AutomatIA
AI Security Analysis
Final Report

Reported by:

Alph Infinity

Junio 2024

ALPHAINFINITY CONFIDENTIAL INFORMATION

V.1.1.
FINAL

TLP:RED

Not for disclosure, restricted to participants only.

Contacts

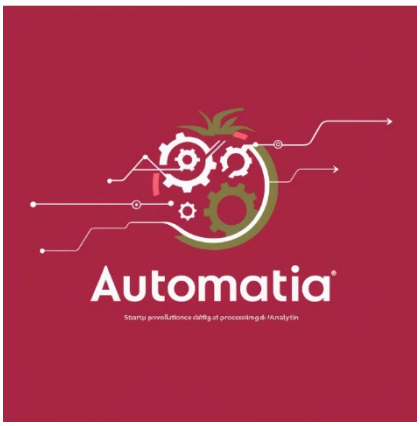
AutomatIA

Name	Title	E-mail
Adriana Lyzeth Vela Peña	Product Director	adriana.pena@alphainfinity.com
Lucas Moyano	UX/UI	lucas.moyano@alphainfinity.com
Wilso Pérez	IoT Manager	w.perez@alphainfinity.com

AlphaInfinity

Name	Title	E-Mail
Héctor Mora	AI Security Consultan	hmora10@alphainfinity.com
Oscar Bertel	ML Engineer	oscar.bertel@alphainfinity.com

AutomatIA



1 Executive Summary

1.1 Objective and Findings

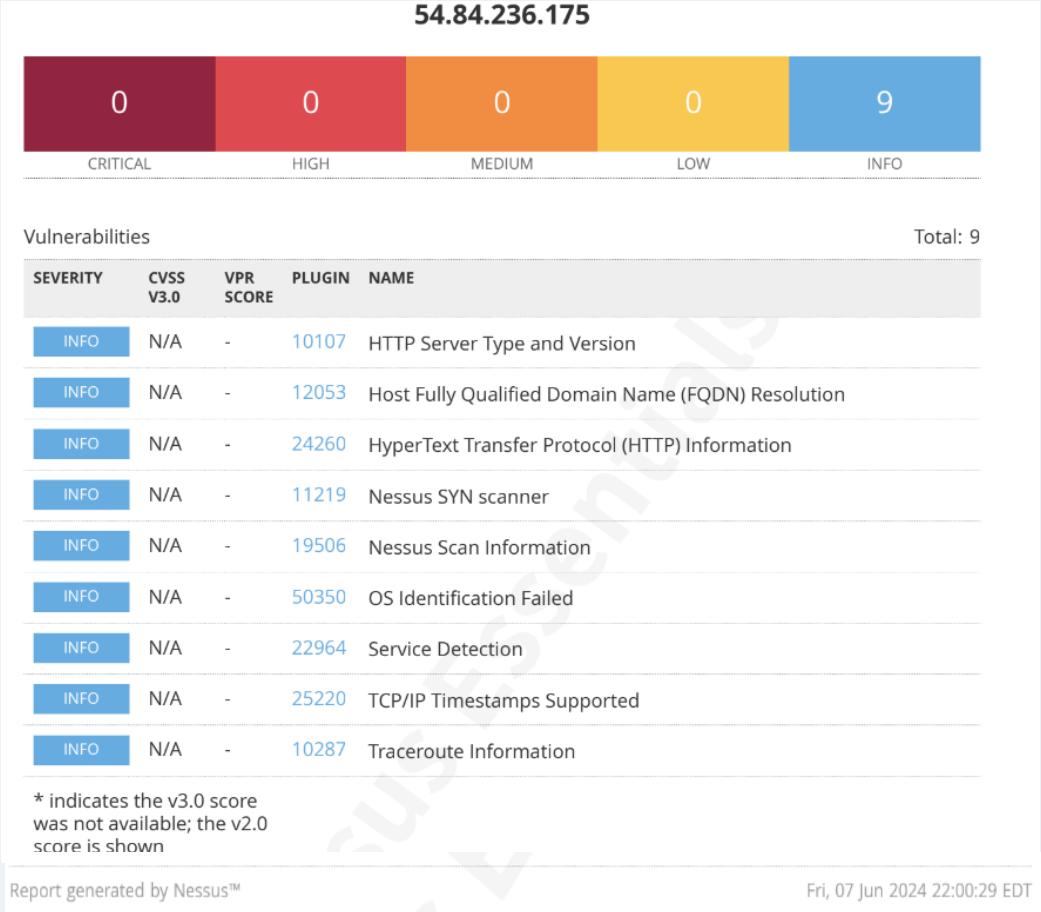
AutomatIA (further referred to as Client) has contracted **AlphaInfinty Security Team** to conduct a specialized AI Security Analysis including penetration tests and a Static Application Security Testing (SAST) service for applications and Source Code. These tests have the purpose of measuring the possible risks, vulnerabilities, and weaknesses in the communications infrastructure when it comes to external cyber-attacks.

The tests were performed during June - 2024 using standard methods of penetration testing following the PCI, NIST, and OWAST guidelines, which include active and passive scans of the systems.

The objective of this report is to present the findings obtained during the penetration tests and provide recommendations as to avoid any incidents that could compromise the confidentiality, integrity, and availability of the services provided by the **client**.

1.2 Findings External Pentest Summary

The following graph summarizes the severity level found



54.84.236.175

0	0	0	0	9
CRITICAL	HIGH	MEDIUM	LOW	INFO

Vulnerabilities Total: 9

SEVERITY	CVSS V3.0	VPR SCORE	PLUGIN	NAME
INFO	N/A	-	10107	HTTP Server Type and Version
INFO	N/A	-	12053	Host Fully Qualified Domain Name (FQDN) Resolution
INFO	N/A	-	24260	HyperText Transfer Protocol (HTTP) Information
INFO	N/A	-	11219	Nessus SYN scanner
INFO	N/A	-	19506	Nessus Scan Information
INFO	N/A	-	50350	OS Identification Failed
INFO	N/A	-	22964	Service Detection
INFO	N/A	-	25220	TCP/IP Timestamps Supported
INFO	N/A	-	10287	Traceroute Information

* indicates the v3.0 score was not available; the v2.0 score is shown

Report generated by Nessus™ Fri, 07 Jun 2024 22:00:29 EDT

No critical vulnerabilities were detected,

OVERALL SEVERITY LEVEL: INFORMATIVE

STATUS: CLOSED

10180 - Ping the remote host

Synopsis

It was possible to identify the status of the remote host (alive or dead).

Description

Nessus was able to determine if the remote host is alive using one or more of the following ping types :

- An ARP ping, provided the host is on the local subnet and Nessus is running over Ethernet.

- An ICMP ping.

- A TCP ping, in which the plugin sends to the remote host a packet with the flag SYN, and the host will reply with a RST or a SYN/ACK.

- A UDP ping (e.g., DNS, RPC, and NTP).

