Security Assessment Report

Prepared For

[REDACTED]

[REDACTED]

1. Executive Summary

Between [REDACTED] and [REDACTED], OccamSec, LLC (OSec) performed an external network penetration test on behalf of [REDACTED]. The main goal of this assessment was to evaluate the security resilience of the external network perimeter, including web applications, to identify potential vulnerabilities that malicious actors could exploit. This report is intended to support informed decision-making, guide security enhancements, and strengthen [REDACTED]'s ability to protect sensitive assets, data, and operations.

OSec was successful in breaching the external network and gaining access to internal systems and private cloud infrastructure. By taking advantage of an uninitialized InfluxDB instance and abusing built-in features of the application, OSec was able to extract administrative account credentials and IAM security credentials from AWS metadata services. The obtained credentials could then be used to access the host through file sharing and remote desktop, and pull vast amounts of data from the AWS infrastructure. This critical issue requires immediate remediation, including removing access for these vulnerable services from the internet, credential rotation, and a review of current deployment practices to ensure hosts and services are not provisioned and left in a vulnerable state in the future.

In addition to the critical issue, multiple medium risk issues were also identified, including a username disclosure; unauthenticated source code disclosure for a running API service leading to a vulnerability discovered in the source code; and a lack of authentication required to access an API. Several vulnerable components and information disclosure issues were also identified representing a lower risk to the external network.

[REDACTED] should review the recommendations provided within this document and apply the associated mitigations. An examination of deployment policies and practices is recommended to ensure enforcement of security best practices. It is also recommended to implement strict role-based access controls (RBAC), enforcing the principle of least privilege, and segmenting provisioning accounts from data access roles. Adopting IAM best practices, such as credential rotation, monitoring for exposed secrets, and implementing service control policies (SCPs), will further enhance the security posture of the AWS environment. OSec also recommends regular testing to ensure that the network remains free of vulnerabilities, as well as performing authenticated testing to continue to reduce risk to services on the external network.

2. Scope

The assessment scope consisted of the following component:

- External Network Assessment (Outlined Below)

The following hosts and locations were included as part of the scope:

Scope Component	Targets / Hosts
[REDACTED]	[REDACTED]

3. Rules of Engagement

The OSec team adhered to the following rules of engagement:

1. No brute force attacks were undertaken. Given the capabilities of most modern operating systems to implement an account lockout feature, no attempt was made to brute force any passwords.
2. No Denial-of-Service (DoS) attacks were launched.

4. Findings Summary

The vulnerabilities identified during this assessment have been categorized by their estimated remediation urgency into the following categories: Now or Later. Table 1 below defines these categories.

Remediation Urgency	Definition	Impact
Now	Issue must be remediated immediately.	Exploitation and/or compromise of data in the short term is likely and imminent.
Later	Issue should be remediated at some point.	Exploitation and/or compromise of data in the medium to long term is possible.

Table 1 - Vulnerability Ratings

Based on the findings during the assessment, thirteen (13) vulnerabilities were identified. Tables 2 through 5 below summarize the results of testing and vulnerabilities according to type, remediation urgency, risk level, and count.

Testing Area	Vulnerabilities Present	Critical Data Accessed
External Network	Yes	Yes

Table 2 - Breach Summary

#	Vulnerability Name/Description	Remediation Urgency	Risk Level
1	Uninitialized InfluxDB instance leading to Account Takeover	Now	Critical
2	Gitlab Username Disclosure	Now	Medium
3	Source Code Disclosure	Now	Medium
4	Path Traversal in File Create	Now	Medium
5	Unauthenticated API Access	Now	Medium
6	Unauthenticated GraphQL Introspection Enabled	Now	Medium
7	Username Enumeration	Later	Medium
8	Insecure XMLRPC Functions Enabled	Later	Medium
9	API Keys Disclosed in Source Code	Later	Medium
10	CORS Arbitrary Origin Sharing	Later	Medium
11	Host Running Unmaintained Operating System	Later	Medium
12	Vulnerable and Outdated Components	Later	Low
13	Information Disclosure via Response Headers	Later	Low

Table 3 - List of Issues

Remediation Urgency	Total Number of Vulnerabilities
Now	6
Later	7
Total	**13**

Table 4 - Number of Vulnerabilities by Remediation Urgency

Risk Level	Total Number of Open Vulnerabilities
Critical	1
High	0
Medium	10
Low	2
Informational	0
Total	**13**

Table 5 - Number of Vulnerabilities by Risk Level

The tables below summarize the vulnerabilities discovered and exploited.

