Sri Lanka Institute of Information Technology

Bug Bounty Report 08

brandportal.aswatson.com

IE2062 – Web Security

Submitted by:

IT22227836 – ADHIKARI A.M.V.B.M

Date of submission

2024.20.31

Table of Contents

Report 08 – brandportal.aswatson.com

Main domain	https://www.aswatson.com/
Sub domain	brandportal.aswatson.com
IP address	210.0.245.141
platform	HackerOne

The A.S. Watson Group is the world's largest health and beauty retail group, with over 15,700 stores in 25 markets worldwide serving over 28 million customers per week, and over 3 billion customers and members.

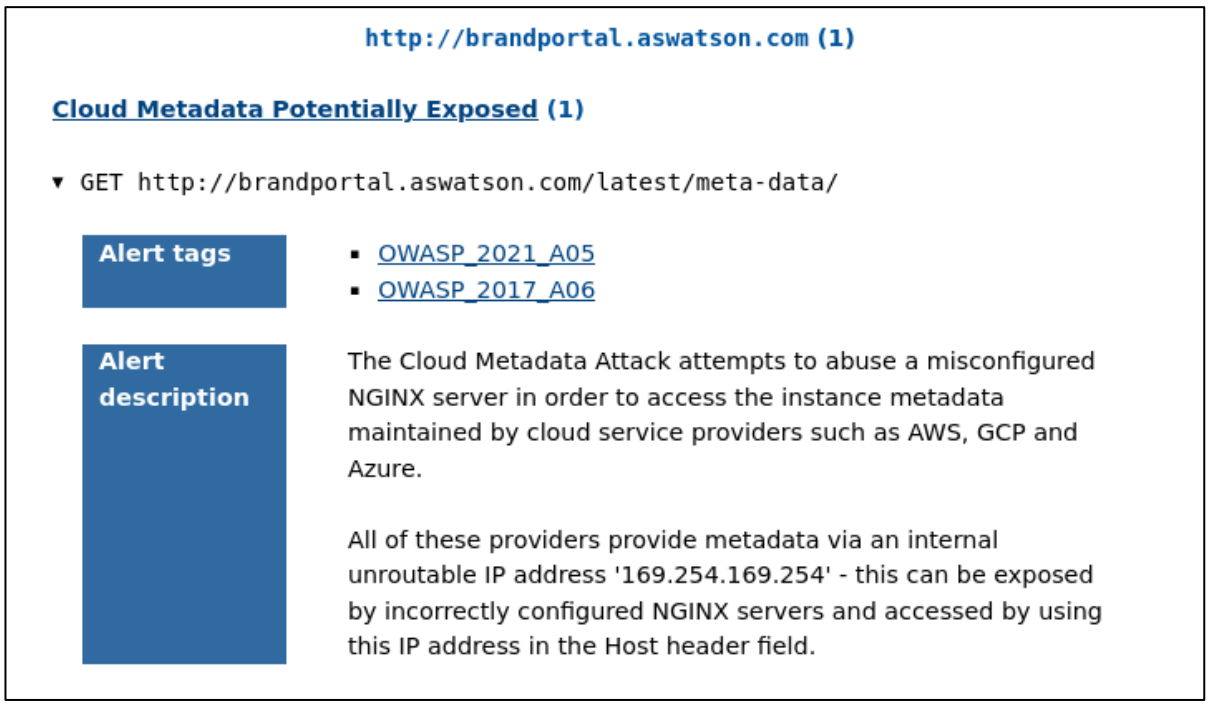
A.S. Watson Group looks forward to working with the security community to discover vulnerabilities to keep our businesses and customers safe. As we operate in many different countries, we will be rolling out our bug bounty program in phases. Our focus within this rollout, is our retail websites (and soon, mobile apps on both Android and IOS)

Vulnerability detected

Alert type	Risk	Count
Cloud Metadata Potentially Exposed	High	1 (16.7%)
Content Security Policy (CSP) Header Not Set	Medium	1 (16.7%)
Hidden File Found	Medium	4 (66.7%)
X-Content-Type-Options Header Missing	Low	1 (16.7%)
Modern Web Application	Informational	1 (16.7%)
User Agent Fuzzer	Informational	36 (600.0%)
Total		6

Vulnerability

1. Title – Cloud Metadata Potentially Exposed



http://brandportal.aswatson.com (1)

Cloud Metadata Potentially Exposed (1)

▾ GET http://brandportal.aswatson.com/latest/meta-data/

Alert tags

- OWASP_2021_A05
- OWASP_2017_A06

Alert description

The Cloud Metadata Attack attempts to abuse a misconfigured NGINX server in order to access the instance metadata maintained by cloud service providers such as AWS, GCP and Azure.

