ESKA your ITeam

Penetration Test Report

*****
*****

ESKA

Hlybochytska Street, 17B, Kyiv, Ukraine

office@eska.global

HYPERLINK "https://en.eska.global/"
eska.global

+380 (44) 247 10 21

Table of Contents

Executive Summary.....3
Summary of Results.....5
1. INFRASTRUCTURE ASSESSMENT.....6
1.1. Discovery.....6
1.2. Cloud SQL testing results.....7
1.3. Cloud Storage testing results.....9
1.4. Cloud Compute Engine testing results.....9
1.5. IAM testing results.....11
1.6. Kubernetes Engine testing results.....13
1.7. Stackdriver Logging & Monitoring testing results.....16
1.8. Conclusion.....18
2. APPLICATION ASSESSMENT.....19
2.1. Introduction.....19
2.2. User enumeration.....20
2.3. Google Captcha bypass.....23
2.4. SSRF on JSON API functionality.....25
2.5. Stored XSS.....28
2.6. IDOR-Privilege Escalation.....30
2.7. Code Review.....33
2.8. Informational: Advices.....35
2.9. Risk Rating.....37
Appendix C: About ESKA.....38

Executive Summary

ESKA was contracted by ***** to conduct a penetration test in order to determine its exposure to a targeted attack, and Infrastructure Assessment to evaluate configurations regarding security best practices. All activities regarding penetration test were conducted in a manner that simulated a malicious actor engaged in a targeted attack against ***** with the goals of:

- Identify if a remote attacker could penetrate *****'s defenses.
- Determine the impact of a security breach on:
    - Confidentiality of the company's private data;
    - Internal infrastructure and availability of *****'s information systems.

Penetration test was expanded with source code analysis for determination of programming errors and unsecure data flows. Efforts were placed on the identification and exploitation of security weaknesses that could allow a remote attacker to gain unauthorized access to organizational data. The attacks were conducted with the level of access that a general application user would have. The source code analysis was conducted with provided by ***** credentials and accesses. The assessment was conducted in accordance with the recommendations outlined in NIST SP 800-115 "Technical Guide to Information Security Testing and Assessment" with all tests and actions being conducted under controlled conditions.

All activities regarding Infrastructure Assessment were conducted according to Google Cloud Platform (GCP) security best practices with the goals of:

- Ensure that necessary security controls are integrated into the design and implementation of a project.

- Check and evaluate security configurations that should ensure the Confidentiality, Integrity and Availability of ***** sensitive data and other resources.

Summary of Results

Initial reconnaissance of the ***** infrastructure resulted in discovery of settings that need attention. The results provided us with a listing of specific settings in the infrastructure. An examination of the Google Cloud Infrastructure revealed 2 HIGH-level and 526 WARNING-level issues within 2 projects (35 total). After using a custom “Gray Box” technique on the ***** infrastructure we were able to find a list of issues according to Google Security Checklist. HIGH-level and some WARNING-level issues were additionally checked with custom scripts and techniques, set of tools like Burp, MetaSploit, etc. There are no critical results, but this requires additional attention anyway. Uncovering the passwords via brute-force was not conducted. Cloud penetration testing (uses simulated cyberattacks against target systems to identify vulnerabilities) engages concepts that are performed on cloud-native systems. This form of security testing is used to identify security risks and vulnerabilities, and provide actionable remediation advice.

Initial reconnaissance of the *****’s network resulted in the discovery of a User Enumeration vulnerability that allows an attacker to enumerate registered emails that exist in application. With Google Captcha Bypass vulnerability there is a possibility to brute force users’ passwords and get access to users’ accounts. While using provided credentials of the user Company.MEMBER there were found an IDOR vulnerability, that allows this user to change company name and the avatar of the company, and Stored XSS vulnerability. Additionally, there were found 2 vulnerabilities regarding API with **CRITICAL**-risk and **HIGH**-risk ratings. Other vulnerabilities have **LOW** and Informational risk ratings but still should be considered to be remediated.

1. INFRASTRUCTURE ASSESSMENT

1.1. Discovery

For the purposes of this assessment, CLIENT provided a cloud account with View permission, suitable for "Gray Box" Pentest. During the enumeration stage there were found 2 projects that require attention (Figure 1).