1 - Uninitialized InfluxDB instance leading to Account Takeover	
Location/s	[REDACTED]
Remediation Urgency	Now
Risk Level	Critical
CVSS 4.0 Score & Vector	9.9 (CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:H/SI:L/SA:N)
CVE/CWE	CWE-284: Improper Access Control

Issue Description

Several instances were discovered that have InfluxDB installed on the endpoint, however the final process of user creation was not completed on any of these installations. This allowed OSec to create a user on one of the InfluxDB instances. While this was a new installation, no sensitive information was obtained through the endpoint itself. However, the software utilizes a function within the 'Notebooks' feature, that allows an end user to write automation scripting using a language called 'Flux Script'. This allowed OSec to perform Server-Side Request Forgery (SSRF) attacks to make requests at the endpoint's behest to AWS meta-data endpoints giving access to the provisioning AIM account tokens, as well as scripts used to automate building the host machine containing hard coded credentials for both the host and the Active Directory domain.

While performing reconnaissance on the AWS meta-dat endpoints, OSec discovered that the [REDACTED] URL returned what appears to be a PowerShell setup script. Within this script are Administrator hard coded credentials that are available to anyone within the organization that has query access to the AWS URL mentioned above.

The script that was utilized to access this information is as follows:

[REDACTED]

Utilizing this Administrator password, OSec discovered which endpoints in scope had SMB and RDP ports open, and attempted to log in to them. OSec was able to connect using this password to eight instances which are listed below. This allowed OSec to extract sensitive information from the Active Directory environment, steal password hashes of domain users, and use the host as a pivot to perform reconnaissance on the internal network.

Steps to Reproduce

1. Via web browser, view one of the affected endpoints listed under 'Location(s)' above.
2. Set up a new user.
3. Click on Notebooks, create a new Notebook, within the Notebook click the plus sign below 'Run', and select 'Flux Script'.
4. Perform the query given within the description.
5. View the response from AWS.

Impact

Failing to set up an initial user in InfluxDB when deploying it in AWS creates a significant security vulnerability. By default, InfluxDB allows unauthenticated access if no user is configured, leaving the instance open to exploitation. By leveraging this access, OSec was able to retrieve AWS metadata from the EC2 instance's metadata service, including PowerShell provisioning scripts from [REDACTED], and AIM security keys from [REDACTED].

The sysprepRole security keys allowed access to hundreds of gigabytes of sensitive data stored in the AWS environment, including data on EC2, ELB, and S3 buckets. The scripts pulled from the [REDACTED] endpoint contained hardcoded credentials, which is another critical misstep in securing cloud infrastructure.

Using the credentials obtained, OSec gained administrative access to multiple Windows servers within the environment. This escalation of privileges allowed complete control of these systems, compromising sensitive data, disrupting services, and creating a persistent foothold in the environment. This scenario highlights the importance of following security best practices, such as immediately setting up an initial user with strong credentials in InfluxDB, securing access to the metadata service, and avoiding the use of hard coded credentials in cloud environments. Without these precautions, attackers can easily chain misconfigurations and design flaws to compromise an entire infrastructure.

Remediation

Enforce secure deployment practices for InfluxDB and other similar services. When deploying InfluxDB, always configure an initial administrative user with strong credentials during installation or immediately after deployment. Additionally, ensure that access to the InfluxDB instance is restricted to trusted IPs or private subnets using AWS Security Groups and Network ACLs.

To further mitigate risks, disable unauthenticated access and enforce role-based access controls (RBAC). Protect the AWS instance metadata service by implementing Instance Metadata Service Version 2 (IMDSv2), which requires session tokens for access, preventing unauthorized queries. Finally, eliminate hard coded credentials from user data scripts or other infrastructure, and instead use AWS Identity and Access Management (IAM) roles with least privilege to securely manage access to resources. Regularly audit configurations and implement automated tools to identify and remediate such misconfigurations before they are exploited.

Associated / Relevant Evidence
[REDACTED] Figure 1 - Uninitialized InfluxDB in Use [REDACTED] Figure 2 -InfluxDB Notebook Query Used to Connect to AWS [REDACTED] Figure 3 - Partial user-data Response with Redacted Password [REDACTED] Figure 4 - Connection to Windows SMB as Administrator [REDACTED] Figure 5 - Connection to Windows RDP as Administrator
References
https://docs.influxdata.com/flux/v0/stdlib/http/requests/get/ https://cwe.mitre.org/data/definitions/284.html

2 - Gitlab Username Disclosure	
Location/s	[REDACTED]
Remediation Urgency	Now
Risk Level	Medium
CVSS 4.0 Score & Vector	6.9 (CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CVE-2020-26413

Issue Description

OSec was able to extract a list of usernames from the GitLab instance by submitting a query to the GraphQL endpoint. The query (see Steps to Reproduce) is designed to retrieve a list of users from a Gitlab instance, including their username, email, avatar URL, and status information (such as an emoji, raw message, and HTML-formatted message). It uses the edges field to iterate through the user list, a common pagination pattern in GraphQL.

This query is useful for displaying user directories, auditing accounts, or showing user statuses in a dashboard. However, this query did not require an authenticated session to be honored. It should be noted that the indicated CVE-2020-26413 is described as an email disclosure, while its use here provided usernames instead.