All of these providers provide metadata via an internal unroutable IP address '169.254.169.254' - this can be exposed by incorrectly configured NGINX servers and accessed by using this IP address in the Host header field.

Description

This section describes the vulnerability itself, including how it works.

In this case, NGINX is a very popular web server and reverse proxy, yet one that is misconfigured in accepting any value in the Host header of the HTTP request. In doing so, when the attacker sets the Host header to the internal IP address 169.254.169.254, the server forwards the request to the cloud metadata service, which is supposed to be accessed by the cloud instance itself. The metadata service would respond with sensitive information related to the cloud instance: IAM roles, credentials, or network configuration

Why is it dangerous?

- Cloud providers such as AWS, Google Cloud, and Azure expose metadata services through the internal IP address 169.254.169.254. These metadata services provide critical information about the cloud instance, which might include security credentials.

- Sensitive Exposure: Attackers can retrieve instance credentials, private keys, or other secrets that can potentially be used to compromise cloud resources.

Affected components

- **NGINX Server Configuration:** It is vulnerable to Host header manipulation, which can make it forward requests to internal services.
- **Meta-data Service of Cloud**: The meta-data service behind this cloud is exposed because of its poor or inappropriate request forwarding mechanism.
- **IAM Role and Credentials**: The IAM role and access credentials assigned to the instance can be leaked, thus enabling privilege escalation.
- **Problematic Configuration of Firewall/Network:** It could be worse without the proper set of firewall rules that block access to the internal cloud services.

All these components are playing an important role in the exploitation of vulnerability; hence, any fix needs to cover these areas so that the system is secured properly.

Impact Assessment

Cloud Metadata Exposure via Misconfigured NGINX Host Header: This vulnerability is critical.

- **Sensitive Information Disclosure**: it allows attackers to retrieve cloud metadata for exposing sensitive information, including instance details, IAM role information, and access credentials. This would allow attackers to escalate privileges based on the obtained credentials and extend access to more critical resources or administrative actions on the cloud platform.

- **Compromise cloud resources**: With this metadata, the attacker can gain unauthorized access to services or resources within the cloud infrastructure and cause further exploitation of the system.

- **Data breach or manipulation of data**: If it involves sensitive data from the cloud, this data breach could lead to financial loss or possibly critical customer or business data exfiltration.

- **Service Disruption:** Depending on the severity of the cloud infrastructure compromise, services may be terminated, operational hours may be reduced, or other implications on operational activities.

In a nutshell, one of the potential impacts of this vulnerability is a total cloud environment compromise, with serious financial, operational, and reputational damage.

Steps to reproduce

- **Identify a Target Server Using NGINX:**
  Identify a server that has been configured to use NGINX either as a reverse proxy or as a web server. Check if the server is forwarding requests based on the Host header.

- **Craft an HTTP Request with a Malicious Host Header:**
  Using any HTTP client (e.g., curl, Postman, or Burp Suite), craft a GET request to the target server.
  Replace the Host header with the cloud metadata IP 169.254.169.254

- **Send the Request:**
  Send the crafted request to the vulnerable server. The request will be proxied by NGINX to the internal cloud metadata service.

- **Analyze the Response:**
  If the server is vulnerable, it will return a response containing cloud metadata including instance details, IAM roles, and sensitive credentials.