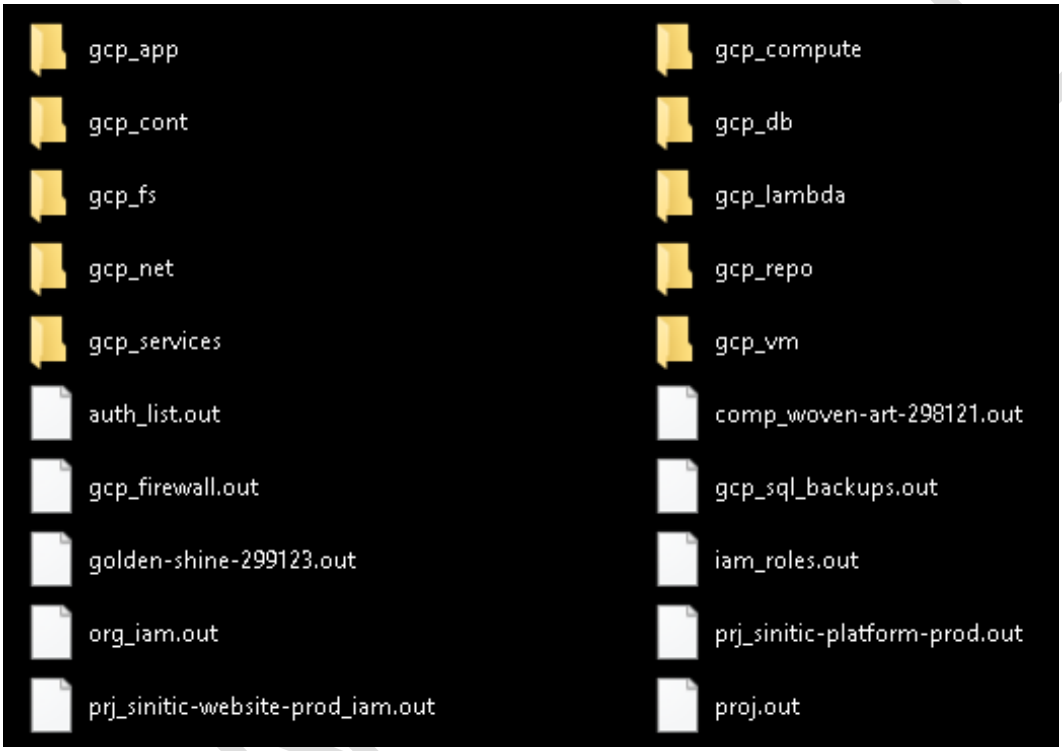


Figure 1 – Discovery process result files

1.2. Cloud SQL testing results

1. **Cloud SQL Database Instances Have Public IPs**

   Severity: HIGH

   Description:

   To lower the organization's attack surface, Cloud SQL databases should not have public IPs. Private IPs provide improved network security and lower latency for your application.

   Remediation:

   From console:

   1. Go to the Cloud SQL Instances page in the Google Cloud Console by visiting https://console.cloud.google.com/sql/instances.
   2. Click the instance name to open its Instance details page.
   3. Select the Connections tab.
   4. Deselect the Public IP checkbox.
   5. Click Save to update the instance.

   Compliance:

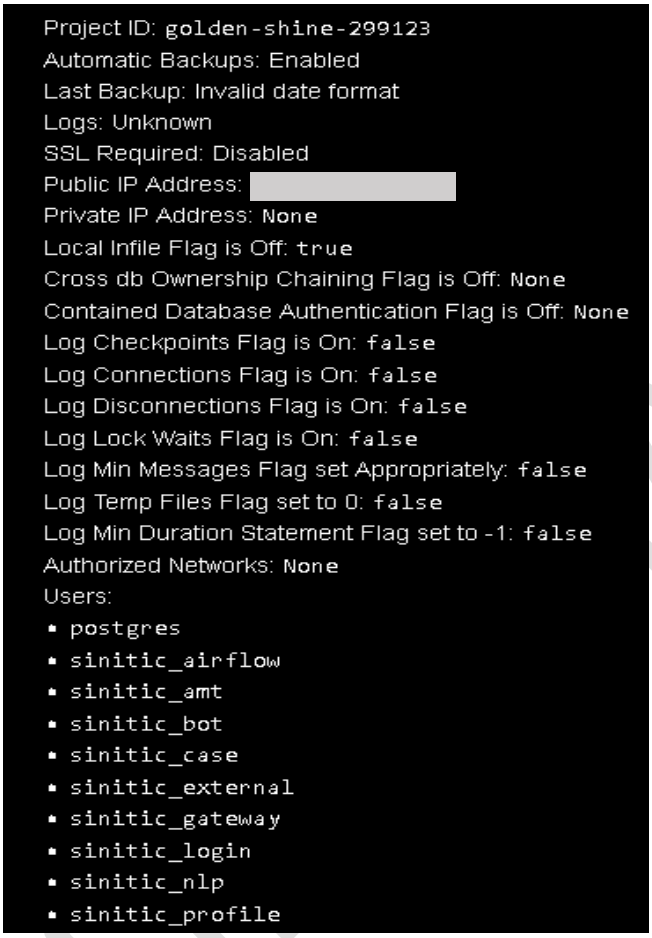
   CIS Google Cloud Platform Foundations version 1.3.0

   Affected Projects:

   [.....HIDDEN INFO.....]

   Databases:

   [.....HIDDEN INFO.....]



1.3. Cloud Storage testing results

1. **Bucket with Logging Disabled**

   Severity: Medium

   Description:

   Enable “access and storage logs” in order to capture all events which may affect objects within target buckets.

   Compliance:

   CIS Google Cloud Platform Foundations version 1.0.0, reference 5.3

   References:

   https://cloud.google.com/storage/docs/access-logs

   Buckets affected:

   [.....HIDDEN INFO.....]

1.4. Cloud Compute Engine testing results

Default Firewall Rule in Use

Severity: Medium

Description:

Some default firewall rules were in use. This could potentially expose sensitive services or protocols to other networks.

Rules:

[.....HIDDEN INFO.....]

Example figure:

1.5. IAM testing results

1. **Basic Role in Use**

   Severity: Medium

   Description:

   Basic roles grant significant privileges. In most cases, usage of these roles is not recommended and does not follow security best practice.

   Compliance:

   CIS Google Cloud Platform Foundations version 1.0.0, reference 1.4

   CIS Google Cloud Platform Foundations version 1.1.0, reference 1.5

   References:

   https://cloud.google.com/sdk/gcloud/reference/iam/service-accounts/

   https://cloud.google.com/iam/docs/understanding-roles

   https://cloud.google.com/iam/docs/understanding-service-accounts

   Affected Roles:

   [.....HIDDEN INFO.....]

