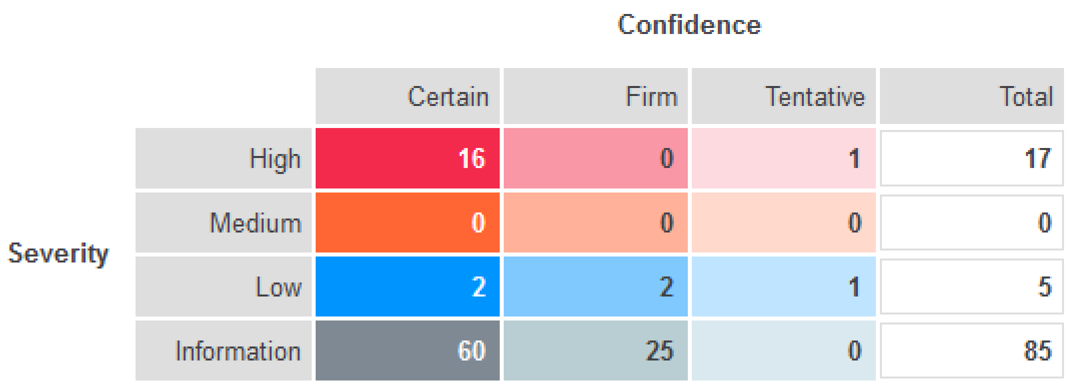
For this reporting period, a total of 1 host were identified out of the 1 address provided to Get Event Log. The scanning revealed 22 total potential vulnerabilities on 1 vulnerable host, 100% of all 64MM hosts. 3 distinct open ports, 1 distinct service, and 1 operating system detected.

22 distinct types of potential vulnerabilities (17 high, 0 medium, and 5 low) were detected, as shown in Table 1. The vulnerabilities that were detected most frequently on 64MM hosts are displayed in Figure 1.

64MM should review the potential vulnerabilities detected and report any false positives back to Get Event Log so they can be excluded from future reports. Refer to Appendix A: Vulnerability Summary for an illustration of the breakdown of vulnerability occurrences over time.

Severity	Distinct Vulnerabilities	Total Vulnerabilities
High	1	17
Medium	0	0
Low	3	5
Total	4	22

Table 1: Number of Vulnerabilities by Severity Level



		Confidence			
		Certain	Firm	Tentative	Total
Severity	High	16	0	1	17
	Medium	0	0	0	0
	Low	2	2	1	5
	Information	60	25	0	85

Figure 1: Top Vulnerabilities by Occurrence

Additionally, the top high-risk hosts are shown here in table 2.

IP Address	High	Medium	Low
72.167.241.134	17	0	5

Table 2: Top High-Risk Hosts

The most frequently detected operating systems and services for 64MM are displayed in Table 3 and Table 4 respectively.

Operating System	Detections
Linux Server	Ubuntu 20.04.2 LTS \n \l

Table 3: Top Operating Systems Detected

Service	Detections
TCP SSH	OpenSSH 7.5 (protocol 2.0)
TCP http	http-proxy HAProxy http proxy 1.3.1 or later
TCP ssl/http	OpenResty web app server

Table 4: Top Services Detected

Methodology

Background

The Get Event Log team conducted a Cyber Hygiene assessment of 64MM's Internet-facing networks and hosts from March 8, 2021 at 15:54 UTC through March 12, 2021 at 03:53 UTC. This report provides result summaries and detailed findings of the assessment activity for 64MM and its associated sub-organizations. All scan results are included in Appendix G: Attachments.

Cyber Hygiene is intended to improve your security posture by proactively identifying and reporting on vulnerabilities and configuration issues present on Internet-facing systems before those vulnerabilities can be exploited.

Cyber Hygiene is a service of Get Event Log, organized under 64MM, Office of Cybersecurity.

Upon submission of an Acceptance Letter, 64MM provided Get Event Log with their public network address information. 64MM and Get Event Log agreed on any time restrictions which would be imposed on the scanning activity.

Process

All Cyber Hygiene scanning activity originates from the 67.190.232.1/24 network.

Get Event Log uses a combination of scanning services for testing:

- Network Mapping
- Vulnerability Scanning

Network Mapping

Using Nmap, we attempt to determine which hosts are available, identify what services (application name and version) those hosts are offering, and what Operating System (OS) versions they are running. We first scan the most detected 1,000 Transmission Control Protocol (TCP) ports of the addresses you have submitted to us to get a quick understanding of the active/dark landscape. An address that has a least one port open/listening service is considered a host and is then fully port-scanned (TCP) and included in the vulnerability scan. For the purposes of this report, tcpwrapped ports are not considered to be open; for more information on tcpwrapped ports, refer here.

If no services are detected in the most common 1,000 ports on a given Internet Protocol (IP) address, that address is considered “dark” in Get Event Log and will be re-scanned after at least 90 days to check for change. Addresses marked dark are not included in the host count of the weekly report. Understand that Get Event Log is not attempting to make a judgment call about why an address is unresponsive. If there is not a port open, it is not a host in the language of Get Event Log.

Vulnerability Scanning

Using Burp Suite, a commercial vulnerability scanner, each host is evaluated against a library of vulnerabilities that an Internet-based actor could exploit. Vulnerabilities are reported with a severity of critical, high, medium, or low to facilitate prioritization of remediation efforts. We enable all Burp Suite plugins except those in the “Denial of Service” category.