Steps to Reproduce

1. Send a query to the [REDACTED] endpoint with the following payload:

```
{"query":"{ users { edges { node { username email avatarUrl status { emoji message messageHtml } } } } } }
```

2. View the returned usernames.

Impact
Exposing usernames on an external web endpoint presents a security risk, as it provides attackers with valuable reconnaissance data for brute-force attacks, credential stuffing, and social engineering tactics. Usernames often serve as a primary authentication factor, and when publicly accessible, they reduce the effort required for threat actors to launch targeted attacks, such as password spraying or phishing campaigns. Additionally, disclosing usernames can violate privacy regulations and organizational security policies, potentially leading to reputational damage, regulatory penalties, and increased attack surface.
Remediation
Update the GitLab instance to version 13.6.2 or later, as these versions contain the necessary patches to address this vulnerability.
Associated / Relevant Evidence
[REDACTED] Figure 6 - Username Disclosure on GitLab Instance
References
https://nvd.nist.gov/vuln/detail/cve-2020-26413

3 - Source Code Disclosure	
Location/s	[REDACTED]
Remediation Urgency	Now
Risk Level	Medium
CVSS 4.0 Score & Vector	6.9 (CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

Issue Description

The server is running a Node.js API on port [REDACTED], with a nginx hosted HTTP service on port [REDACTED]. Fuzzing for hidden content on the HTTP service identified the [REDACTED] file (Figure 7), which appears to be the source code to a Node.js application. Further investigation identified multiple .js files included from the server.js file, and package.json, package-lock.json, tsconfig.json, and protractor.conf.js configuration files. These files appear to be the source files running the API on port [REDACTED].

Analysis of the JavaScript source code allowed OSec to identify a path traversal vulnerability (Finding 4 - Path Traversal in File Create) in the [REDACTED] API action in the [REDACTED] getPathFilename function (Figure 8).

Steps to Reproduce

1. Browse to the [REDACTED] location and observe the source code provided.

Impact
OSec used the disclosed source code to discover details about how the API running on port [REDACTED] operated, including installed library versions in package.json, and how to interact with the API. This also allowed for the discovery of the path traversal vulnerability, and the means to exploit it. This further emphasizes the importance of keeping source code secure, as it can be a means to discover vulnerabilities in an application which can then be exploited.
Remediation
Restrict access to source files of the API application from being accessed by the nginx web server, this can be accomplished by running the Node.js API from outside of Nginx's web root folder. Nginx can then be configured to publish only the minimum required directories used to host static content for the application.
Associated / Relevant Evidence
[REDACTED] Figure 7 - Source Code of the Node.js API [REDACTED] Figure 8 - API Source Code with Filename Parsing Showing Path Traversal Vulnerability
References
https://cwe.mitre.org/data/definitions/200.html

4 - Path Traversal in File Create	
Location/s	[REDACTED]
Remediation Urgency	Now
Risk Level	Medium
CVSS 4.0 Score & Vector	6.9 (CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Issue Description

The server is running a Node.js API on port [REDACTED], with a nginx hosted HTTP service on port [REDACTED]. Fuzzing for hidden content on the HTTP service identified multiple .js files which appear to be the source code to a Node.js application. These files appear to be the source files running the API on port [REDACTED].

Analysis of the JavaScript source code allowed OSec to identify a path traversal vulnerability in the [REDACTED] API action. Posting a filename, and an array of strings in the URL input JSON parameters will create a text file on the server with the URL strings as the content of the file (Figures 9 and 10). Posting a malicious filename with a closing square bracket, followed by relative parent paths, and then an opening square bracket, will cause the API to create the file outside of the designated location (Figures 11 and 12).

OSec tested writing files to locations outside of the shared web folder, and successfully wrote to publicly writable locations on the server (e.g. [REDACTED]). Attempts to write to sensitive locations were unsuccessful as the application was not running with enough permissions to allow write access to these directories. While these attempts to exploit the vulnerability to gain

further access to the system were unsuccessful, an attacker could weaponize this to perform DoS attacks by filling up space on the filesystem with arbitrary files, exhausting the number of files that can be placed in specific directories, and causing financial impact to the organization as these files are transferred to S3 buckets by the application, incurring extra charges due to the extra storage required.

Steps to Reproduce

1. Post the following payload to the API service: [REDACTED]
2. The JSON response will contain a “filename” property, removing the path information from the property’s value, browse to the following URL, replacing the placeholder: [REDACTED]
3. The server should provide the text file with the “Url” content that was provided in the post request.

Impact

Attackers could exploit this flaw to create files in sensitive user-writable locations, such as application configuration directories, log directories, or web-accessible file paths. This could lead to a variety of exploits, including injecting malicious scripts into web application directories, altering application behavior by modifying application configurations, or filling up disk space to cause DoS conditions.

Remediation

Ensure all user input is validated and sanitized before use, and include ensure development continuous integration and continuous delivery (CI/CD) systems are checking for input validation early in the Software Development Lifecycle to prevent missing validation issues from reaching production systems.