2. **Service Account with Admin Privileges**

   Severity: Medium

   Description:

   Service accounts represent service-level security of the Resources (application or a VM) which can be determined by the roles assigned to it. Enrolling Service Accounts with administrative privileges grants full access to assigned application or a VM, Service Account Access holder can use.

   Compliance:

   CIS Google Cloud Platform Foundations version 1.0.0, reference 1.4

   CIS Google Cloud Platform Foundations version 1.1.0, reference 1.5

   References:

https://cloud.google.com/sdk/gcloud/reference/iam/service-accounts/

https://cloud.google.com/iam/docs/understanding-roles

https://cloud.google.com/iam/docs/understanding-service-accounts

Affected Accounts:

[.....HIDDEN INFO.....]

1.6. Kubernetes Engine testing results

1. Clusters Lacking Labels

Severity: Medium

Description:

Labels enable users to map their own organizational structures onto system objects in a loosely coupled fashion, without requiring clients to store these mappings. Labels can also be used to apply specific security settings and auto configure objects at creation.

Compliance:

CIS Google Cloud Platform Foundations version 1.0.0, reference 7.5

References:

https://cloud.google.com/kubernetes-engine/docs/how-to/hardening-your-cluster#use_namespaces_and_rbac_to_restrict_access_to_cluster_resources

Affected Clusters:

[.....HIDDEN INFO.....]

2. Default Service Account in Use

Severity: Medium

Description:

You should create and use a minimally privileged service account to run your Kubernetes Engine cluster instead of using the Compute Engine default service account.

Compliance:

CIS Google Cloud Platform Foundations version 1.0.0, reference 7.17

CIS GKE Benchmark version 1.0.0, reference 6.2.1

References:

https://www.cisecurity.org/benchmark/kubernetes/

https://cloud.google.com/kubernetes-engine/docs/how-to/hardening-your-cluster#use_least_privilege_sa

https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks#default_values_on

Affected Clusters:

[.....HIDDEN INFO.....]

3. Pod Security Policy Disabled

Severity: Medium

Description:

A Pod Security Policy is a cluster-level resource that controls security sensitive aspects of the pod specification. The PodSecurityPolicy objects define a set of conditions that a pod must run with in order to be accepted into the system, as well as defaults for the related fields.

Compliance:

CIS Google Cloud Platform Foundations version 1.0.0, reference 7.14

CIS GKE Benchmark version 1.0.0, reference 6.10.3

References:

https://www.cisecurity.org/benchmark/kubernetes/

https://cloud.google.com/kubernetes-engine/docs/how-to/pod-security-policies

https://kubernetes.io/docs/concepts/policy/pod-security-policy

https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks#default_values_on

Affected Clusters:

[.....HIDDEN INFO.....]

1.7. Stackdriver Logging & Monitoring testing results

1. Log Metric Filter Issues

Severity: Medium

Log Metric Filter Doesn't Exist for Audit Configuration Changes

Description:

Configuring the metric filter and alerts for audit configuration changes ensures the recommended state of audit configuration is maintained so that all activities in the project are audit-able at any point in time.

Log Metric Filter Doesn't Exist for Cloud Storage IAM Permission Changes

Description:

Monitoring changes to cloud storage bucket permissions may reduce the time needed to detect and correct permissions on sensitive cloud storage buckets and objects inside the bucket.

Log Metric Filter Doesn't Exist for Custom Role Changes

Description:

Google Cloud IAM provides predefined roles that give granular access to specific Google Cloud Platform resources and prevent unwanted access to other resources. However, to cater to organization-specific needs, Cloud IAM also provides the ability to create custom roles. Project owners and administrators with the Organization Role Administrator role or the IAM Role Administrator role can create custom roles. Monitoring role creation, deletion and updating activities will help in identifying any over-privileged role at early stages.

Log Metric Filter Doesn't Exist for Project Ownership Assignments/Changes

Description:

Project ownership has the highest level of privileges on a project. To avoid misuse of project resources, the project ownership assignment/change actions mentioned above should be monitored and alerted to concerned recipients.

Log Metric Filter Doesn't Exist for SQL Instance Configuration Changes

Compliance:

CIS Google Cloud Platform Foundations version 1.1.0

References:

https://cloud.google.com/logging/docs/logs-based-metrics/

https://cloud.google.com/monitoring/custom-metrics/

https://cloud.google.com/monitoring/alerts/

https://cloud.google.com/logging/docs/reference/tools/gcloud-logging

Affected Logging Configurations:

[.....HIDDEN INFO.....]