Vulnerability Scoring

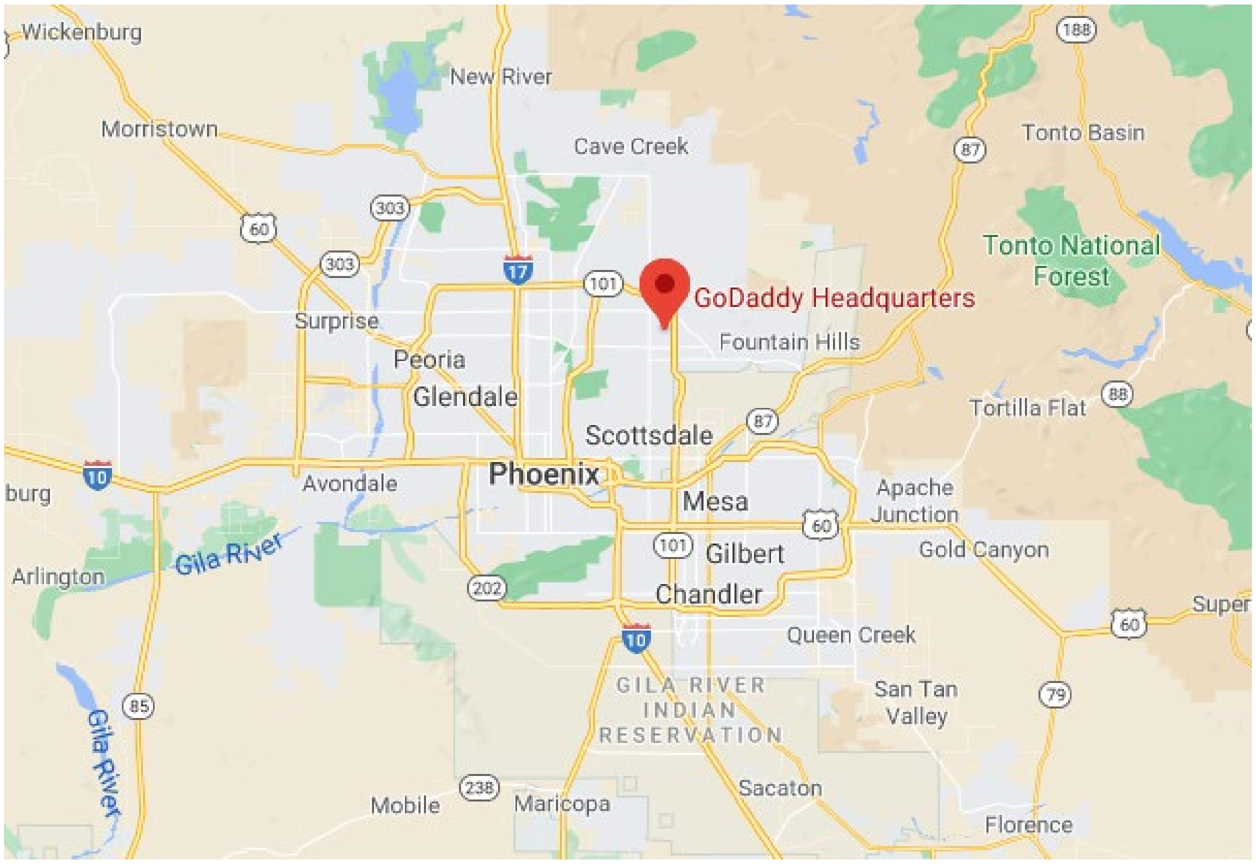
The Burp Suite vulnerability scanner references the National Vulnerability Database for its vulnerability information. The NVD provides CVSS scores for many known vulnerabilities. NVD supports the CVSS version standard for all Common Vulnerabilities and Exposures (CVE) vulnerabilities.

The CVSS is a free and open industry standard for assessing the severity of computer system security vulnerabilities. CVSS attempts to assign severity scores to vulnerabilities, allowing responders to prioritize responses and resources according to threat. The NVD uses severity rankings of “Low,” “Medium,” and “High” in addition to the numeric CVSS scores, but these qualitative rankings are simply mapped from the numeric CVSS base scores:

- Vulnerabilities are labeled “Low” severity if they have a CVSS base score of 0.0-3.9.
- Vulnerabilities will be labeled “Medium” severity if they have a base CVSS score of 4.0-6.9.
- Vulnerabilities will be labeled “High” severity if they have a CVSS base score of 7.0-10.0.

Approximate Host Locations

The map below shows the approximate locations of detected hosts as listed in a geo-location database. This map is provided as a tool to identify hosts that may have been mistakenly added in to or removed from scope. The map is scaled to include all known 64MM host locations.



Vulnerability Scan Results

For this period, Get Event Log detected 22 occurrences of 4 distinct vulnerabilities (0 critical, 17 high, 0 medium, and 5 low). 64MM should review the vulnerabilities detected and report any false positives back to Get Event Log so these can be excluded from future reports.

The scanning detected 1 vulnerable host—1 host with one to five vulnerabilities were identified; 1 host had between six and nine vulnerabilities; 1 host had ten or more vulnerabilities identified.