Associated / Relevant Evidence
[REDACTED] Figure 9 - API Post to Create a File [REDACTED] Figure 10 - The Created File with the Contents Provided from the Post Request [REDACTED] Figure 11 - Post Request Exploiting Path Traversal Vulnerability [REDACTED] Figure 12 - Created File is Outside of the Directory
References
https://cwe.mitre.org/data/definitions/200.html https://cwe.mitre.org/data/definitions/22.html

5 - Unauthenticated API Access	
Location/s	[REDACTED]
Remediation Urgency	Now
Risk Level	Medium
CVSS 4.0 Score & Vector	6.9 (CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CWE-306: Missing Authentication for Critical Function

Issue Description

This application running on the specified locations is a single load balanced instance of the TigerGraph REST API service. The service provides a full list of API endpoints (Figure 13) and version information (Figure 14), as well as server and network data, as well as metrics on various tracked applications (Figures 15 and 16). Other instances of TigerGraph services were discovered in the environment, however they were configured to require authentication to access (Figure 17). Given the significant amount of data provided by the API, this instance should also require authentication to access its data.

Steps to Reproduce

1. Browse to one of the locations listed.
2. Observe the list of API endpoints that are allowed access.

Impact
The API contains a significant amount of IP and application information, allowing unauthenticated access to this information provides an attacker with additional targets, and details to develop attacks against network assets. The additional metric data could be used to target specific applications or services with Distributed Denial of Service (DDoS) attacks.
Remediation
Configure the TigerGraph REST API service to require authentication.
Associated / Relevant Evidence
[REDACTED] Figure 13 - List of Endpoints Provided by the API [REDACTED] Figure 14 - Endpoint Showing Version Information for the API and its Components [REDACTED] Figure 15 - Application Information Provided by the [REDACTED] API Endpoint [REDACTED] Figure 16 - Requesting IP Information for a Specific App Returns IP Addresses in Decimal Notation [REDACTED] Figure 17 - API Endpoints Requests from Another Instance Requiring Authentication

References
https://cwe.mitre.org/data/definitions/306.html

6 - Unauthenticated GraphQL Introspection Enabled	
Location/s	[REDACTED]
Remediation Urgency	Now
Risk Level	Medium
CVSS 4.0 Score & Vector	6.9 (CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CWE-306: Missing Authentication for Critical Function

Issue Description

The GraphQL API service running on [REDACTED] allows users to perform an introspection query with no prior authorization (Figure 18). The API is a large and complex one, and handles the data for the Gitlab server hosted on the same machine. The ability to perform introspection with no authorization on this API could therefore give an attacker a large amount of detailed information on the users, data, and projects in that version control system, potentially impacting many systems in the environment.

This is of particular importance due to the likely presence of code deployed throughout the environment that is maintained by this Gitlab server. An attacker who is able to dump the entire API schema would have much more detailed insight into both the users of the Gitlab

server and the projects being deployed.

Steps to Reproduce

1. Execute the following command:

   [REDACTED]

2. Observe the GraphQL Schema information returned.

Impact

The API contains a significant amount of information pertaining to a Gitlab repository that appears to be used to develop and deploy a number of projects and applications in the environment. Allowing unauthenticated access to this information could provide an attacker with additional targets, and details to develop attacks against network assets.

Remediation

Configure the GraphQL API service to require authentication prior to allowing users to enumerate the structure of the graph via introspection.

Associated / Relevant Evidence

[REDACTED]

Figure 18 - Unauthenticated Access to the GraphQL Introspection Query Permitted

References

https://cwe.mitre.org/data/definitions/306.html

7 - Username Enumeration	
Location/s	[REDACTED]
Remediation Urgency	Later
Risk Level	Medium
CVSS 4.0 Score & Vector	6.3 (CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CWE-204: Observable Response Discrepancy

Issue Description

OSec was able to enumerate a number of valid usernames from the web application. This was achieved via informational pages which were exposed to unauthorized users (Figure 19). OSec attempted to test the accounts for weak credentials by utilizing the [REDACTED] interface (detailed below). None of the accounts appeared to be using default or extremely weak passwords. However, by using the [REDACTED] method for password spraying, attackers are generally able to circumvent the normal account lockout mechanism enforced on the login page, which would allow an attacker to perform brute force attacks against the accounts given sufficient time.

Steps to Reproduce

1. Request the specified URL.
2. Observe the usernames in response.

Impact
The disclosure of usernames facilitates account takeovers. If usernames cannot be enumerated from a web application, it becomes significantly harder for an attacker to brute-force the password of an account and perform an account takeover. This is because an attacker will have to guess both the usernames and the passwords for the account, which requires far more resources and time to successfully exploit. In addition, on WordPress sites which have the [REDACTED] endpoint with insecure RPC calls enabled (Finding 8 - Insecure XMLRPC Functions Enabled**),** brute forcing is significantly easier than would be the case otherwise, as this endpoint could facilitate multiple password attempts in a single request. This underscores how username disclosure is an especially significant issue in WordPress sites.
Remediation
Restrict unauthenticated access to information pages disclosing usernames.
Associated / Relevant Evidence
[REDACTED] Figure 19 - WordPress Users Disclosed in [REDACTED]