Solution

n/a

Risk Factor

None

Plugin Information

Published: 1999/06/24, Modified: 2024/03/25

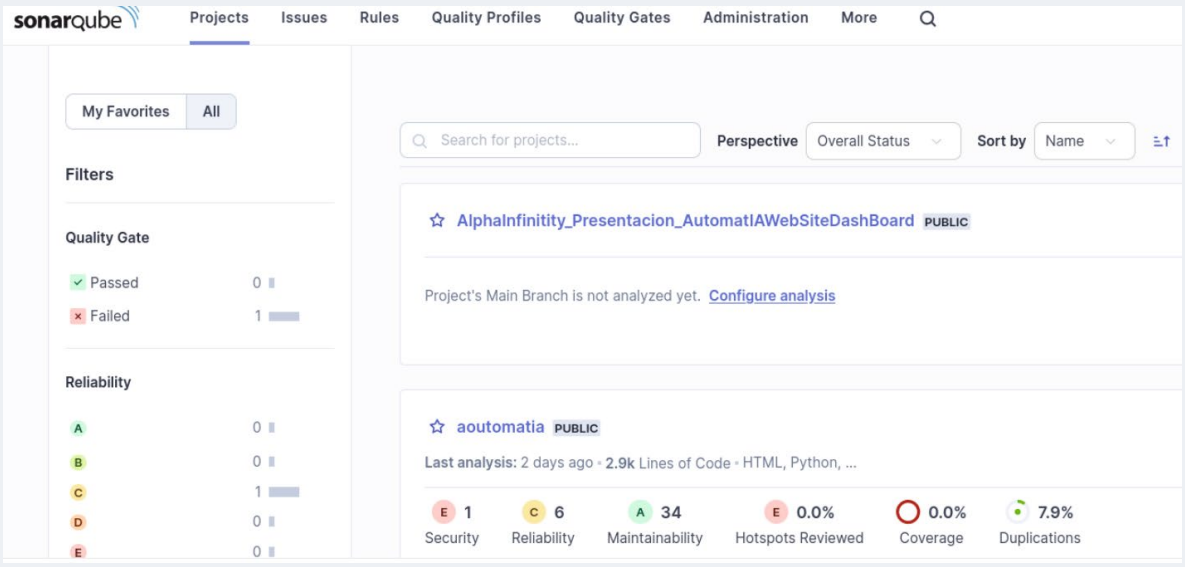
Plugin Output

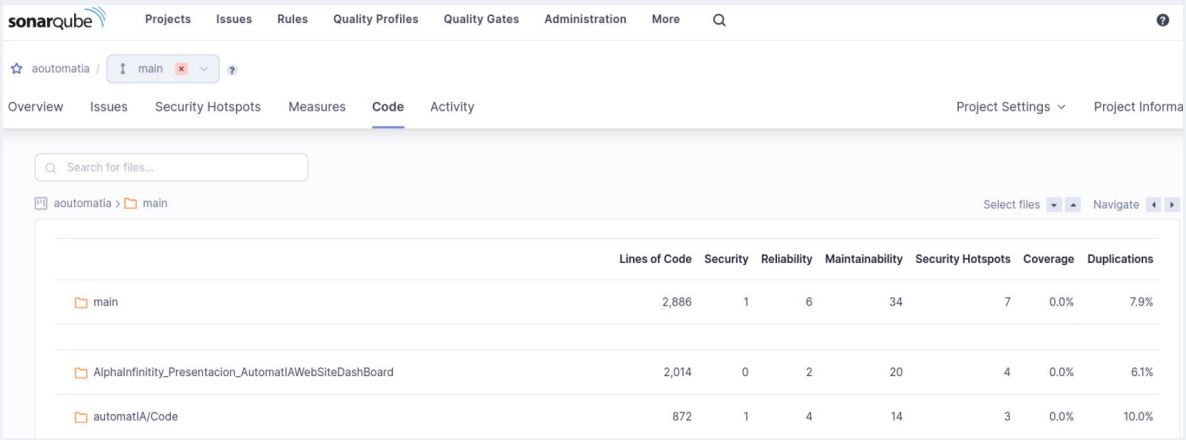
tcp/0

```
The remote host is up
The remote host replied to a TCP SYN packet sent to port 80 with a SYN,ACK packet
```

1.3 Findings SAST Summary

Static Application Security Testing (SAST) or static code analysis detects application vulnerability by scanning the source code, bytecode, or binaries of an application. By analyzing code patterns, control flows, and data flows within an application, SAST can identify a variety of vulnerabilities without running the application.





Only one security issue was discovered.

OVERALL SEVERITY LEVEL: Security

STATUS: CLOSED

1.4 Scope

1.4.1 Domains and IP Addresses

The following IP addresses and domains are the official scope given by the client:

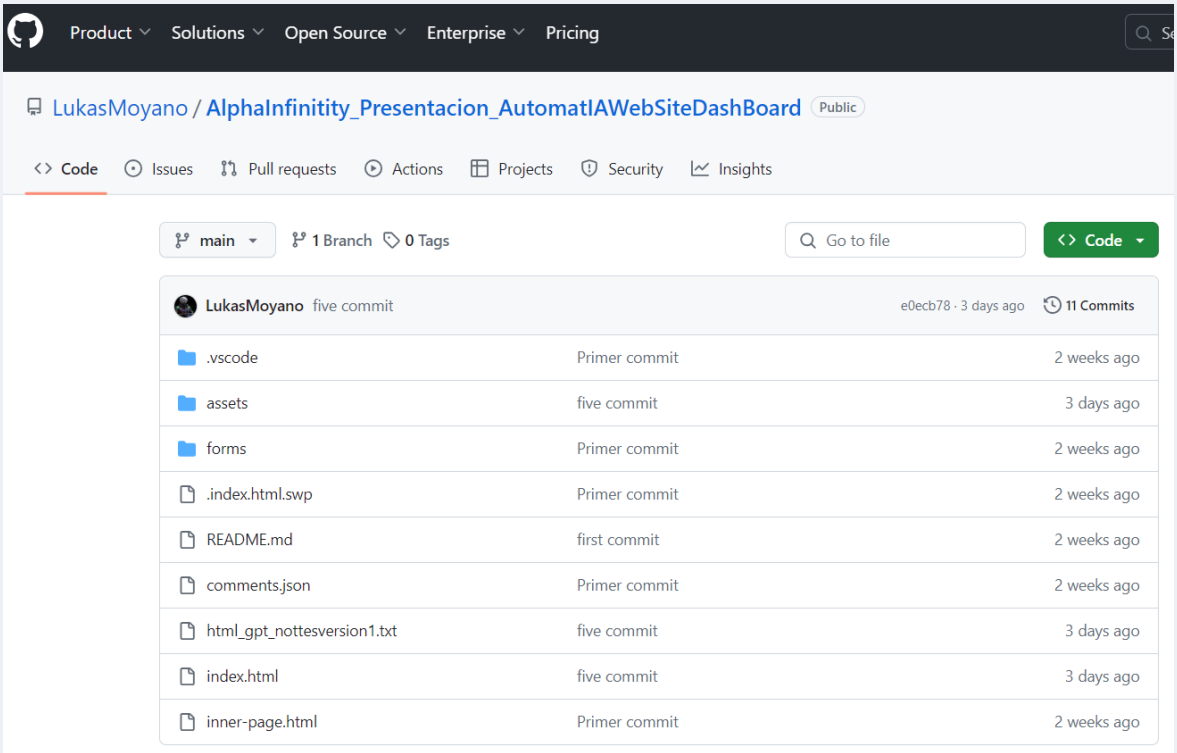
The uniform resource locator (URL) scanned:

https://main--alphainfinity-automatiadashboard.netlify.app

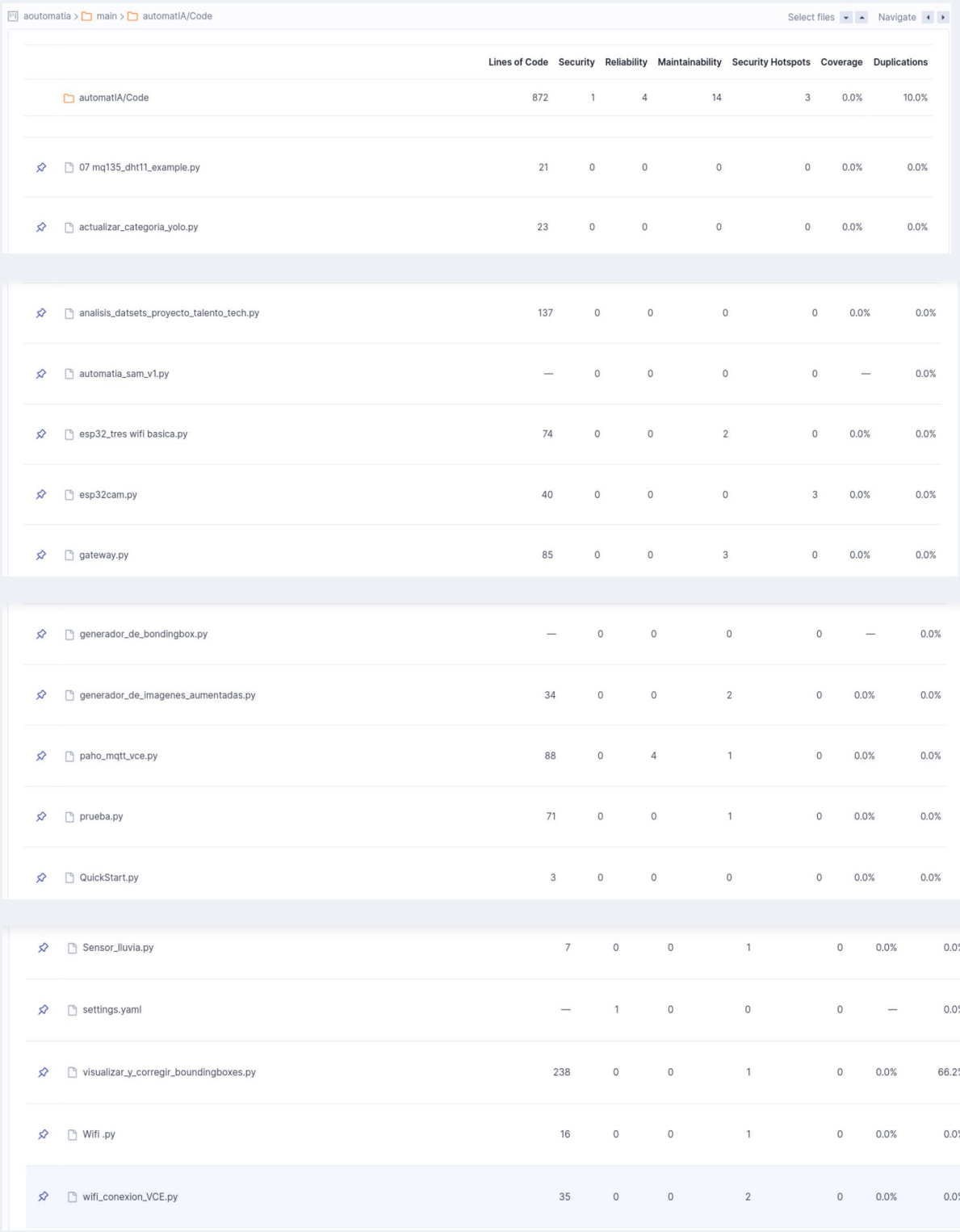
Host Information

DNS Name:	ec2-54-84-236-175.compute-1.amazonaws.com
IP:	54.84.236.175

1.4.2 URL source code on GitHub



1.4.3 ML Model and IoT Source Code



aoutomatia > main > automatIA/Code — Select files — Navigate

	Lines of Code	Security	Reliability	Maintainability	Security Hotspots	Coverage	Duplications
automatIA/Code	872	1	4	14	3	0.0%	10.0%
07 mq135_dht11_example.py	21	0	0	0	0	0.0%	0.0%
actualizar_categoria_yolo.py	23	0	0	0	0	0.0%	0.0%
analisis_datsets_proyecto_talento_tech.py	137	0	0	0	0	0.0%	0.0%
automatia_sam_v1.py	—	0	0	0	0	—	0.0%
esp32_tres wifi basica.py	74	0	0	2	0	0.0%	0.0%
esp32cam.py	40	0	0	0	3	0.0%	0.0%
gateway.py	85	0	0	3	0	0.0%	0.0%
generador_de_bondingbox.py	—	0	0	0	0	—	0.0%
generador_de_imagenes_aumentadas.py	34	0	0	2	0	0.0%	0.0%
paho_mqtt_vce.py	88	0	4	1	0	0.0%	0.0%
prueba.py	71	0	0	1	0	0.0%	0.0%
QuickStart.py	3	0	0	0	0	0.0%	0.0%
Sensor_lluvia.py	7	0	0	1	0	0.0%	0.0%
settings.yaml	—	1	0	0	0	—	0.0%
visualizar_y_corregir_boundingboxes.py	238	0	0	1	0	0.0%	66.2%
Wifi .py	16	0	0	1	0	0.0%	0.0%
wifi_conexion_VCE.py	35	0	0	2	0	0.0%	0.0%