Severity	Distinct Vulnerabilities	Total Vulnerabilities
HIGH	1	17
MEDIUM	0	0
LOW	4	5
Total	5	22

Table 5: Number of Vulnerabilities by Severity Level

Top Vulnerabilities According to CVSS Score

Vulnerability Name	Severity	Hosts	CVSS Score
Cross-site scripting (DOM-based)	High	72.167.241.134	8.0
Cross-origin resource sharing: arbitrary origin trusted	High	72.167.241.134	7.1
Vulnerable JavaScript dependency	Low	72.167.241.134	3.9

Table 6: Top Vulnerabilities by CVSS

A complete list of distinct vulnerabilities detected, including severity level and number of hosts having the vulnerability can be found in Appendix A: Vulnerability Summary. Full details on every detected vulnerability can be found in Appendix C: Detailed Findings and Recommended Mitigations by Vulnerability. Every critical and high finding detected, along with the hosts that have these findings, are listed in Appendix D: Critical and High Vulnerability Mitigations by IP Address.

The top high-risk hosts are identified in Table 7 by combining the total number of vulnerabilities, the severity of the vulnerabilities, and a weighted CVSS score for vulnerabilities detected. For more information on the formula, please refer to Table 8: Risk Rating System.

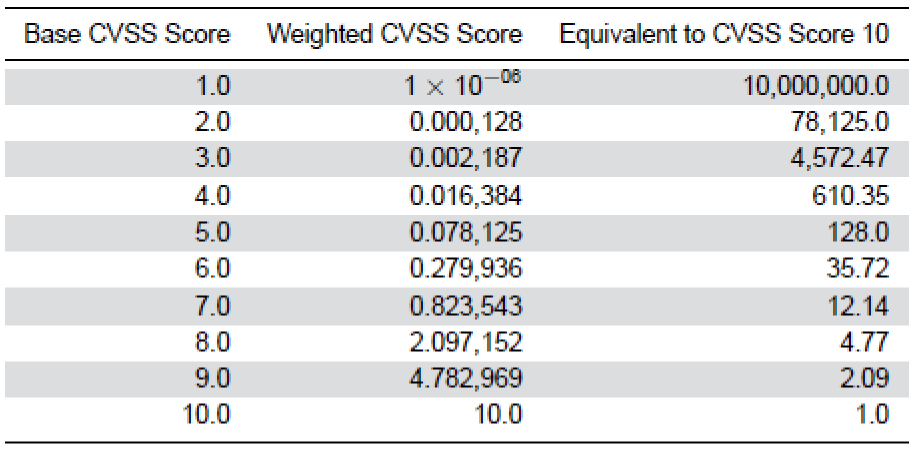
IP Address	High	Medium	Low	Total
72.167.241.134	17	0	5	22

	🟥	🟨	🟦	
	🟥	🟨	🟦	

Table 7: Top Hosts by Weighted Risk

The Risk Rating System (RRS) emphasizes higher-rated CVSS scores to ensure that hosts with many lower-risk vulnerabilities do not outweigh hosts with a smaller number of high-risk vulnerabilities, while ensuring that hosts with an extreme number of low-risk vulnerabilities are not overshadowed by hosts with a single higher-risk issue. The RRS also ensures that hosts with a significant number of high-risk vulnerabilities will not be overshadowed by a host with only a single critical vulnerability.

Table 8 illustrates the base and weighted CVSS scores and shows the equivalent number of lower-risk vulnerabilities to weigh evenly with a single critical (CVSS score of 10) vulnerability.



Base CVSS Score	Weighted CVSS Score	Equivalent to CVSS Score 10
1.0	1×10^{-06}	10,000,000.0
2.0	0.000,128	78,125.0
3.0	0.002,187	4,572.47
4.0	0.016,384	610.35
5.0	0.078,125	128.0
6.0	0.279,936	35.72
7.0	0.823,543	12.14
8.0	2.097,152	4.77
9.0	4.782,969	2.09
10.0	10.0	1.0

Table 8: Risk Rating System

Results Trending

This section contains numbers from scans over time.

	Previous Report	Current Report	Percent Change
Hosts	N/A	1	0%
Vulnerable Hosts	N/A	1	0%
Distinct Services	N/A	3	0%
Distinct Vulnerabilities	N/A	5	0%
Distinct Operating Systems	N/A	1	0%

Table 9: Comparison with Previous Report

Conclusion

64MM should use the data provided in this report to correct any identified vulnerabilities, configuration errors, and security concerns in your external network perimeter. If 64MM has questions, comments, or concerns about the findings or data contained in this report, please work with your designated technical point of contact when requesting assistance from Get Event Log at chris@geteventlog.com.

Appendix A Vulnerability Summary

This section presents counts of all distinct vulnerabilities that were detected in the latest scans. It shows the name of the vulnerability, the severity level of the vulnerability, and the number of vulnerability detections in the previous report vs. this report. Low, medium, high, and critical vulnerabilities are displayed.

Vulnerability	Severity	Previous	Current	Change
Cross-site scripting (DOM-based)	High	N/A	1	0%
Cross-origin resource sharing: arbitrary origin trusted	High	N/A	17	0%
Vulnerable JavaScript dependency	Low	N/A	1	0%
Password field with autocomplete enabled	Low	N/A	1	0%
Client-side HTTP parameter pollution (reflected)	Low	N/A	1	0%
Unencrypted communications	Low	N/A	1	0%

Appendix B Vulnerability Changes Since Last report

Mitigated Vulnerabilities

This section lists the vulnerabilities that were included on the previous report but were not detected by the latest scans. The table provides the initial detection and mitigation detection dates, plus the number of days it took to mitigate each vulnerability.

Owner	Vulnerability	Severity	Host	Port	Initial Detection	Mitigation Detected	Days to Mitigate
N/A							

New Vulnerabilities Detected

This section lists the new vulnerabilities that were detected for the first time in the latest scans. The table provides the initial detection and latest detection dates for each vulnerability.