References
https://cwe.mitre.org/data/definitions/204.html

8 - Insecure XMLRPC Functions Enabled	
Location/s	[REDACTED]
Remediation Urgency	Later
Risk Level	Medium
CVSS 4.0 Score & Vector	6.3 (CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CWE-285: Improper Authorization
Issue Description	
The [REDACTED] endpoint on the [REDACTED] site is accessible to unauthenticated users. The endpoint, which is used to perform a number of remote procedure calls (RPCs) on behalf of the WordPress site, has several insecure RPC calls enabled (Figure 20). These calls allow for several attacks such as internal scanning via SSRF and password brute forcing. OSec attempted a small-scale password spraying attack using this method using the top 250 passwords from the crackstation.txt list (see References) but the accounts in question did not contain these weak passwords. Likewise, OSec attempted to perform SSRF via the [REDACTED] RPC, however no open ports were discovered on the local host.	

Steps to Reproduce
1. Send a request (see Figure 20) to server. 2. Observe the response to see RPC methods enabled on server.
Impact
The insecure RPC methods, [REDACTED] and [REDACTED], impact the confidentiality of the system. The [REDACTED] procedure allows attackers to brute force passwords of known usernames (such as those exposed by the username enumeration described above). The [REDACTED] call allows for external attackers to perform simple ping scans of the internal network, enumerating the open ports and other information about the internal network. Together, these represent significant breaches of the confidentiality of the application.
Remediation
Restrict access to the [REDACTED] endpoint to properly authorized users. This can be done by setting the following code in the functions.php file (see Reference): [REDACTED]
Associated / Relevant Evidence
[REDACTED] Figure 20 - RPCs Enabled
References
https://cwe.mitre.org/data/definitions/285.html https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/wordpress-xml-rpc-

pingback-vulnerability-analysis/

9 - API Keys Disclosed in Source Code	
Location/s	[REDACTED]
Remediation Urgency	Later
Risk Level	Medium
CVSS 4.0 Score & Vector	6.3 (CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

Issue Description

The web application hosted a JavaScript source code file which contained a number of API keys to various paid services. These services include [REDACTED]. OSec attempted to exploit this disclosure of information, via enumerating sensitive information or accessing additional services, though these attempts did not lead to a successful compromise.

Steps to Reproduce

1. Request the JavaScript source code file.
2. Observe the response to see API keys embedded in JavaScript.

Impact
The main impact comes from the API keys allowing unauthorized users to run up the API costs for the organization, causing financial losses. [REDACTED] are all paid API services, meaning that with a valid API key, an attacker could likely generate a large number of queries and traffic to the API. This could run the bill for that API's usage up or potentially exhaust the limit on the traffic permitted for that API, impacting the availability of the service.
Remediation
Avoid hard-coding API credentials into source code. In areas where this is not possible, make sure to restrict viewing access to the source code to properly authenticated users.
Associated / Relevant Evidence
[REDACTED] Figure 21 - API Keys Disclosed in JavaScript Source Code File
References
https://cwe.mitre.org/data/definitions/200.html

10 - CORS Arbitrary Origin Sharing	
Location/s	[REDACTED]
Remediation Urgency	Later
Risk Level	Medium
CVSS 4.0 Score & Vector	6.0 (CVSS:4.0/AV:N/AC:H/AT:N/PR:N/UI:P/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CWE-942: Permissive Cross-domain Policy with Untrusted Domains

Issue Description

The application implements an HTML cross-origin resource sharing (CORS) policy for this request that allows access from any domain (Figures 22 and 23). OSec investigated several methods to exploit this misconfiguration, such as attempting malicious redirects, XSS, or other similar attacks. However, these attacks all required either user interaction (XSS for example) or a valid user account to demonstrate a proof of concept, restricting OSec from exploiting this vulnerability further.

Steps to Reproduce

1. Make a valid request to any site on the specified page with the addition of the “Origin” HTTP request header with a fake or malicious website name (ex: Origin: https://fakesite.com).
2. Observe the response headers to confirm CORS wildcard policy.

Impact
An HTML5 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 (SOP), 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. The web application is a [REDACTED] server with many dynamic pages and content or functionality that requires authentication and authorization to access, and therefore is likely an application that requires a more robust CORS policy to be secure. Additionally, 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.
Remediation
Review the business functionality of the site and determine an appropriate whitelist of permitted domains to allow under the CORS policy.
Associated / Relevant Evidence
[REDACTED] Figure 22 - CORS Policy Allows for Arbitrary Origins to be Trusted [REDACTED] Figure 23 - Source Code Reveals Wildcard in Use

References
https://cwe.mitre.org/data/definitions/942.html

11 - Host Running Unmaintained Operating System	
Location/s	[REDACTED]
Remediation Urgency	Later
Risk Level	Medium
CVSS 4.0 Score & Vector	5.9 (CVSS:4.0/AV:N/AC:H/AT:P/PR:H/UI:N/VC:H/VI:L/VA:L/SC:N/SI:N/SA:N)
CVE/CWE	CWE-1329: Reliance on Component That is Not Updateable
Issue Description	
The server is running Microsoft Windows Server 2019 (Figure 24). This operating system was declared End-of-Life in January 2024 (see References), meaning that it no longer receives regular security updates.	
Steps to Reproduce	
1. Navigate to [REDACTED]. 2. Observe the operating system version in the response.	