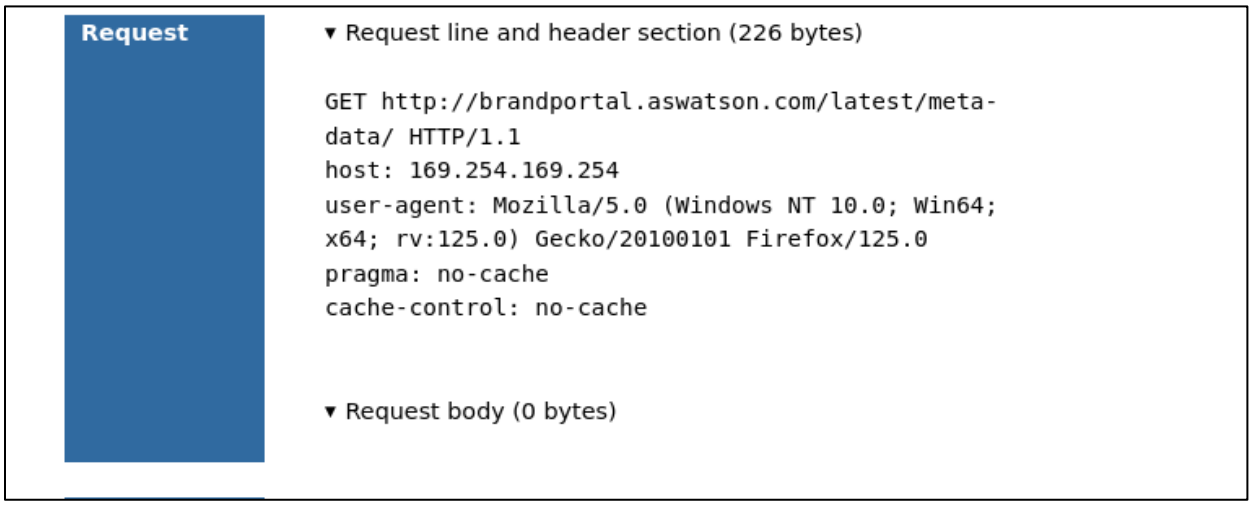
- **Extract Sensitive Cloud Metadata:**
  The response may contain access tokens or IAM role information that would enable the attacker to assume privileges and access cloud resources.

This vulnerability, when used correctly by an attacker, allows access to metadata from the cloud that may lead further to compromise the cloud infrastructure.

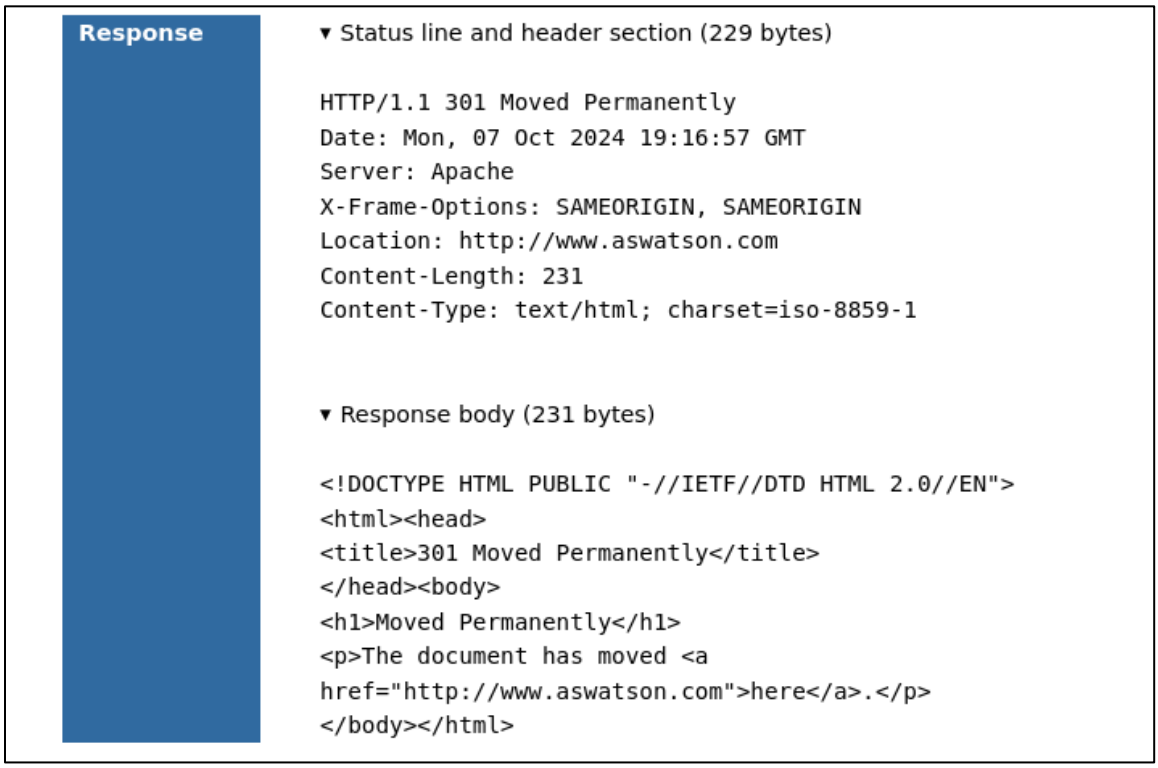
Proof of concept (if applicable)