Vulnerability	Severity	Host	Port	Initial Detection	Latest Detection
Cross-site scripting (DOM-based)	High	72.167.241.134	443	3/8	3/12
Cross-origin resource sharing: arbitrary origin trusted	High	72.167.241.134	443	3/8	3/12
Vulnerable JavaScript dependency	Low	72.167.241.134	443	3/8	3/12
Password field with autocomplete enabled	Low	72.167.241.134	443	3/8	3/12
Client-side HTTP parameter pollution (reflected)	Low	72.167.241.134	443	3/8	3/12
Unencrypted communications	Low	72.167.241.134	443	3/8	3/12

Re-Detected (Previously Mitigated) Vulnerabilities

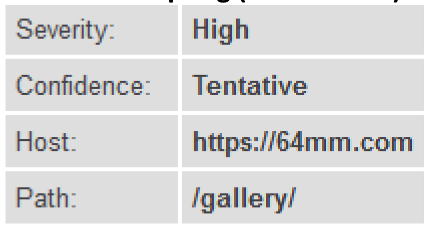
This section lists the vulnerabilities that were previously detected, then mitigated, and were re-detected in the latest scans. The table provides the initial detection and latest detection dates for each vulnerability.

Vulnerability	Severity	Host	Port	Initial Detection	Latest Detection	Days Age
N/A						

Appendix C Detailed Findings and Recommended Mitigations by Vulnerability

This section presents detailed scan results from the network mapping and vulnerability scans. Vulnerabilities identified have a recommended mitigation solution that should be considered to establish or maintain a secure network.

1. **Cross-site scripting (DOM-based)**



Severity:	**High**
Confidence:	**Tentative**
Host:	**https://64mm.com**
Path:	**/gallery/**

Issue detail
The application may be vulnerable to DOM-based cross-site scripting. Data is read from window.location.hash and passed to $().

Note: The exploitability of this issue might depend on the specific version of jQuery that is being used.

Issue background
DOM-based vulnerabilities arise when a client-side script reads data from a controllable part of the DOM (for example, the URL) and processes this data in an unsafe way.

DOM-based cross-site scripting arises when a script writes controllable data into the HTML document in an unsafe way. An attacker may be able to use the vulnerability to construct a URL that, if visited by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application.

The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.

Users can be induced to visit the attacker's crafted URL in various ways, like the usual attack delivery vectors for reflected cross-site scripting vulnerabilities.

Testing automatically identifies this issue using static code analysis, which may lead to false positives that are not actually exploitable. The relevant code and execution paths should be reviewed to determine whether this vulnerability is indeed present, or whether mitigations are in place that would prevent exploitation.
Issue remediation

The most effective way to avoid DOM-based cross-site scripting vulnerabilities is not to dynamically write data from any untrusted source into the HTML document. If the desired functionality of the application means that this behavior is unavoidable, then defenses must be implemented within the client-side code to prevent malicious data from introducing script code into the document. In many cases, the relevant data can be validated on a whitelist basis, to allow only content that is known to be safe. In other cases, it will be necessary to sanitize or encode the data. This can be a complex task and depending on the context that the data is to be inserted may need to involve a combination of JavaScript escaping, HTML encoding, and URL encoding, in the appropriate sequence.

Vulnerability classifications

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
CWE-116: Improper Encoding or Escaping of Output
CWE-159: Failure to Sanitize Special Element

Request 1

```
GET /gallery/ HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Referer: https://64mm.com/
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

Response 1

...[SNIP]...

```
</script>
<script src='https://secureservercdn.net/72.167.241.134/p19.543.myftpupload.com/wp-content/uploads/bb-plugin/cache/162-layout.js?ver=2d6e849f5a3eb5523128b0ca2a39f2bb&#038;time=1616097526' id='fl-builder-layout-162-js'></script>
```

...[SNIP]...

2. **Cross-origin resource sharing: arbitrary origin trusted**
   There are 16 instances of this issue:

   /wp-json/
   /wp-json/oembed/1.0/embed
   /wp-json/wp/v2/categories/28
   /wp-json/wp/v2/pages/156
   /wp-json/wp/v2/pages/160
   /wp-json/wp/v2/pages/162
   /wp-json/wp/v2/pages/19

/wp-json/wp/v2/pages/2
/wp-json/wp/v2/pages/723
/wp-json/wp/v2/pages/739
/wp-json/wp/v2/pages/745
/wp-json/wp/v2/posts/1
/wp-json/wp/v2/tags/19
/wp-json/wp/v2/tags/36
/wp-json/wp/v2/tags/37
/wp-json/wp/v2/users/1

Issue background
An HTML5 cross-origin resource sharing (CORS) policy controls whether and how content running on other domains can perform two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.

Trusting arbitrary origins effectively disables the same-origin policy, allowing two-way interaction by third-party web sites. Unless the response consists only of unprotected public content, this policy is likely to present a security risk.

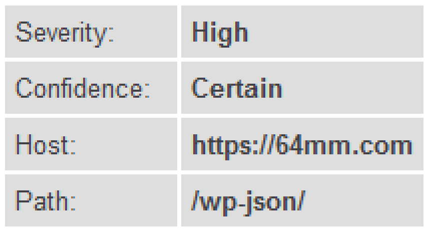
If the site specifies the header Access-Control-Allow-Credentials: true, third-party sites may be able to carry out privileged actions and retrieve sensitive information. Even if it does not, attackers may be able to bypass any IP-based access controls by proxying through users' browsers.

Issue remediation
Rather than using a wildcard or programmatically verifying supplied origins, use a whitelist of trusted domains.

References

[CWE-942: Overly Permissive Cross-domain Whitelist]

2.1



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/**

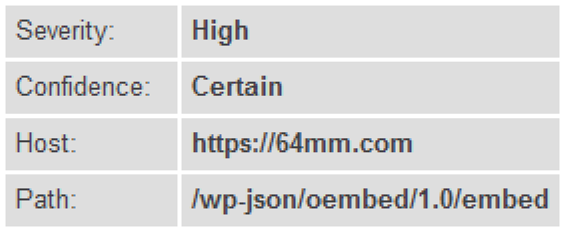
Issue detail
The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://iykyqjenzvgf.com