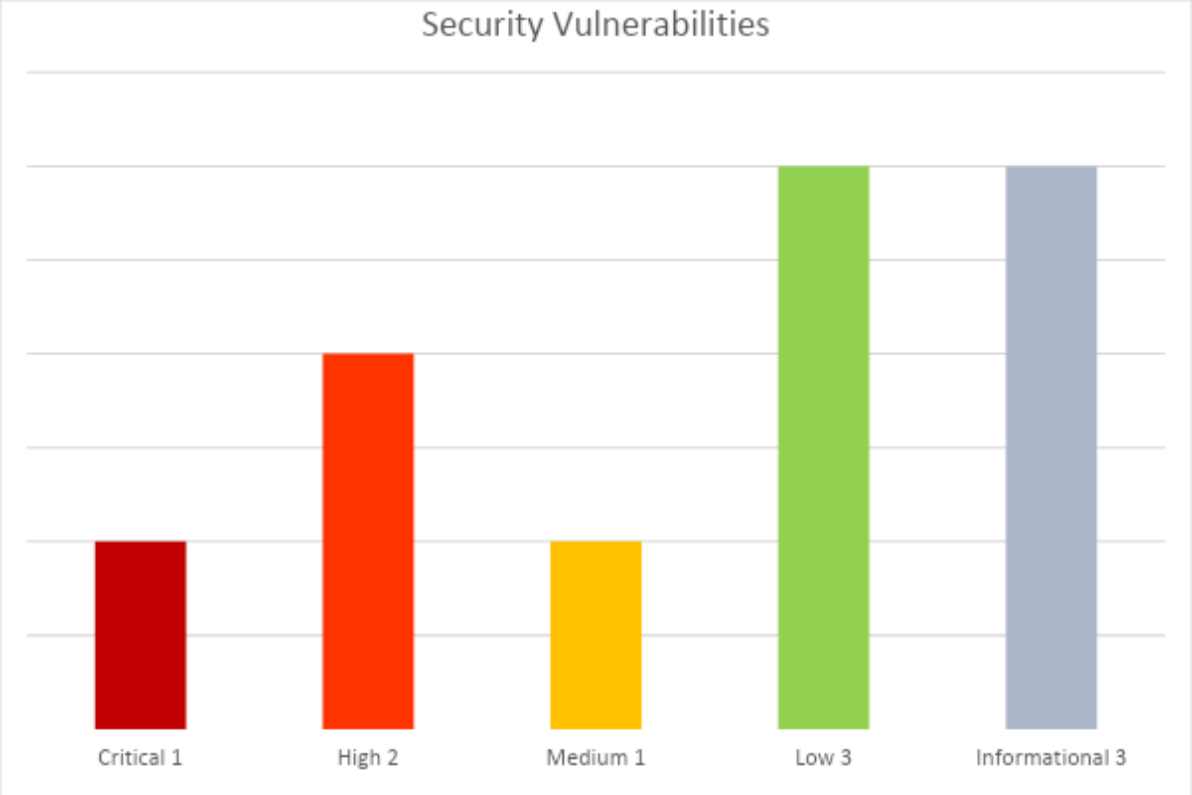
1.8. Conclusion

An examination of the Google Cloud Infrastructure revealed 2 HIGH-level and 526 WARNING-level issues within 2 projects (35 total). After using a custom “Gray Box” technique on the ***** infrastructure we were able to find a list of issues according to Google Security Checklist. HIGH-level and some WARNING-level issues were additionally checked with custom scripts and techniques and with set of tools like Burp, MetaSploit, etc. There are no critical results, but this still requires additional attention.

2. APPLICATION ASSESSMENT

2.1. Introduction

The cybersecurity team performed Pentest on the *****'s application using a blackbox and whitebox approaches, simulating attack vectors that attackers could perform in real life. When the web application was tested, 7 security vulnerabilities were found and rated for their critical, high, medium and low level.



2.2. User enumeration

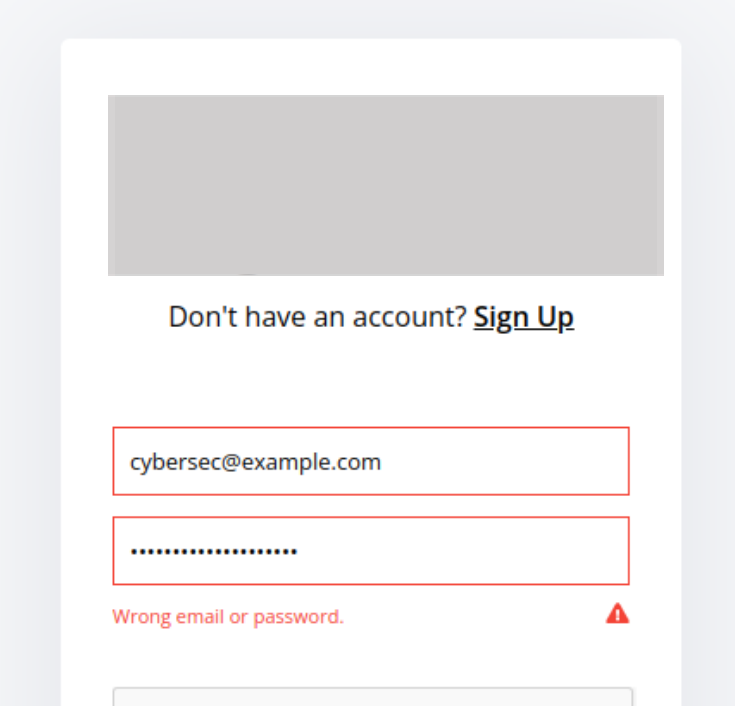
Category: OWASP Top 10 (A07:2021-Identification and Authentication Failures)

Severity: Low

Vulnerability explanation:

User enumeration is a type of attack when a malicious actor can use brute-force techniques to either guess or confirm valid users in a system. In the login page of the *****'s platform, if either one of the email or password is wrong, "Wrong email or password." message is displayed to the user. Thus, it is not indicated which (email or password) is incorrect.

For example, in the screenshot below, although the email address cybersec@example.com is registered, the response message does not show any information about it.