Impact
Although still on the “Extended End-of-Life” update schedule (see References) until 2029, the lack of regular security updates to the server operating system is a cause for concern. Windows Server 2019 has a large and diverse array of publicly disclosed vulnerabilities, ranging from remote code execution and buffer overflows to DoS and others. It is likely that more such vulnerabilities will be discovered in the coming years, and this underscores the need for consistent security patching.
Remediation
Review the business use cases for this application and what role it plays in the larger ecosystem. If possible, switch the underlying server operating system to a more modern version of Windows Server which is still receiving regular security patches.
Associated / Relevant Evidence
[REDACTED] Figure 24 - Windows Server 2019 Seen in Use
References
https://cwe.mitre.org/data/definitions/1329.html https://learn.microsoft.com/en-us/lifecycle/products/windows-server-2019 https://app.opencve.io/cve/?product=windows_server_2019&vendor=microsoft

12 - Vulnerable and Outdated Components	
Location/s	[REDACTED]
Remediation Urgency	Later
Risk Level	Low
CVSS 4.0 Score & Vector	2.3 (CVSS:4.0/AV:N/AC:H/AT:N/PR:L/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CWE-1352: Vulnerable and Outdated Components

Issue Description

Testing identified the presence of vulnerable and outdated libraries:

- jQuery (Version: 1.4.2, 1.12.4)
- jQuery Migrate (Version: 1.4.1)
- PHP (Version: 8.3.6 and 7.4.15)
- Apache (Version: 2.4.46)
- OpenSSL 1.0.2k-fips
- Perl (Version: 5.16.3)
- Microsoft-IIS (Version: 7.5 and 7.0)

Steps to Reproduce

1. Open the following URL in a browser:

[REDACTED]

Impact

Outdated libraries may contain known security vulnerabilities that have been patched in newer versions, making applications susceptible to exploitation by malicious actors. Regularly updating libraries is important for maintaining a secure, efficient, and up-to-date web application that can adapt to evolving web standards and security requirements.

Although these libraries did not lead directly to compromise, the presence of unsafe code significantly increases the likelihood of vulnerabilities being introduced in the future via source code changes and new features that include the unsafe functions and libraries.

Remediation

Update the affected libraries to the latest stable versions where possible. Furthermore, review the current patch management strategies to ensure that updates are promptly applied. Note that updating a library from a major release to a different major release may involve additional planning due to the changes that occurred between said releases, as compatibility issues may arise, and therefore additional planning and testing could be needed.

Associated / Relevant Evidence

[REDACTED]

Figure 25 - Vulnerable and Outdated Component jQuery 1.4.2

[REDACTED]

Figure 26 - Vulnerable and Outdated Component jQuery 1.12.4

[REDACTED]

Figure 27 - Vulnerable and Outdated jQuery Migrate 1.4.1

[REDACTED]

Figure 28 - Vulnerable and Outdated Component PHP 8.3.6

[REDACTED]

Figure 29 - Vulnerable and Outdated Component Apache 2.4.46, OpenSSL 1.0.2k-fips PHP 7.4.15, and Perl 5.16.3

[REDACTED]

Figure 30 - Vulnerable and Outdated Component Microsoft-IIS 7.5

[REDACTED]

Figure 31 - Vulnerable and Outdated Component Microsoft-IIS 7.5

[REDACTED]

Figure 32 - Vulnerable and Outdated Component Microsoft-IIS 7.5

[REDACTED]

Figure 33 - Vulnerable and Outdated Component Microsoft-IIS 7.5

[REDACTED]

Figure 34 - Vulnerable and Outdated Component Microsoft-IIS 7.0

[REDACTED]

Figure 35 - Vulnerable and Outdated Component Microsoft-IIS 7.0

[REDACTED]

Figure 36 - Vulnerable and Outdated Component Microsoft-IIS 7.0

[REDACTED]

Figure 37 - Vulnerable and Outdated Components Apache 2.4.33, OpenSSL 1.0.2k-fips, and PHP 7.0.30

References
https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/ https://security.snyk.io/package/npm/jquery https://www.tenable.com/plugins/nessus/193201 https://security.snyk.io/package/linux/alpine%3A3.13/apache2 https://security.snyk.io/package/linux/alpine%3A3.4/openssl https://security.snyk.io/package/linux/debian%3A11/php7.4 https://security.snyk.io/package/linux/amzn%3A2/perl https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=iis+7.5 https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=iis+7.0