Vulnerability scanning using OWASP ZAP

- **Request**



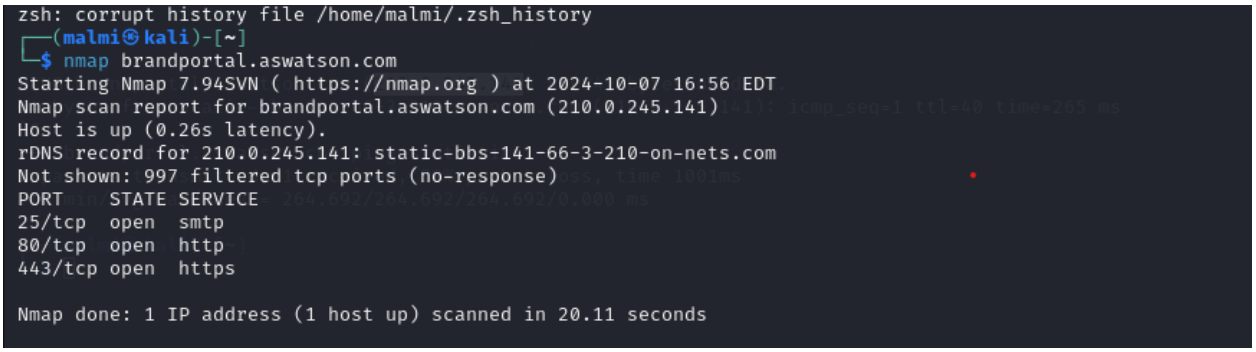
- **Response**



Vulnerability finding using Namp

Used command

- nmap brandportal.aswatson.com



Founded open port

- 25/tcp open smtp
- 80/tcp open http
- 443/tcp open https

Proposed mitigation or fix

Solution

Do not trust any user data in NGINX configs. In this case it is probably the use of the $host variable which is set from the 'Host' header and can be controlled by an attacker.

Metadata Service Access Restriction:

- **Use the firewall rules:** to restrict access to metadata service. For instance, 169.254.169.254 should be internal only.
  Whitelist IPs from trusted sources only.

- **Secure Server Configuration:**
  NGINX: One way of disabling host header manipulation is by using a default server block that would return a 404 for invalid hosts.

- **IAM Policies:**
  Use least privilege IAM roles and policies, which restrict the calling of metadata only to the required service. Explicitly deny permission to any services that should not need to call metadata.

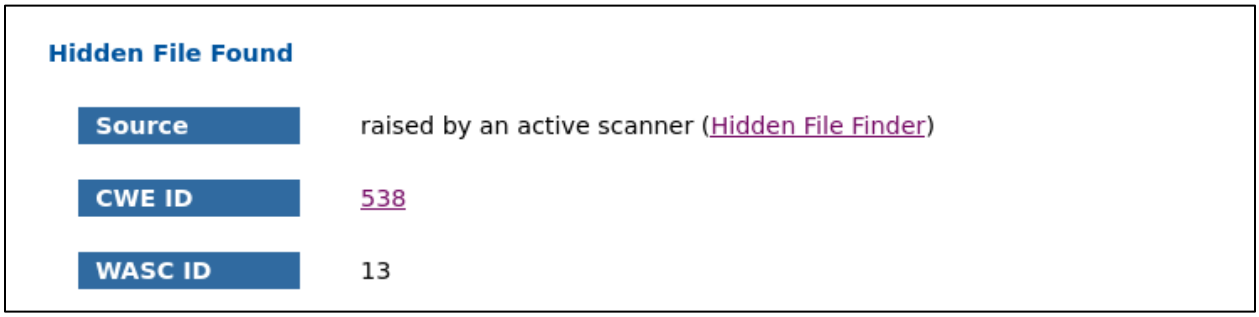
- **Regular Audits:**
  Periodic security auditing and penetration testing to find any misconfigurations.