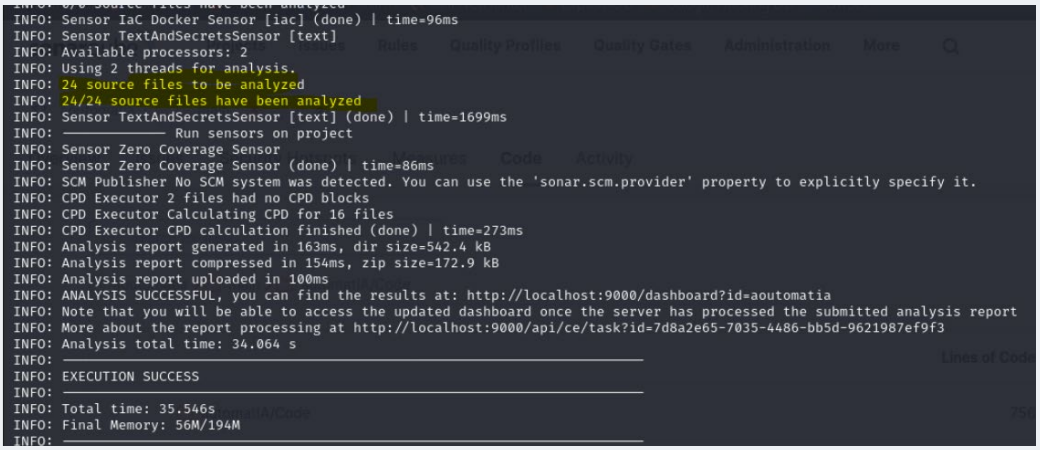
2 Findings

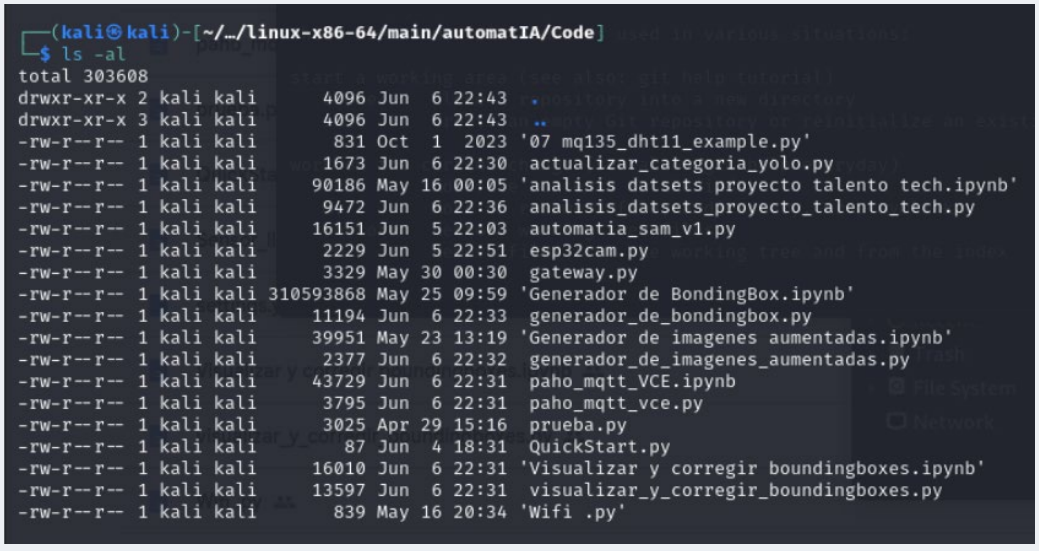
The tools used for this scan were:

NESSUS

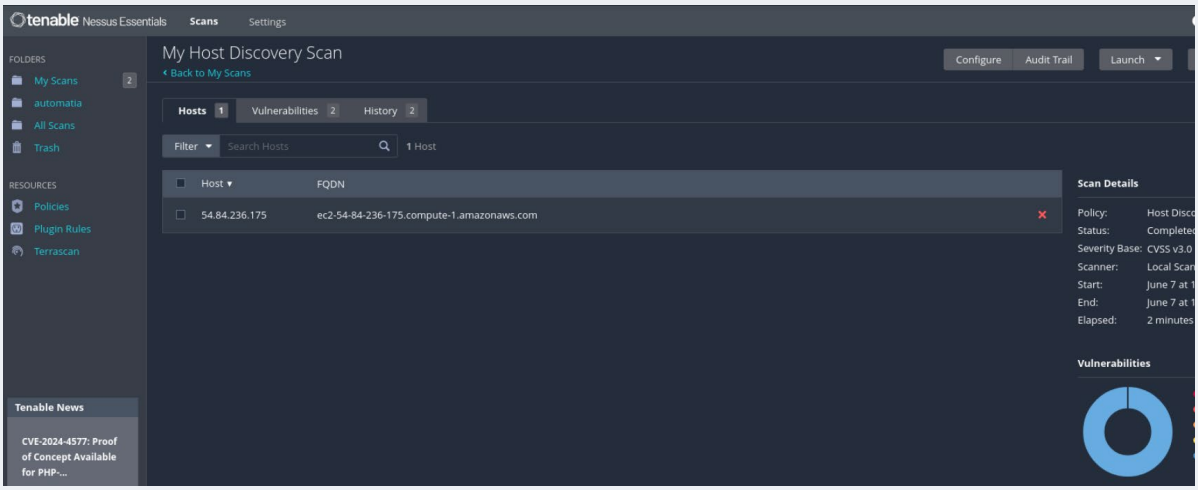
SonarQube

Running SonarQube





Running Nessus



Based on the results obtained it is possible to determine the severity of the impact on the security of the assets defined in the scope of this test. Each vulnerability will be classified as, CRITICAL, HIGH, MEDIUM, LOW, or INFORMATIVE. This classification will be supported by the levels published in CVSS-v3 which is available at: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator and OWASP Secure Coding Practice

This level of impact is determined using among others the following criteria:

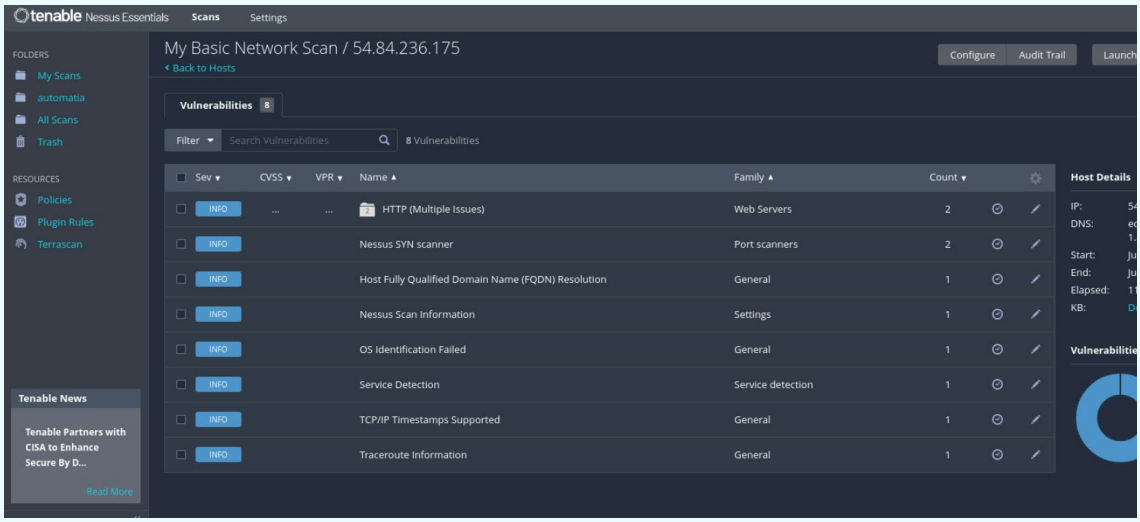
- Level of difficulty of the attacks executed (using exploitability metrics in CVSS).
- Level of damage that can be achieved with the attacks (using impact metrics in CVSS). • Impact of the attacks on the confidentiality, integrity, and availability of the system.