13 - Information Disclosure via Response Headers	
Location/s	[REDACTED]
Remediation Urgency	Later
Risk Level	Low
CVSS 4.0 Score & Vector	2.1 (CVSS:4.0/AV:N/AC:L/AT:P/PR:H/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N)
CVE/CWE	CWE-200: Exposure of Sensitive Information

Issue Description
The application discloses sensitive information in its response headers, such as the 'Server: Microsoft-ISS/10.0' and the 'X-Powered-By: ASP.NET', headers. These headers reveal the underlying technologies used by the applications, providing attackers with valuable information about the software stack. Exploiting this information, attackers could identify known vulnerabilities or weaknesses associated with the disclosed technology, increasing the risk of targeted attacks against the applications.
Steps to Reproduce
1. Run the following commands: [REDACTED]
Impact
Disclosing technology details in response headers increases the risk of targeted attacks, as attackers can exploit known vulnerabilities in the identified platforms. This can result in unauthorized access, data breaches, or service disruptions. Additionally, such exposure violates security best practices, potentially leading to reputational harm and compliance issues.
Remediation
Configure the web server to remove or obscure sensitive headers such as `Server: Microsoft-ISS/10.0` and 'X-Powered-By: ASP.NET'. This can be done by modifying server settings or using web application firewalls to strip unnecessary response headers. Ensure that only the necessary headers for functionality and security purposes are exposed.
Associated / Relevant Evidence
[REDACTED]

Figure 38 - `Server: Microsoft-IIS/10.0' and 'X-Powered-By: ASP.NET' Shown In Response

[REDACTED]

Figure 39 - 'Server: Microsoft-IIS/10.0' and 'X-Powered-By: ASP.NET' Shown In Response

[REDACTED]

Figure 40 - 'Server: Microsoft-IIS/7.5' and ‘X-Powered-By: ASP.NET’ Shown in Response

[REDACTED]

Figure 41 - 'Server: Apache' Shown In Response

References

https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

https://cwe.mitre.org/data/definitions/200.html

5. Walkthrough

Initial testing started with reconnaissance and network scanning to build a profile of the network assets, identify live hosts, and determine accessible services. S3 buckets as defined within the project scope were enumerated, network port scans, and vulnerability scans were performed to identify any low hanging fruit and vulnerable services.

After initial scans completed, OSec moved to manual service enumeration to gather more details about running applications, operating systems, and potential misconfigurations. Scan data was analyzed to identify and link open ports with services, inspect service banners, and probe for known vulnerabilities in software versions. Network assets were checked for common vulnerabilities, weak authentication mechanisms, and misconfigured systems. This identified multiple outdated components, including JavaScript libraries, web servers, and disclosed scripting languages. This also uncovered some discrepancies in API access where some systems were accessible without authentication when most required JWT or other token authentication.

Web application testing was performed on multiple targets, including fuzzing for hidden endpoints, forced browsing to access endpoints that would otherwise require authentication, and parameter tampering to identify input sanitization issues which could lead to injection of malicious payloads. Using these techniques, OSec identified a source code disclosure issue that revealed the inner workings of an API running on the same host, as well as a path traversal vulnerability in that API.

OSec also identified several InfluxDB instances that had not been configured, allowing anyone on the internet to register a user with the application. Further research and testing allowed for the discovery of a SSRF technique using the Flux Script language. OSec was able to use this attack vector to gain access to sensitive data stored in the AWS Instance Metadata Service, allowing full compromise of the host and significant access to internal network data, Active Directory, and vast amounts of EC2 and ELB data as well as stored S3 Buckets and SSM secrets.

Upon the discovery of the credentials in the InfluxDB SSRF attack, OSec reviewed the scanning information for the host to see ports [REDACTED] were open. OSec verified the credentials by connecting to the host over the SMB file share. After the credentials were confirmed, the other open SMB ports were accessed to confirm if the credentials were shared between hosts. This showed the InfluxDB instances identified were accessible using the same set of credentials.

[REDACTED]

Figure 42 - Accessing SMB File Share Listings Using the Administrator Credentials with Successful Connections

[REDACTED]

Figure 43 - Remote Desktop Access Using the Administrator Credentials

After gaining initial access to the system, OSec reviewed installed services and user data for sensitive information, however as this was an automated deployment and had yet to be fully configured, no sensitive data was discovered.

OSec attempted to pull cached account hashes from the Windows registry SAM and SECURITY hives using multiple methods. The host was installed with endpoint detection and response agent, which detected and blocked local and remote registry access to those hives. Using esentutl.exe to perform a Volume Shadow Copy of the files on disk, scp was then used to copy them off of the host.

[REDACTED]

Figure 44 - Using Volume Shadow Copy Service to Pull Registry Hive Information from the Host

[REDACTED]

Figure 45 - Extracting Hashes from the Registry Hive Information Pulled from the Host

The hashes extracted from the SAM hive were blank, which would be expected for an unconfigured host that had only completed initial provisioning. The SECURITY hive contained

three (3) domain cached credentials from users accessing this host. Attempts to crack the hashes using common password lists, lists of publicly disclosed passwords, and brute forcing attempts were unsuccessful.