```
Request 1
ET /wp-json/ HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://iykyqjenzvgf.com
```

```
Response 1
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:14:34 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 99293
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://iykyqjenzvgf.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.2



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/oembed/1.0/embed**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://ekdyqhnjeafs.com**

Request 1

```
GET /wp-json/oembed/1.0/embed?url=https%3A%2F%2F64mm.com%2F HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://ekdyqhnjeafs.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:20:32 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 2169
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://ekdyqhnjeafs.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.3

Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/categories/28**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://ekdyqhnjeafs.com

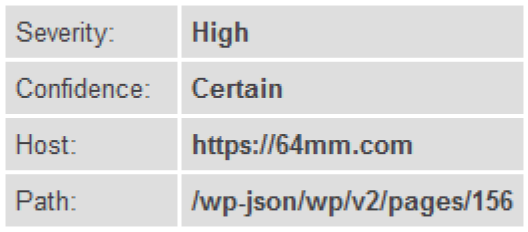
Request 1

```
GET /wp-json/oembed/1.0/embed?url=https%3A%2F%2F64mm.com%2F HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://ekdyqhnjeafs.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:20:32 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 2169
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://ekdyqhnjeafs.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.4



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/pages/156**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://ekdyqhnjeafs.com

Request 1

```
GET /wp-json/oembed/1.0/embed?url=https%3A%2F%2F64mm.com%2F HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://ekdyqhnjeafs.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:20:32 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 2169
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://ekdyqhnjeafs.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.5

Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/pages/160**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://ekdyqhnjeafs.com

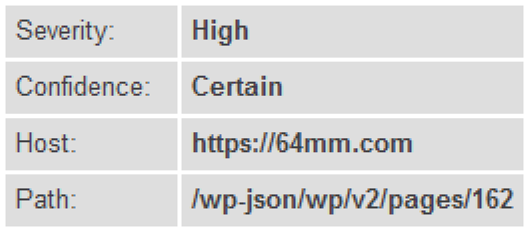
Request 1

```
GET /wp-json/oembed/1.0/embed?url=https%3A%2F%2F64mm.com%2F HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://ekdyqhnjeafs.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:20:32 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 2169
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://ekdyqhnjeafs.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.6



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/pages/162**

Issue Detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://ntqcgfoatidv.com**

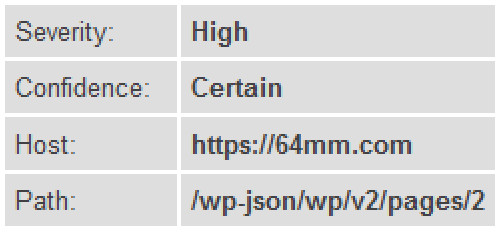
Request 1

```
GET /wp-json/wp/v2/pages/162 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://ntqcgfoatidv.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:10:29 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 1303
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://ntqcgfoatidv.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.7



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/pages/2**

Issue detail
The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://rshouopegqsf.com

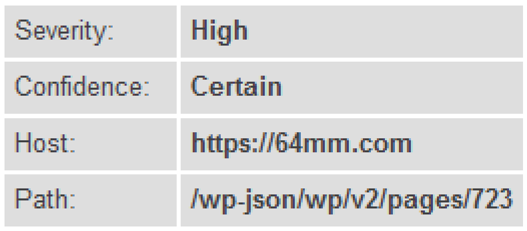
Request 1

```
GET /wp-json/wp/v2/pages/2 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://rshouopegqsf.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:18:30 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 3008
```

2.8



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/pages/723**

Issue Detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://kcwdfyhxuqmc.com

Request 1

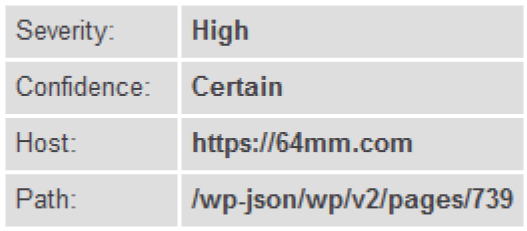
```
GET /wp-json/wp/v2/pages/723 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

```
Origin: https://kcwdfyhxuqmc.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:19:04 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 2723
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://kcwdfyhxuqmc.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.10



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/pages/739**

Issue Detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://vjqygjawuhwi.com

Request 1

```
GET /wp-json/wp/v2/pages/739 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://vjqygjawuhwi.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:08:42 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 2549
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://vjqygjawuhwi.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.11

Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/pages/745**

Issue Details

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://dmgplomqxlot.com

Request 1

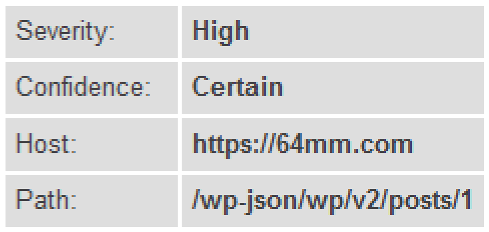
```
GET /wp-json/wp/v2/pages/745 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://dmgplomqxlot.com
```