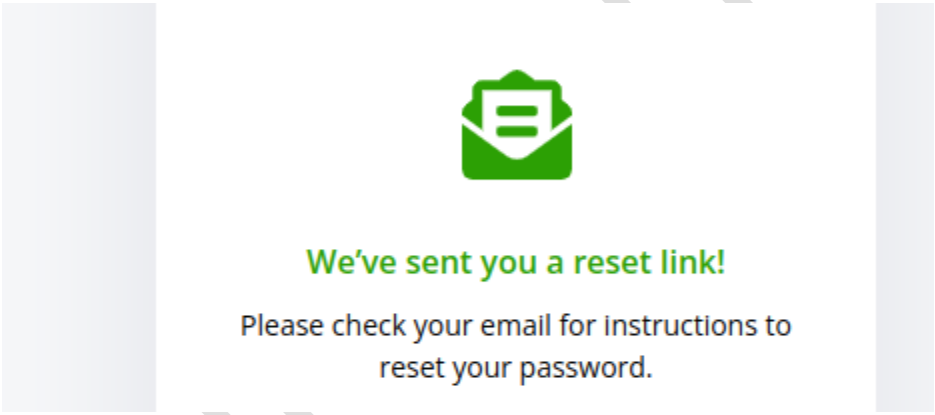
However, this mechanism is not implemented on the “password reset” and “registration” pages. Therefore, an attacker can find out whether any email address is registered on the *****’s platform.

Exploitation process:

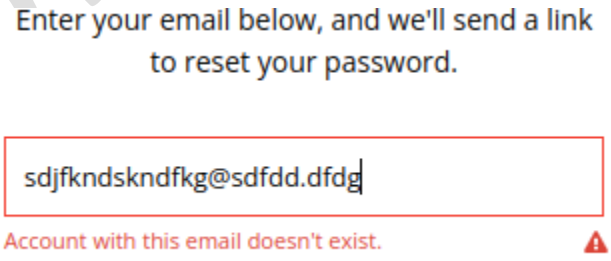
To verify the vulnerability, we will use one existing (cybersec@example.com) and one non-existing email addresses.

Steps to Reproduce:

1. In the password reset page, when the user enters an existing mail address, the following message is displayed.



But, with the non-existing email, the user will see the following message:



2. In the registration page, if the user tries to register with an existing email address, the following error message will be displayed.

But, with a non-existing one, a new account will be created.

Based on these two response messages, it's possible to determine whether an email address is registered.

Remediation:

The same response should be returned whether the email address entered by the user exists or not.

2.3. Google Captcha bypass

Category: A2:2017 – Broken Authentication

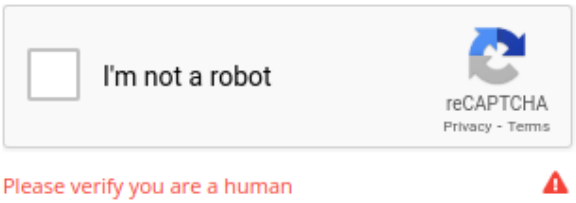
Severity: Medium

Vulnerability explanation:

There are 3 pages that use reCAPTCHA: Login, Password Reset, Registration. While testing these functions, it turned out that captcha provided by reCAPTCHA is not validated.

Exploitation process:

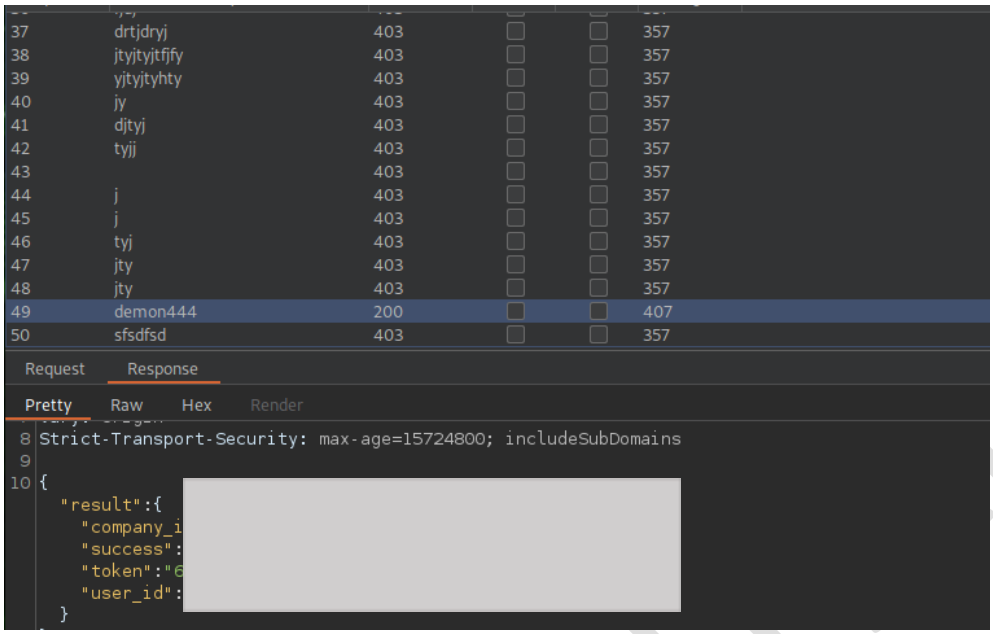
- In the login page, Captcha is required



- When the user confirms the captcha and clicks to “Login” button, a request is sent as follows:
  [.....HIDDEN INFO.....]

- The problem here is, if we re-use captcha token, or even remove it completely, the server doesn't show any error message, and the request is accepted.
  [.....HIDDEN INFO.....]

- Further analysis revealed that there is no rate-limiting here either. This means that an attacker can try as many passwords as he wants until he finds the right one.
- The screenshot below proves that despite checking 50 passwords, no protection mechanism is triggered



In this simulation the attacker tried 50 passwords and was able to find the correct one in the 49th request. In the response, the victim's user session token is sent to the attacker.

Note: The captcha doesn't work on any of the 3 pages named above. Each of them has its own impact. These are:

- Login – Brute-Force
- Password reset – Sending large volumes of email from proto's mail address.
- Registration – Creating a large number of fake accounts.