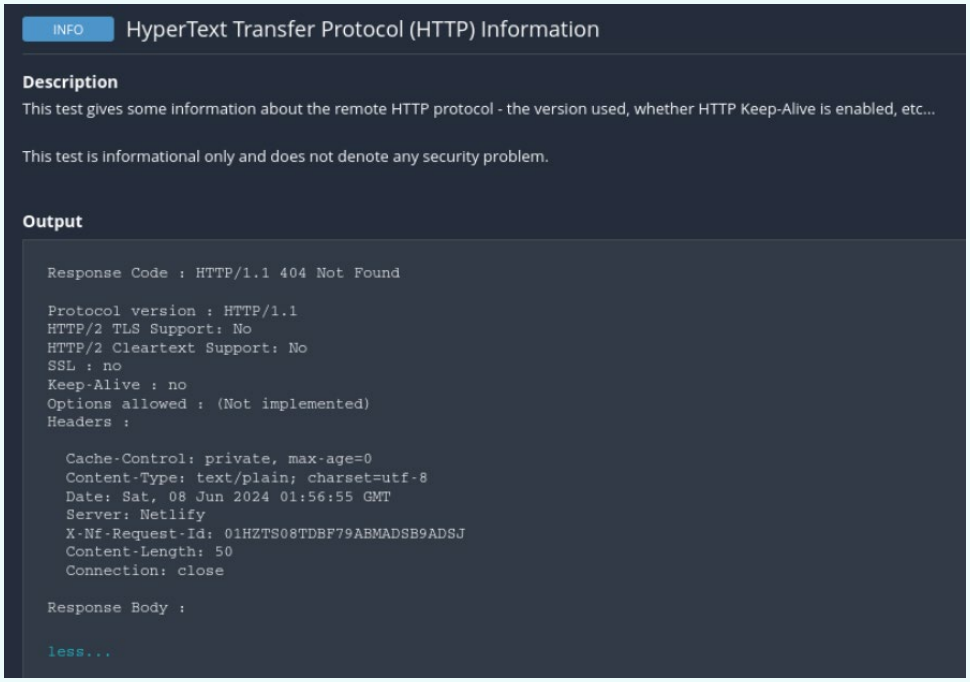
It is important to keep in mind that this pentest is limited in time and scope and that an attacker without such limitations could obtain different results.

Critical	*A finding is rated as critical if any of the following criteria is met:* • It is possible to gain access to the system with high level of privileges. • An exploitation of the vulnerability causes the system or mission-critical applications to stop, restart, or become unavailable. These actions do not require interaction with the user. • System access is achieved using well know public exploits. • if they have a CVSS3 base score of 9.0-10.0.
High	*A finding is rated as high if any of the following criteria is met:* • It is possible to gain access to the system with a low privileges profile. • An "Exploit" can be easily executed by an unauthenticated remote attacker, which causes the system or mission-critical applications to stop, restart, or become unavailable. These actions require access to local users. • System access is not achieved, but a weakly secured system access point can be attacked without such actions being detected. • The finding could represent an automatic failure to the client's compliance (for PCI DSS pentest). • If they have a CVSS3 base score of 7.0-8.9.
Medium	*A finding is rated as Medium if any of the following criteria is met:* • Access to adjacent systems. • An "Exploit" can be easily executed by an unauthenticated remote attacker which causes the system or mission-critical applications to stop, restart, or become unavailable. These actions require access to administrator accounts or local users. • Attempts to access were detected and systematically followed and scaled. • Any vulnerability that may generate the effects of a high classification criterion, but for which at the time of execution of the exercise it does not have published "exploit". • If they have a base CVSS3 score of 4.0-6.9.
Low	*A finding is rated as Low if any of the following criteria is met:* • Can allow an attacker to obtain system statistics, user account lists, and other information that is typically used to mount more sophisticated attacks. • It is also said that a vulnerability is low if all access points are protected and secured by sophisticated access control devices. • If they have a CVSS3 base score of 0.1-3.9.
Informative	*A finding is rated as Informative if any of the following criteria is met:* • These are just informative findings. • There is no direct severity associated. • If they have a CVSS3 base score of 0.0.