Response 1

```
HTTP/1.1 200 OK
```

```
Server: openresty
Date: Sat, 20 Mar 2021 18:09:49 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 1374
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://dmgplomqxlot.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.12



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/posts/1**

Issue Details

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://uzbgspwvolkw.com

Request 1

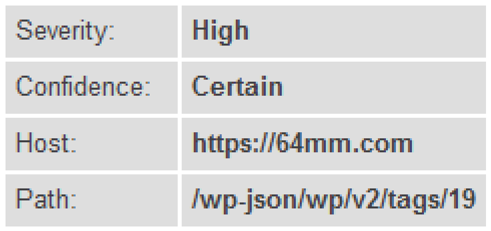
```
GET /wp-json/wp/v2/posts/1 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://uzbgspwvolkw.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:23:07 GMT
```

```
Content-Type: application/json; charset=UTF-8
Content-Length: 3232
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://uzbgspwvolkw.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.13



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/tags/19**

Issue Detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://igochfilnzar.com

Request 1

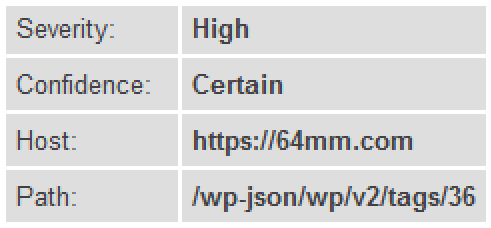
```
GET /wp-json/wp/v2/tags/19 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://igochfilnzar.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:21:41 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 521
Connection: close
```

```
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://igochfilnzar.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.14



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/tags/36**

Issue Detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://pxywojgrtcac.com

Request 1

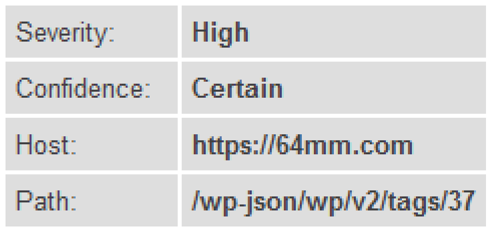
```
GET /wp-json/wp/v2/tags/36 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://pxywojgrtcac.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:18:48 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 527
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
```

```
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://pxywojgrtcac.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.15



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/tags/37**

Issue Detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://qzojmqiyqkap.com

Request 1

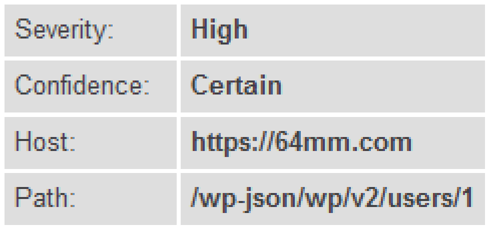
```
GET /wp-json/wp/v2/tags/37 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://qzojmqiyqkap.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:14:05 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 539
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
```

```
Allow: GET
Access-Control-Allow-Origin: https://qzojmqiyqkap.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
```

2.16



Severity:	**High**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-json/wp/v2/users/1**

Issue Detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin https://urjmumhrhecu.com

Request 1

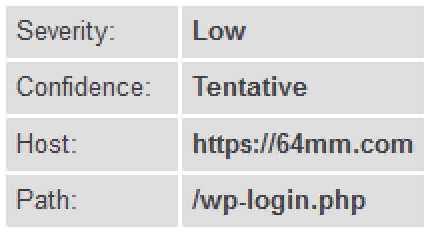
```
GET /wp-json/wp/v2/users/1 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://urjmumhrhecu.com
```

Response 1

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 20 Mar 2021 18:16:28 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 581
Connection: close
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5,
Content-Type
Allow: GET
Access-Control-Allow-Origin: https://urjmumhrhecu.com
```

Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Content-Security-Policy: upgrade-insecure-requests

3. Vulnerable JavaScript dependency



Severity:	**Low**
Confidence:	**Tentative**
Host:	**https://64mm.com**
Path:	**/wp-login.php**

Issue Detail

We observed a vulnerable JavaScript library.

We detected bootstrap version 2.2.1, which has the following vulnerabilities:

CVE-2019-8331: XSS in data-template, data-content and data-title properties of tooltip/popover
CVE-2018-14041: XSS in data-target property of scrollspy
CVE-2018-14040: XSS in collapse data-parent attribute
CVE-2018-14042: XSS in data-container property of tooltip

Issue background

The use of third-party JavaScript libraries can introduce a range of DOM-based vulnerabilities, including some that can be used to hijack user accounts like DOM-XSS.

Common JavaScript libraries typically enjoy the benefit of being heavily audited. This may mean that bugs are quickly identified and patched upstream, resulting in a steady stream of security updates that need to be applied. Although it may be tempting to ignore updates, using a library with missing security patches can make your website exceptionally easy to exploit. Therefore, it's important to ensure that any available security updates are applied promptly.

Some library vulnerabilities expose every application that imports the library, but others only affect applications that use certain library features. Accurately identifying which library vulnerabilities apply to your website can be difficult, so we recommend applying all available security updates regardless.
Issue remediation

Develop a patch-management strategy to ensure that security updates are promptly applied to all third-party libraries in your application. Also, consider reducing your attack surface by removing any libraries that are no longer in use.

Vulnerability classifications

[CWE-1104: Use of Unmaintained Third Party Components]()
[A9: Using Components with Known Vulnerabilities]()

Request 1