Remediation:

There is a logical flaw in captcha implementation. Make sure that every request is checked for the correct captcha and is then processed.

2.4. SSRF on JSON API functionality

Category: A10:2021 – SSRF

Severity: **Critical**

Vulnerability explanation:

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

According to the documentation, the "JSON API" bot block provides the user with the ability to send an HTTP request to another (bot owner's) server.

But, it's also possible to send HTTP requests to internal addresses that are belong to *****'s environment and should not be accessible by users.

While analyzing the application, it turned out that the environment is deployed on Google Cloud platform. So, it automatically has access to the metadata server API **without any additional authorization**. As the requests are issued from the server (VM), an attacker can access metadata server API and retrieve **confidential** data from the VM metadata server.

Exploitation process:

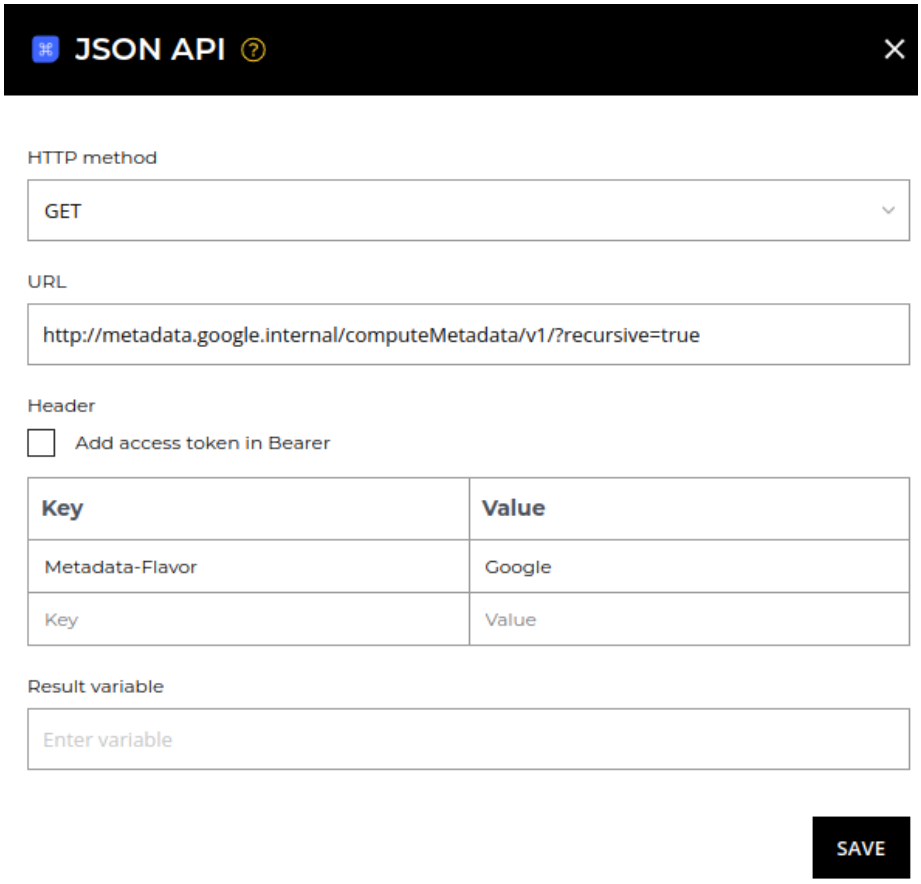
As Proof of Concept, we'll retrieve VM metadata from Google Cloud VM metadata server

1. Attacker creates a new "JSON API" bot block.

2. As a URL, the address of the metadata server of the google cloud system is entered. (http://metadata.google.internal/computeMetadata/v1/?recursive=true)

3. Regarding Google Cloud's documentation, to query metadata information, "Metadata-Flavor" header must be in all requests. In traditional SSRF exploitation, it's

not possible to add HTTP header to requests. But, as “JSON API” block provides users with the ability to add headers to requests, it’s not a problem for an attacker



4. Attacker saves the block and clicks the “Test Chat” button.

5. When the chat starts, the HTTP request is sent. It’s possible to view webhook history on ***** page.

In this page, we can see requests and responses that are sent via JSON API block.

[.....HIDDEN INFO.....]

In this response, confidential information such as Kubernetes environment variables, Network interfaces, service accounts, SSH keys is leaked.

Example of leaked data:

[.....HIDDEN INFO.....]

Since staging and development environments are accessible, we've tested this vulnerability on those environments. There are more local users that use SSH

- *****
- *****
- *****

Remediation:

We discussed this vulnerability and your developer team needs a unique solution for this case. Because the best practice of remediation can reflect badly on your business process.

2.5. Stored XSS

Category: A03:2021 – Injection

Severity: **High**

Vulnerability explanation:

Unrestricted file upload leads to Stored-XSS.

Endpoint: *****

In the “Create Case” page, no validation is performed on the uploaded files. In that case, a user can upload an arbitrary file to the server. Then, in the preview page of the attachment, this file will be served.

Exploitation process:

As a PoC, we have uploaded an HTML file to the server, then executed it on the victim’s browser.