- **Monitoring:**
  Enable AWS CloudTrail logging and set up notifications in case someone tries to access without authorization.

- **Training:**
  Provide security awareness training for DevOps teams in order to impress the importance of how metadata access should be secured.

This would in turn create a way through which the cloud metadata could be kept safe against unauthorized access.

2. Tittle – Hidden File Found



Description

It disclosed the presence of a hidden. hg directory at AutoDiscover.aswatson.com, which was related to the Mercurial Version Control System. Further, this may contain sensitive information such as source code, configurations, credentials, and so on, which can be utilized by an attacker. If it gets exposed, that may allow an attacker to get deeper insight into the system and increase the chance of further exploitation.

However, this may be mitigated either by cleanup of the unwanted version control directories from a production environment or by locking down access via appropriate authentication. This way, sensitive information will not be disclosed to unauthorized users.

Affected components

- **Sensitive Information Disclosure:** This could lead to attackers having access to sensitive project information, like code history, internal changes, and even credentials.
- **Publicly Disclosable Source Code:** Publicly available hg. directories can allow for complete access to source code disclosure; attackers can then find and exploit security vulnerabilities.
- **Poor Authorization Controls:** The so-called unauthorized access to version control files is because of poor authorization controls in place, where sensitive resources are made available to anyone knowing a particular URL.
- **Poor Security:** The existence of a. hg folder indicates poor concern for security best practices; such files must be banned from exposure to the outside world

Impact Assessment

- **Exposed Sensitive Information:** Publicly accessible version control files like the .hg directory have publicly exposed some details of the project-such as source code, configurations, internal documentation-exposing them to attackers, which can be very useful to find vulnerabilities in the system.

- **Increased Risk of Attacks**: More attack vectors could allow an attacker to realize code analysis using version control system access, which might expose security vulnerabilities, configuration flaws, or logical errors that can be used against the system. The risk related to code injections, privilege escalation, and remote code execution would increase manifold.

- **Compliance Exposure:** It might also result in the exposure of sensitive configuration files, which directly violates different security standards related to industries, including PCI DSS, OWASP, and HIPAA. This may involve some legal or financial consequences too.

- **Social Engineering:** The leaked administrative or credential information can be used in social engineering attacks. An attacker can manipulate employees or users into offering further unauthorized access to attackers.

Steps to reproduce

Identify the Target URL:
Use a web browser or command line utility to identify the target URL.

Send an HTTP GET Request:
Using a command line utility, like curl or wet, make a GET request for the backdoor

Analyze the Response:
Observe the response of the server. A response indicating a redirect, HTTP 301, or an error status, for instance 404, does not necessarily indicate a vulnerability. If you receive a 200 OK or any other successful response, know that the (hidden) file is accessible.

Existence Check of File:
If it comes in response that the file exists or is redirected to other locations, then this may be because of the possible vulnerability in exposing sensitive files.

Document findings:
by screenshotting the request and response, to include in your vulnerability report, as this will be your proof that the sensitive file was disclosed. These enable you to double-check the existence of potentially sensitive files, which really should not be available to unauthorized users.