```
GET /wp-login.php?redirect_to=https%3A%2F%2F64mm.com%2Fhello-world%2F HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Referer: https://64mm.com/hello-world/
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

Response 1

```
...[SNIP]...
</div>
<script src="//netdna.bootstrapcdn.com/twitter-bootstrap/2.2.1/js/bootstrap.min.js"></script>
...[SNIP]...
```

4. **Password field with autocomplete enabled**

Severity:	**Low**
Confidence:	**Certain**
Host:	**https://64mm.com**
Path:	**/wp-login.php**

Issue detail
The page contains a form with the following action URL:

https://64mm.com/wp-login.php

The form contains the following password field with autocomplete enabled:

pwd

Issue background

Most browsers have a facility to remember user credentials that are entered into HTML forms. This function can be configured by the user and by applications that employ user credentials. If the function is enabled, then credentials entered by the user are stored on their local computer and retrieved by the browser on future visits to the same application.

The stored credentials can be captured by an attacker who gains control over the user's computer. Further, an attacker who finds a separate application vulnerability such as cross-site scripting may be able to exploit this to retrieve a user's browser-stored credentials.
Issue remediation

To prevent browsers from storing credentials entered into HTML forms, include the attribute autocomplete="off" within the FORM tag (to protect all form fields) or within the relevant INPUT tags (to protect specific individual fields).

Please note that modern web browsers may ignore this directive. In spite of this there is a chance that not disabling autocomplete may cause problems obtaining PCI compliance.

Vulnerability classifications

CWE-200: Information Exposure

Request 1
```
GET /wp-login.php?redirect_to=https%3A%2F%2F64mm.com%2Fhello-world%2F HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Referer: https://64mm.com/hello-world/
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

Response 1
```
...[SNIP]...
</h1>
    <form name="loginform" id="loginform" action="https://64mm.com/wp-login.php"
method="post">
        <p>
...[SNIP]...
<div class="wp-pwd">
                <input type="password" name="pwd" id="user_pass" class="input password-input"
value="" size="20" />
                <button type="button" class="button button-secondary wp-hide-pw hide-if-no-js"
data-toggle="0" aria-label="Show password">
...[SNIP]...
```

5. **Client-side HTTP parameter pollution (reflected)**

There are 2 instances of this issue:

/ [name of an arbitrarily supplied URL parameter]
/hello-world/ [name of an arbitrarily supplied URL parameter]

Issue background

Client-side HTTP parameter pollution (HPP) vulnerabilities arise when an application embeds user input in URLs in an unsafe manner. An attacker can use this vulnerability to construct a URL that, if visited by another application user, will modify URLs within the response by inserting additional query string parameters and sometimes overriding existing ones. This may result in links and forms having unexpected side effects. For example, it may be possible to modify an invitation form using HPP so that the invitation is delivered to an unexpected recipient.

The security impact of this issue depends largely on the nature of the application functionality. Even if it has no direct impact on its own, an attacker may use it in conjunction with other vulnerabilities to escalate their overall severity.
Issue remediation

Ensure that user input is URL-encoded before it is embedded in a URL.
References

HTTP Parameter Pollution

Vulnerability classifications
CWE-233: Improper Handling of Parameters
CWE-20: Improper Input Validation

5.1. https://64mm.com/ [name of an arbitrarily supplied URL parameter]

Severity:	**Low**
Confidence:	**Firm**
Host:	**https://64mm.com**
Path:	**/**

Issue detail

The name of an arbitrarily supplied URL parameter is copied into the response within the query string of a URL.

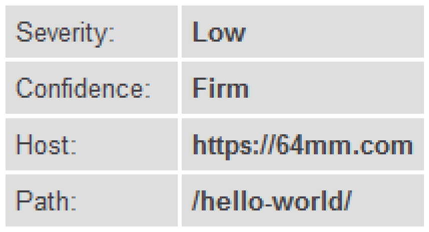
The payload tib&rwy=1 was submitted in the name of an arbitrarily supplied URL parameter. This input was echoed unmodified within the response header Location.

This proof-of-concept attack demonstrates that it is possible to inject arbitrary query string parameters into URLs in the application's response.

Request 1
```
GET /?p=2&tib%26rwy%3d1=1 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

Response 1
```
HTTP/1.1 301 Moved Permanently
Server: openresty
Date: Sat, 20 Mar 2021 18:18:03 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
X-Redirect-By: WordPress
Content-Security-Policy: upgrade-insecure-requests
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=300
Location: https://64mm.com/about/?tib&rwy=1%3D1
X-Cacheable: NO:HTTPS Redirect
```

5.2. https://64mm.com/hello-world/ [name of an arbitrarily supplied URL parameter]



Severity:	**Low**
Confidence:	**Firm**
Host:	**https://64mm.com**
Path:	**/hello-world/**

Issue detail

The name of an arbitrarily supplied URL parameter is copied into the response within the query string of a URL.

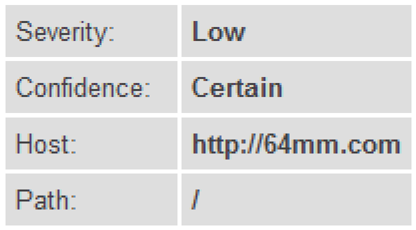
The payload lye&gfr=1 was submitted in the name of an arbitrarily supplied URL parameter. This input was echoed as lye&amp;gfr=1 within the "href" attribute of an "a" tag.

This proof-of-concept attack demonstrates that it is possible to inject arbitrary query string parameters into URLs in the application's response.