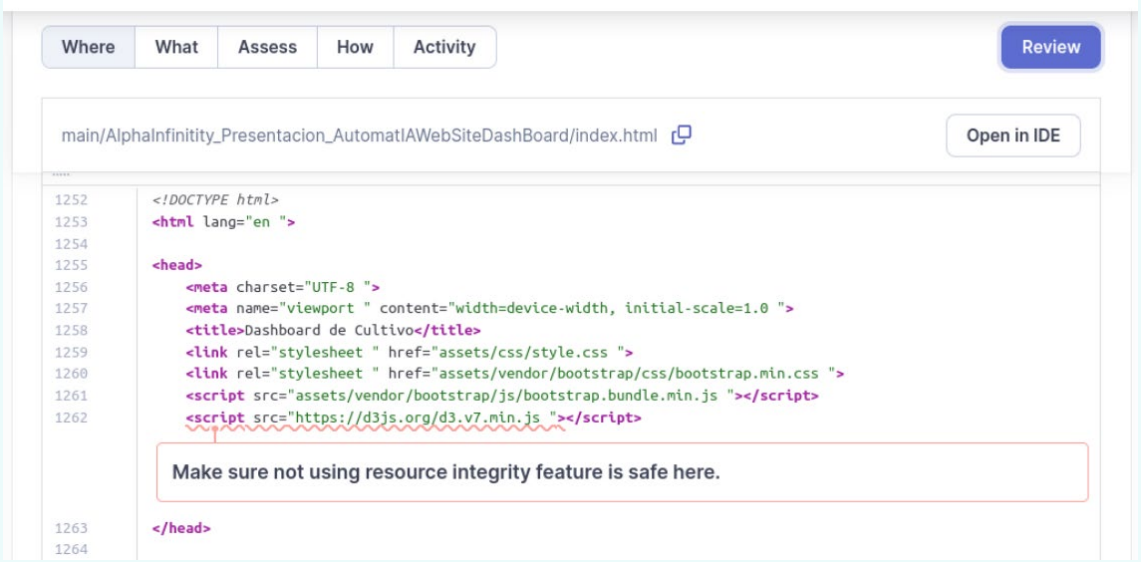
2.1 Important Findings

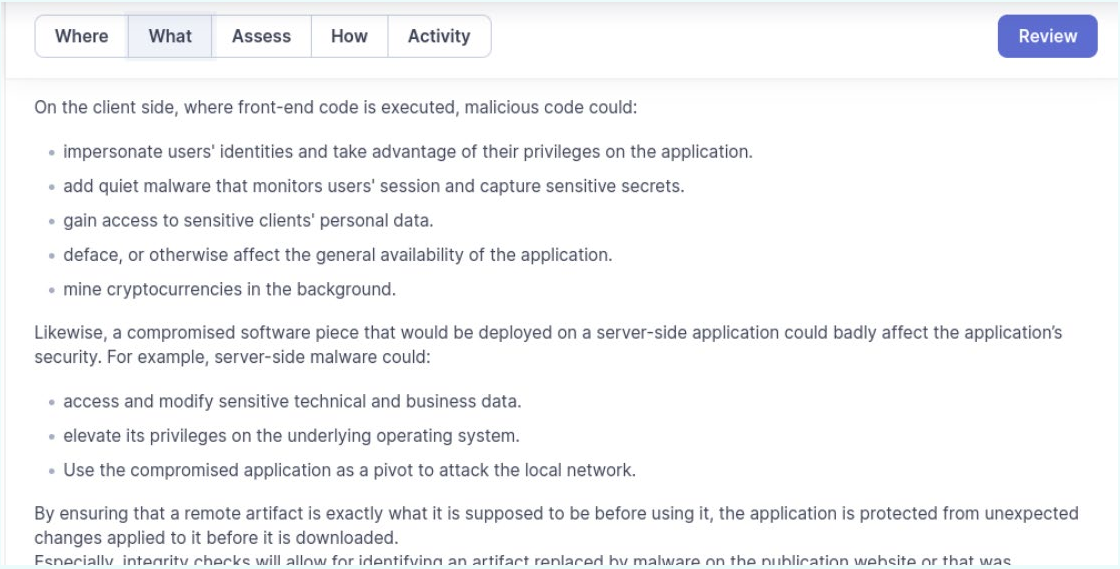
2.1.1 Informative Findings

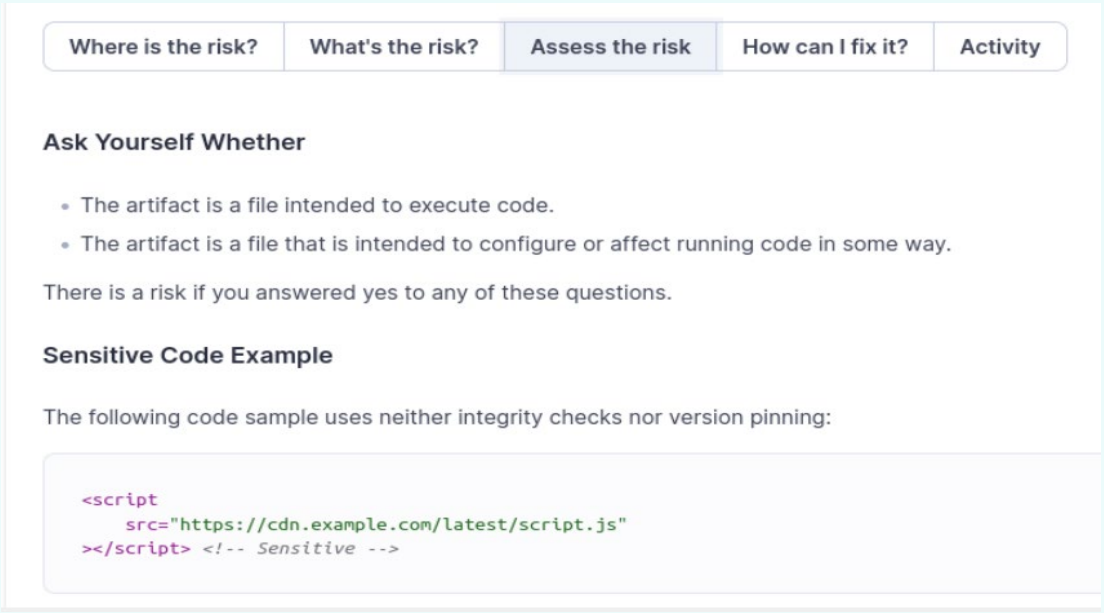




2.1.2 High Findings







Recommended Secure Coding Practices

To check the integrity of a remote artifact, hash verification is the most reliable solution. It does ensure that the file has not been modified since the fingerprint was computed.

In this case, the artifact's hash must:

- Be computed with a secure hash algorithm such as `SHA512`, `SHA384` or `SHA256`.
- Be compared with a secure hash that was **not** downloaded from the same source.

To do so, the best option is to add the hash in the code explicitly, by following Mozilla's official documentation on how to generate integrity strings.

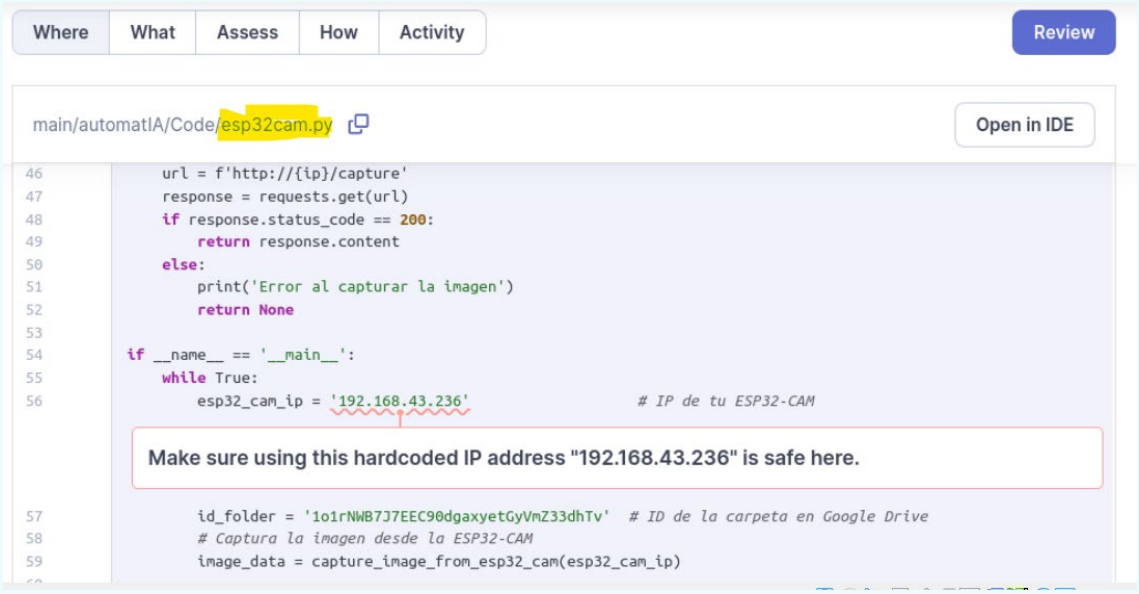
Note: Use this fix together with version binding on the remote file. Avoid downloading files named "latest" or similar, so that the front-end pages do not break when the code of the latest remote artifact changes.