Initial network reconnaissance identified a locally accessible domain controller, however, other internal network resources seemed to be segmented off to prevent access. With the endpoint detection and response agent installed, using the host to attack the domain controller would be difficult as tooling would likely be detected and quarantined. It was determined that using the host to pivot attacks through would allow better chance of success with less risk of detection.

OSec setup a PowerShell script to test for open ports that could be accessed through the AWS Security Group port restrictions that may be in place. After completion of the script, OSec tested multiple random ports for access, and each port tested allowed external connections. This may be an indication that security groups have not been assigned properly to these hosts.

[REDACTED]

Figure 46 - Testing External Access Using Random Port Numbers with no Blocked Connections

With the connection testing completed, OSec created a Go based Socks5 proxy that restricts access to OSec IP Addresses.

[REDACTED]

Figure 47 - s5proxy.go source code

The data obtained from the InfluxDB SSRF attack also included a second set of credentials as part of the PowerShell provisioning script. These credentials appear to be used to automate joining the host to the Active Directory domain.

[REDACTED]

Figure 48 - Second Set of Credentials Used for Joining the Host to the Domain

Using these credentials externally did not allow any access, however attempting to log into the identified domain controller with the credentials gave an error that the user is valid but lacked authorization for the remote desktop service.

[REDACTED]

Figure 49 - Attempting to Log into the Internal Domain Controller Using the Second Set of Credentials

OSec compiled and copied a compatible version of Godap (https://github.com/Macmod/godap) to do some Active Directory enumeration using the LDAP service of the domain controller. This allowed extracting information on user accounts, domain administrators, domain controllers, and domain joined computers. Using the Godap application, OSec was able to query domain joined computers based on the reported operating system version, this list was used to get IP addresses and attempt to access the systems via SMB and RDP. Network segmentation prevented access to these systems, or the systems were not active at the time, preventing confirmation of their presence on the network.

[REDACTED]

Figure 50 - Godap Showing Domain Admins Group with List of Members

[REDACTED]

Figure 51 - Godap Showing Listing of Computers with Outdated Operating Systems

Using the user listing pulled from Active Directory, a list of users with defined attributes set was created. Using this list, OSec was able to request Kerberos tickets for the accounts in order to use the ticket hash to crack the account's password. Attempts to crack the hashes using common password lists, lists of publicly disclosed passwords, and brute forcing attempts were unsuccessful.

[REDACTED]

Figure 52 - Kerberos Tickets Created for the Users

InfluxDB Exploitation - AWS

Once the ability to query AWS meta-data was confirmed, OSec discovered AWS credentials by making a request to the [REDACTED] endpoint. Utilizing automated and manual methods, the credentials were checked against AWS resources to determine what access it was allotted. The amount of data that the exposed set of credentials afforded was vast, including: EC2 Instances, EC2 AMIs, EC2 Volumes, EC2 Snapshots, ELB Load Balancers, ELBv2 Load Balancers, Global S3 Buckets, and SSM Parameters.

Various secrets were able to be retrieved from SSM including passwords such as the following:

[REDACTED]

Figure 53 - Discovered Database Password

[REDACTED]

Figure 54 - Discovered Admin Password

Due to time constraints, not all of the AWS data was able to be analyzed, however within the S3 buckets OSec discovered the following: Elastic and other miscellaneous Logs; Database dumps; Configuration files; SSH keys; Jenkins backups; *and* User lists.

[REDACTED]

Figure 55 - Elastic Load Balancing Logs

[REDACTED]

Figure 56 - Dump and Log Files

[REDACTED]

Figure 57 - User and Utilization CSV Files

6. Conclusion

OSec conducted a penetration test on the external network infrastructure to assess its security resilience and identify vulnerabilities that could be exploited by malicious actors. The assessment revealed significant security gaps, including a critical issue that allowed OSec to breach the external network, gain access to internal systems, and extract administrative credentials from AWS metadata services. This exposure underscores the need for immediate remediation, including restricting internet access to vulnerable services, rotating compromised credentials, and reviewing deployment practices to prevent similar misconfigurations.

Beyond the critical finding, multiple medium and low-risk issues were identified, such as username disclosure, unauthenticated access to API source code, and various information disclosure vulnerabilities. While these issues pose a lower immediate threat, they still require attention to minimize the risk of exploitation.

To strengthen the organization's security posture, it is imperative to implement the recommended mitigations outlined in this report. Key actions include enforcing security best practices for system provisioning, implementing strict RBAC to uphold the principle of least privilege, and segmenting provisioning accounts from data access roles. Further, adopting IAM best practices, such as credential rotation, monitoring for exposed secrets, and enforcing SCPs will enhance AWS security. Regular penetration testing, including authenticated assessments, is also advised to proactively identify and address vulnerabilities before they can be exploited. By addressing these findings and reinforcing security measures, [REDACTED] can significantly reduce its attack surface and improve its overall cybersecurity resilience.