Request 1
```
GET /hello-world/?lye%26gfr%3d1=1 HTTP/1.1
Host: 64mm.com
Upgrade-Insecure-Requests: 1
Referer: https://64mm.com/blog/
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

Response 1
```
...[SNIP]...
<a rel="nofollow" id="cancel-comment-reply-link" href="/hello-
world/?lye&amp;gfr=1%3D1#respond" style="display:none;">
...[SNIP]...
```

6. Unencrypted communications



Severity:	**Low**
Confidence:	**Certain**
Host:	**http://64mm.com**
Path:	**/**

Issue description

The application allows users to connect to it over unencrypted connections. An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the application and obtain any information the user supplies. Furthermore, an attacker able to modify traffic could use the application as a platform for attacks against its users and third-party websites. Unencrypted connections have been exploited by ISPs and governments to track users, and to inject adverts and malicious JavaScript. Due to these concerns, web browser vendors are planning to visually flag unencrypted connections as hazardous.

To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Please note that using a mixture of encrypted and unencrypted communications is an ineffective defense against active attackers, because they can easily remove references to encrypted resources when these references are transmitted over an unencrypted connection.
Issue remediation

Applications should use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server. The Strict-Transport-Security HTTP header should be used to ensure that clients refuse to access the server over an insecure connection.

References
Marking HTTP as non-secure
Configuring Server-Side SSL/TLS
HTTP Strict Transport Security

Vulnerability classifications
CWE-326: Inadequate Encryption Strength

Appendix F FAQ

This section seeks to answer the most frequently asked questions about Cyber Hygiene reports.

1. May I add my third-party hosted/managed servers?

Yes, and we recommend this, but we request that you obtain authorization/consent before we begin scanning them.

2. I have added a new host and your scans are not picking it up.

Get Event Log is not scanning your entire IP scope every period. If you have installed a new server in a range that we only recently scanned and found nothing in, it is possible that the new server will not appear until next scan. If you want the new host to be scanned immediately, you can email Get Event Log and we will manually scan it and add it to your next report.

3. Why does the host count in my Cyber Hygiene report not match the number of known Internet-facing end points on my network?

This is likely due to a difference in what we are defining as a host. Get Event Log considers a device a host if there is at least one open port/service operating at the address. When we scan, any number of things can occur that make it appear that nothing is at that address (e.g., our scans are blocked by host or network filters, the device is down for maintenance, packets are dropped or lost in route, etc.).

If a port is detected as ’tcpwrapped’, it means that the TCP handshake was completed, but the connection was closed before any data was sent back. For the purposes of this report, tcpwrapped ports are not considered to be ’open’. If a device only responds with tcpwrapped ports, then it will not be considered a host by Get Event Log. For more information about tcpwrapped ports, see tcpwrapped.

The intent of Get Event Log is to find vulnerabilities, not count hosts, and our metrics should not be relied upon as a verified host count of your organization. The weekly host count should be taken as an estimate. If, however, there are no or extremely low host counts reported when there are known active hosts, it is possible that the Get Event Log scans are being blocked.

4. I am getting SSL/TLS certificate vulnerabilities that I believe are false.

In our scans, we will use the Mozilla trust store. Get Event Log will not accept any other roots. This is done as a matter of practice and principle: as practice, because maintaining private roots from our various stakeholders is operationally infeasible; as principle, because our scans aim to ensure that the user of your services is protected. The Mozilla trust store is generally representative of a ’lowest common denominator’ in what a public-serving site can reasonably expect of those users whose devices they do not manage.

Ensure that the root your certificate is issued from is included in the Mozilla root store. You should also verify that the intermediate certificates are presented with your site certificate. This allows the scanner to validate the certificate's chain of trust.

Good information on this can be found at information can be found here regarding Hypertext Transfer Protocol Secure (HTTPS), much of which is applicable for SSL/TLS.

5. Can Get Event Log scan my IPv6 addresses?
   yes

6. How can I change who receives my Cyber Hygiene report?

If you need to change the distro we mail to, email us here.

Acronyms

CVE	Common Vulnerabilities and Exposures; for more information refer here.
CVSS	Common Vulnerability Scoring System; for more information refer here.
HTTPS	Hypertext Transfer Protocol Secure; for more information refer here.
IP	Internet Protocol; for more information refer here.
IT	Information Technology; for more information refer here.
NVD	National Vulnerability Database; for more information refer here.
OS	Operating System; for more information refer here.
POA&M	Plan Of Action and Milestones.
RRS	Risk Rating System.
TCP	Transmission Control Protocol; for more information refer here.

Glossary

active vulnerability report.	A vulnerability that was detected in the most recent scan of a host used for this
false positive	Any normal or expected behavior that is identified in this report as a potentially exploitable vulnerability.
host	A device that has a least one open port/listening service.
initial detection	The initial point in time when Cyber Hygiene scans identified a vulnerability. This date is used to calculate the vulnerability's age.
IP address	A numerical label that identifies each device using the Internet Protocol to communicate over a network.
latest detection	The most recent time when Cyber Hygiene scans identified a particular vulnerability.
mitigation detection	The date when a previously identified vulnerability was no longer detected by Cyber Hygiene scans.
service	An application running at the network application layer that provides communications capabilities across an IP computer network.
severity	Please review the following guide for vulnerability severity scoring information: guide.
vulnerability	A weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source.
vulnerability age	The time between a vulnerability's initial detection date and its latest detection date.
vulnerable host	A host with at least one vulnerability detected on the most recent scan used for this report.