1. Attacker creates a new “JSON API” bot block.

```
$cat malicious.html
<script>

window.history.pushState('page2', 'Proto'
alert("We've moved our domain to ==> hacker.com\nPlease, visit our new website and login to your account");

</script>
```

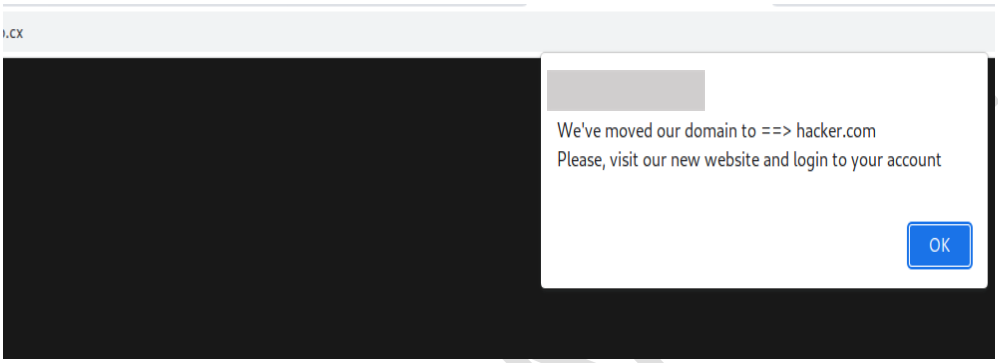
2. Uploads the file to the server, using /case/attachment endpoint.

[.....HIDDEN INFO.....]

3. Delivers the malicious URL to the victim's user. The URL is value of "*****" parameter in the response. (Above screenshot)

URL: *****

When the victim clicks to the link, the following page will be displayed.



4. With the help of 2nd line (Figure 1), the path section after the domain is deleted using the "pushState" method. This makes the URL even more realistic.

5. This behavior can lead to the following security issue:

An attacker copies ***** login page, and modifies it to send user submitted data to his own server. Then uploads the HTML file via the vulnerable endpoint, and sends it to the user, convincing them that it is the legit login page. As the domain (*****) belongs to *****, the victim has a high chance of being deceived

Note: Since the session token is stored under ***** domain's Local Storage, this vulnerable endpoint does not have access to it, so, it's not possible to escalate this to an "account takeover".

Remediation:

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

2.6. IDOR-Privilege Escalation

Category: A01:2021 – BROKEN ACCESS CONTROL

Severity: **High**

Vulnerability explanation:

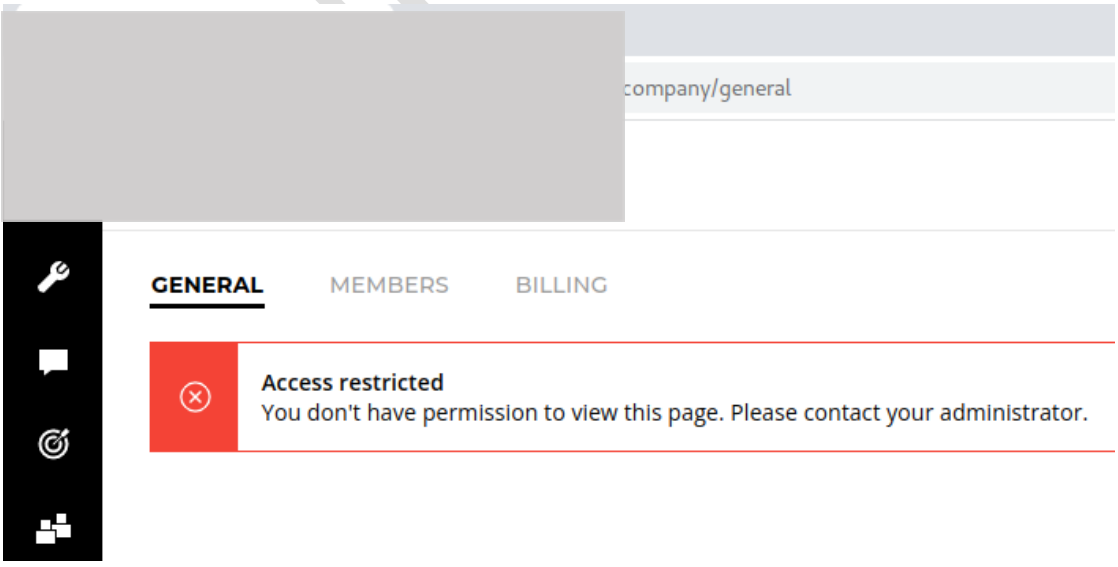
An unprivileged user (Company.MEMBER) can change the name and avatar of the company. (even if it doesn't belong to the company).

Exploitation process:

1. Attacker gets ***** of the target company. It can be retrieved with the help of "*****" endpoint.

[.....HIDDEN INFO.....]

Note: As the current user is "Company.MEMBER" on the company called "SecureComp" (*****), he cannot modify this company's settings.



2. Then, attacker modifies settings of the company where he has

"Company.ADMIN" permission(*****), and intercept the request with proxy software

[.....HIDDEN INFO.....]

3. As seen in the above screenshot, company ID is sent via request. However, it is not checked whether the user who sent the request has enough privileges on that company. So, user can replace his own company's ID(*****) with the target company's ID(*****). Modified request is shown in the following image.

   [.....HIDDEN INFO.....]

4. After sending the request, the target company's name is changed. To confirm this, we can use " *****" endpoint.

   [.....HIDDEN INFO.....]

Source code analysis:

Source code of the vulnerable endpoint is shown in the following screenshot:

```
@bp.route('/v2/company/<company_id>', methods = ['PUT'])
@login_required()
@req_validator(
    json = {
        'company_name': dict(type = 'string', required = False, nullable = True),
        'avatar_id': dict(type = 'string', required = False, is_uuid = True, nullable = True),
        'beta_program': dict(type = 'boolean', required = False)
    }
)
@resp_validator2(BooleanSchema)
def update_company_v2(company_id):
    request.logged_user.can_do(ProfileOperation.CREATE_COMPANY)

    company_name = request.j.get('company_name', None)
    avatar_id = request.j.get('avatar_id', None)
    beta_program = request.j.get('beta_program', None)

    _update_company(company_id, company_name, avatar_id, beta_program)

    return {'success': True}
```

Privilege check process is implemented in the line 101 (*****).