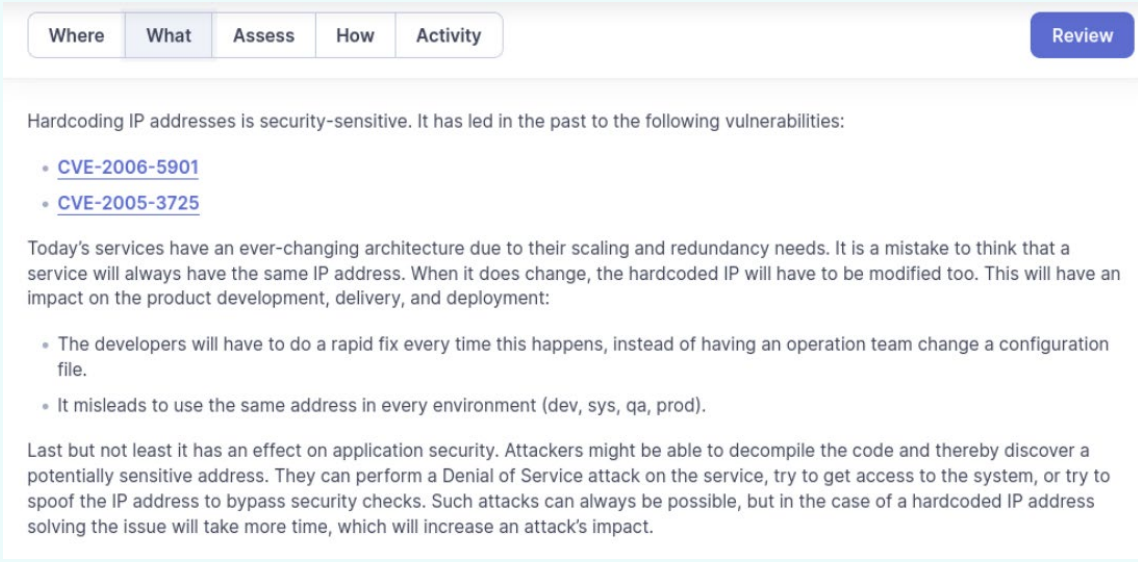
Compliant Solution

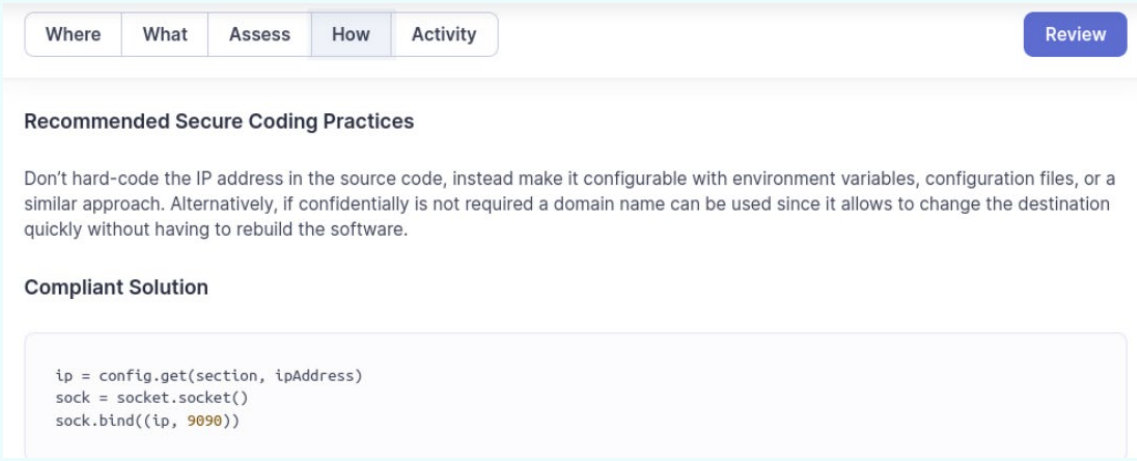
```
<script
    src="https://cdn.example.com/v5.3.6/script.js"
    integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQho1wx4JwY8wC"
></script>
```

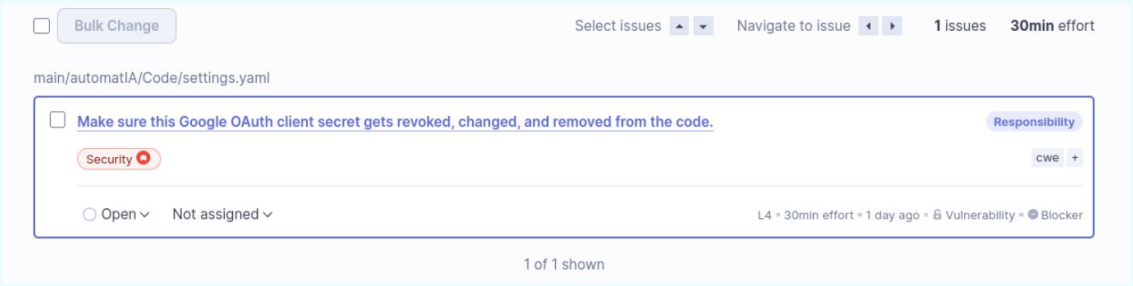
See

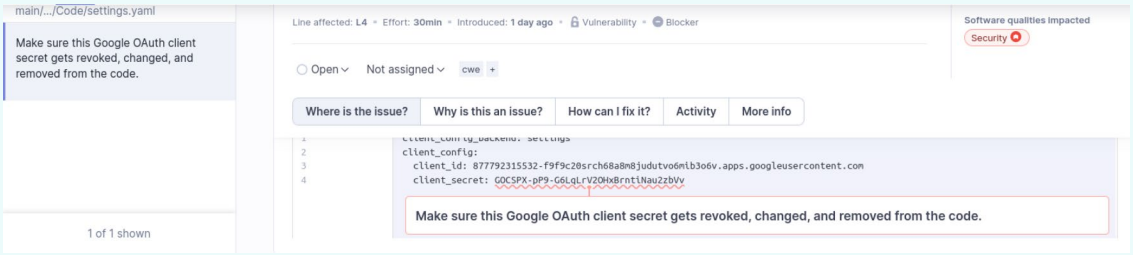
- OWASP - Top 10 2021 Category A8 - Software and Data Integrity Failures
- CWE - CWE-353 - Missing Support for Integrity Check
- OWASP - Top 10 2017 Category A6 - Security Misconfiguration
- developer.mozilla.org - Subresource Integrity
- Wikipedia, Watering Hole Attacks




Where
What
Assess
How
Activity
Review
Hardcoding IP addresses is security-sensitive. It has led in the past to the following vulnerabilities:
CVE-2006-5901
CVE-2005-3725
Today's services have an ever-changing architecture due to their scaling and redundancy needs. It is a mistake to think that a service will always have the same IP address. When it does change, the hardcoded IP will have to be modified too. This will have an impact on the product development, delivery, and deployment:
The developers will have to do a rapid fix every time this happens, instead of having an operation team change a configuration file.
It misleads to use the same address in every environment (dev, sys, qa, prod).
Last but not least it has an effect on application security. Attackers might be able to decompile the code and thereby discover a potentially sensitive address. They can perform a Denial of Service attack on the service, try to get access to the system, or try to spoof the IP address to bypass security checks. Such attacks can always be possible, but in the case of a hardcoded IP address solving the issue will take more time, which will increase an attack's impact.


Where
What
Assess
How
Activity
Review
Recommended Secure Coding Practices
Don't hard-code the IP address in the source code, instead make it configurable with environment variables, configuration files, or a similar approach. Alternatively, if confidentially is not required a domain name can be used since it allows to change the destination quickly without having to rebuild the software.
Compliant Solution
ip = config.get(section, ipAddress)
sock = socket.socket()
sock.bind((ip, 9090))


Bulk Change
Select issues
Navigate to issue
1 issues
30min effort
main/automatIA/Code/settings.yaml
Make sure this Google OAuth client secret gets revoked, changed, and removed from the code.
Responsibility
Security
cwe
Open
Not assigned
L4 • 30min effort • 1 day ago • Vulnerability • Blocker
1 of 1 shown


main/.../Code/settings.yaml
Make sure this Google OAuth client secret gets revoked, changed, and removed from the code.
1 of 1 shown
Line affected: L4 • Effort: 30min • Introduced: 1 day ago • Vulnerability • Blocker
Open
Not assigned
cwe
Software qualities impacted
Security
Where is the issue?
Why is this an issue?
How can I fix it?
Activity
More info
client_config:
client_id: 877792315532-f9f9c20srch68a8m8judutvo6mlb3o6v.apps.googleusercontent.com
client_secret: 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
Make sure this Google OAuth client secret gets revoked, changed, and removed from the code.