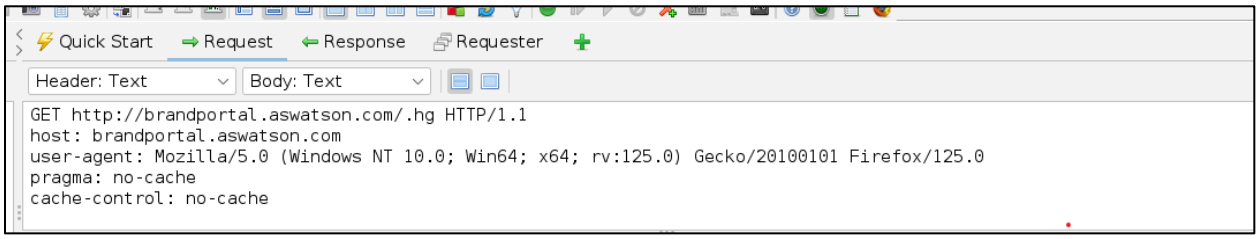
Proof of concept (if applicable)

Vulnerability scanning using OWASP ZAP

Hidden File Found

Source	raised by an active scanner (Hidden File Finder)
CWE ID	538
WASC ID	13
Reference	▪ https://blog.hboeck.de/archives/892-Introducing-Snallygaster-a-Tool-to-Scan-for-Secrets-on-Web-Servers.html

- **Request**



- **Response**

Vulnerability finding using namp

Used command

- nmap brandportal.aswatson.com

```
zsh: corrupt history file /home/malmi/.zsh_history
┌──(malmi㉿kali)-[~]
└─$ nmap brandportal.aswatson.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-07 16:56 EDT
Nmap scan report for brandportal.aswatson.com (210.0.245.141)
Host is up (0.26s latency).
rDNS record for 210.0.245.141: static-bbs-141-66-3-210-on-nets.com
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE SERVICE
25/tcp  open  smtp
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 20.11 seconds
```

Founded open port

- 25/tcp open smtp
- 80/tcp open http
- 443/tcp open https

Vulnerability finding using Nikto

- **Used command**
  - nikto -h brandportal.aswatson.com

```
┌──(malmi㉿kali)-[~]
└─$ nikto -h brandportal.aswatson.com
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          210.0.245.141
+ Target Hostname:    brandportal.aswatson.com
+ Target Port:        80
+ Start Time:         2024-10-08 01:24:54 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type.
 See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Root page / redirects to: https://brandportal.aswatson.com/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ ERROR: Error limit (20) reached for host, giving up. Last error: opening stream: getaddrinfo problems (Temporary failure in name resolution): Resource temp
orarily unavailable
+ Scan terminated: 20 error(s) and 1 item(s) reported on remote host
+ End Time:           2024-10-08 01:33:29 (GMT-4) (515 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

- **Results**
  - The scan output indicates that the server is running Apache and does not have the X-Content-Type-Options header set. This suggests that the server might be misconfigured and could allow unintended behavior regarding file serving

Proposed mitigation or fix

Solution

Consider whether or not the component is actually required in production, if it isn't then disable it. If it is then ensure access to it requires appropriate authentication and authorization, or limit exposure to internal systems or specific source IPs, etc.

- **Limit Access to Critical Files:**
  File Permissions: Ensure file permissions are set as restrictive as possible for sensitive files (. env, configuration files, backups); only certain user and service access should be granted.

- **Protection of Directories:**
  Protect sensitive directories against direct access by configuring server settings. For example, restrict access to the. git or backup directories.

- **Server Settings Configuration:**
  Go to server settings and turn off directory listing. Additionally, in server configurations, ensure that Options -Indexes for Apache are turned off.

- **Use the .htaccess file:**
  Create an .htaccess file in which certain sensitive files are denied access.

- **Security Headers:**
  Add security headers such as X-Content-Type-Options, which prevents the sniffing of MIME types.

- **Regular Audits:**
  Regular audits on security, with regard to deletion of unwanted files and securing sensitive files.

- **Monitoring and Logging:**
  In addition, monitoring and logging should be implemented to identify unauthorized attempts to access sensitive files promptly. Utilize tools like fail2ban that can monitor server logs for suspicious events and auto-email such events to an administrator.

- **Web Application Firewalls:**
  Use a WAF to help filter and monitor HTTP requests against attempts at accessing hidden files.
- **Education and Awareness:**

Training in best practices among development and operations staff will ensure proper handling of sensitive information and configuration files.