Where is the issue? | Why is this an issue? | How can I fix it? | Activity | More info

If a Google client OAuth secret leaks to an unintended audience, it can have serious security implications. Attackers who obtain the client secret can use it to impersonate the application and gain unauthorized access to user data. They can potentially access sensitive information, modify data, or perform actions on behalf of the user without their consent.

The exact capabilities of the attackers will depend on the authorizations the corresponding application has been granted.

Where is the issue? | Why is this an issue? | How can I fix it? | Activity | More info

Revoke the secret

Revoke any leaked secrets and remove them from the application source code.

Before revoking the secret, ensure that no other applications or processes are using it. Other usages of the secret will also be impacted when the secret is revoked.

Analyze recent secret use

When available, analyze authentication logs to identify any unintended or malicious use of the secret since its disclosure date. Doing this will allow determining if an attacker took advantage of the leaked secret and to what extent.

This operation should be part of a global incident response process.

Google Cloud console provides a Logs Explorer feature that can be used to audit recent access to a cloud infrastructure.

Use a secret vault

A secret vault should be used to generate and store the new secret. This will ensure the secret's security and prevent any further unexpected disclosure.

Depending on the development platform and the leaked secret type, multiple solutions are currently available.

Noncompliant code example

```
props.set("client_secret", "TgxYWFmND-1NTYwNTgzMDM3N") // Noncompliant
```

Compliant solution

```
props.set("client_secret", System.getenv("CLIENT_SECRET"))
```

main/automatIA/Code/generador_de_imagenes_aumentadas.py

Add replacement fields or use a normal string instead of an f-string. Intentionality

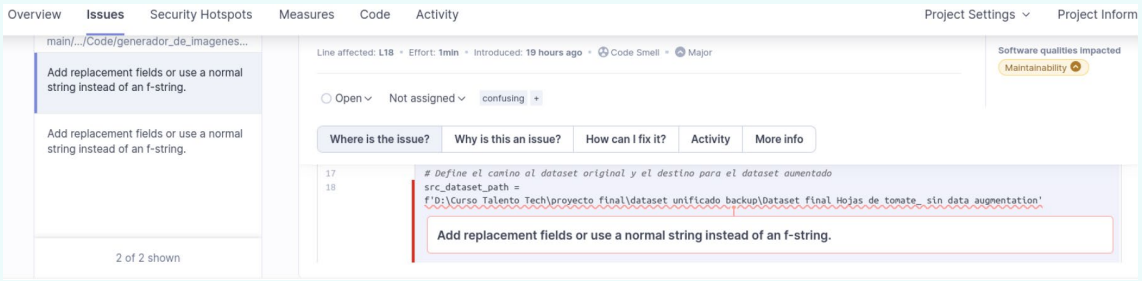
Maintainability confusing +

Open Not assigned L18 • 1min effort • 19 hours ago • Code Smell • Major

Add replacement fields or use a normal string instead of an f-string. Intentionality

Maintainability confusing +

Open Not assigned L19 • 1min effort • 19 hours ago • Code Smell • Major



Where is the issue? | Why is this an issue? | How can I fix it? | Activity | More info

A format string is a string that contains placeholders, usually represented by special characters such as "%s" or "{}", depending on the technology in use. These placeholders are replaced by values when the string is printed or logged. Thus, it is required that a string is valid and arguments match replacement fields in this string.

This applies to the % operator, the str.format method, and loggers from the logging module. Internally, the latter use the `%-formatting`. The only difference is that they will log an error instead of raising an exception when the provided arguments are invalid.

Formatted string literals (also called "f-strings"; available since Python 3.6) are generally simpler to use, and any syntax mistake will cause a failure at compile time. However, it is easy to forget curly braces, which will not lead to any detectable errors.

Where is the issue? | Why is this an issue? | How can I fix it? | Activity | More info

A `printf-`-style format string is a string that contains placeholders, which are replaced by values when the string is printed or logged. Mismatch in the format specifiers and the arguments provided can lead to incorrect strings being created.

To avoid issues, a developer should ensure that the provided arguments match format specifiers.

Where is the issue? | Why is this an issue? | How can I fix it? | Activity | More info

A `printf-`-style format string is a string that contains placeholders, which are replaced by values when the string is printed or logged. Mismatch in the format specifiers and the arguments provided can lead to incorrect strings being created.

To avoid issues, a developer should ensure that the provided arguments match format specifiers.

Noncompliant code example

```
"Error %(message)s" % {"message": "something failed", "extra": "some dead code"}  # Noncompliant. Remove the unused argument "extra" or add a replac
"Error: User {} has not been able to access []".format("Alice", "MyFile")  # Noncompliant. Remove 1 unexpected argument or add a replacement field.
```

Compliant solution

```
"Error %(message)s" % {"message": "something failed"}
"Error: User {} has not been able to access {}".format("Alice", "MyFile")
```

```
user = "Alice"
resource = "MyFile"
message = f"Error: User {user} has not been able to access {resource}"

import logging
logging.error("Error: User %s has not been able to access %s", "Alice", "MyFile")
```

3 CONCLUSIONS AND RECOMMENDATIONS

3.1 Conclusion

During the tests and analysis of the client's IP addresses and Source code, AlphaInfinity was able to identify and confirm vulnerabilities in the external accessible systems and in the code. This can affect the integrity, availability, and confidentiality of the client computer systems.

These findings conclude that there are areas that are affected by unsafe configurations. Details such as application vulnerabilities, security misconfigurations and information disclosure can put the client's infrastructure in a vulnerable position in the face of cyber-attacks.

The operational impact that these vulnerabilities can have on the business can affect the confidentiality, integrity, and in the most malicious cases the systems availability. Even allowing an unauthorized user system access to extract sensitive information.

Based exclusively on the results of this analysis and considering the scope of the tests at the time of its execution, AlphaInfinity considers that the platform reflects the following state of security criticality**:** **HIGH**.

3.2 Recommendations

Based on the tests performed and their results, the following recommendations are provided:

- **Remediation and Mitigation Plan**

  It is recommended immediately, that the client develops a remediation/mitigation plan using this report, to correct the identified vulnerabilities, minimize the potential risks and reduce the attack surface.

- **Periodic Vulnerability scans and penetration tests**

  Due to the constant vulnerabilities and threats, it is recommended to do a quarterly vulnerability scan, and a minimum external/internal penetration test once a year or when required by changes to the infrastructure or applications.

- **Perform a review test (re-test)**

  Perform a review test to verify that these findings have been successfully corrected. This re-test must be done no later than 90 days after the report was delivered.

- **Reviewing security events via SIEM system**

  Validate that the vulnerability scan and related activities were logged and detected by the security event logging and correlation system (SIEM) that handles alerts and incidents.

- **Security Awareness Program**

  Training and social engineering exercises should be conducted as an on-going program to ensure that the users engage in a high level of security awareness at all times. This provides maturity and increases the security-related posture of the client.