The CREATE_COMPANY method is defined in the ***** file.

```
1178
1179    @staticmethod
1180    def CREATE_COMPANY(user):
1181        # if user.is_super_admin():
1182        #     return True
1183
1184        # TODO: Determine the new permission rule of CREATE_COMPANY.
1185        return True
1186
```

Unlike other methods, no verification is performed here. This method will return "True" in all cases.

Remediation:

The only real solution to this issue is to implement access control. The user needs to be authorized for the requested information before the server provides it.

2.7. Code Review

Category: Insecure Randomness

Severity: Low

Vulnerability explanation:

Standard pseudorandom number generators cannot withstand cryptographic attacks. A PRNG is an algorithm used to produce random-looking numbers with certain desirable statistical properties. In order for a PRNG to be cryptographically secure, it must be resistant to prediction.

Code Block:

- *****, line 480

```
477  params = action['params']
478
479  if response := ctx.get_value_by_lang(params['response']):
480  reply = random.choice(response)
481  else:
482  reply = ""
```

- *****, line 119

```
116  # start sending messages every second ..
117  ind = 0
118  while True:
119  text = random.choice(INPUT_DATABASE)
120
121  # Wait for the websocket server to send us the cid
122  await self.waiting_for_cid.wait()
```

- *****, line 44

```
43 def _verify_challenge(url: str, verify_token: str):
44  challenge = randint(100000, 999999)
45  data = {"type": "verify", "verify_token": verify_token, "challenge": challenge}
46  # TODO: implement try/catch
47  r = requests.post(url, json = data)
```

Remediation:

We recommend using the secrets module's PRNG as follows:

https://docs.python.org/dev/library/secrets.html#secrets.SystemRandom

2.8. Informational: Advices

1. The old version of the software: Grafana v7.1.1

   *****

   The older version of Grafana has multiple security vulnerabilities. We advise updating your system to an up-to-date version.

2. ***** : Ngrok service

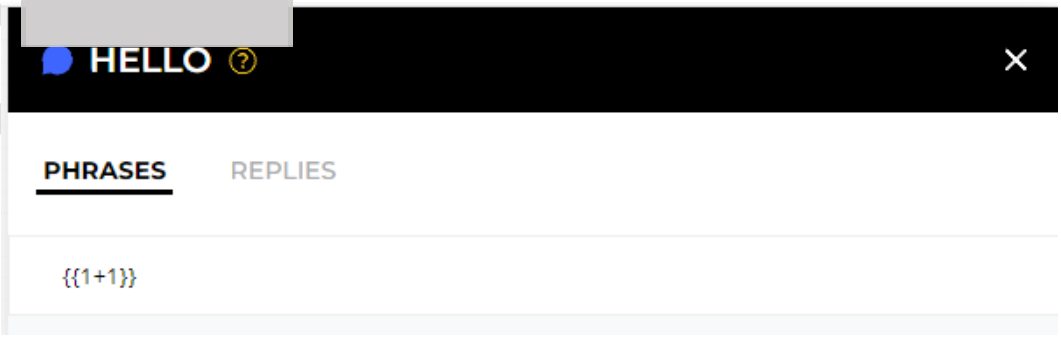
   In the future, please notice this service. Because developers can publish sensitive services or API's with the Ngrok application.

3. Dangerous allowlist policy

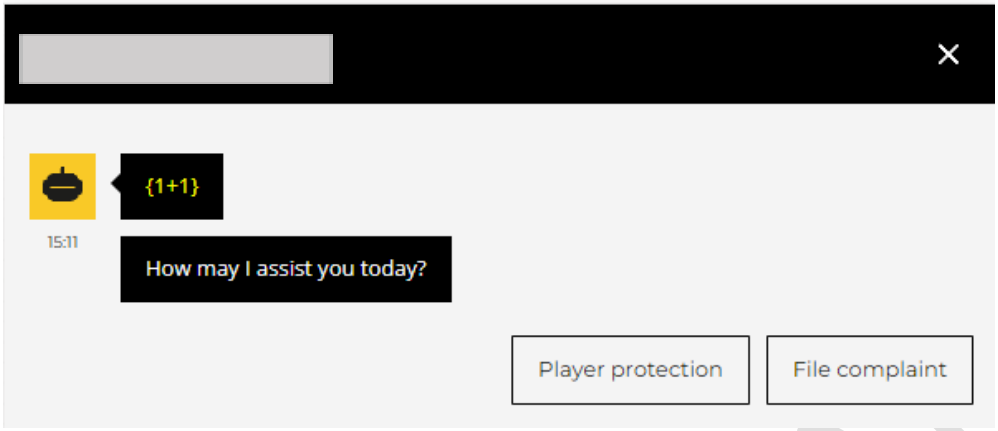
   The ***** application uses the Flask Jinja template engine. We notice when we started to build a bot, we can inject mathematical operations on the template engine but developers use the ***** library for security. Therefore we couldn't inject a malicious payload (for example: A remote code execution payload) into the system.

```
from jinja2.nodes import _cmpop_to_func
from jinja2.sandbox import SandboxedEnvironment
from jinja2.visitor import NodeVisitor
```

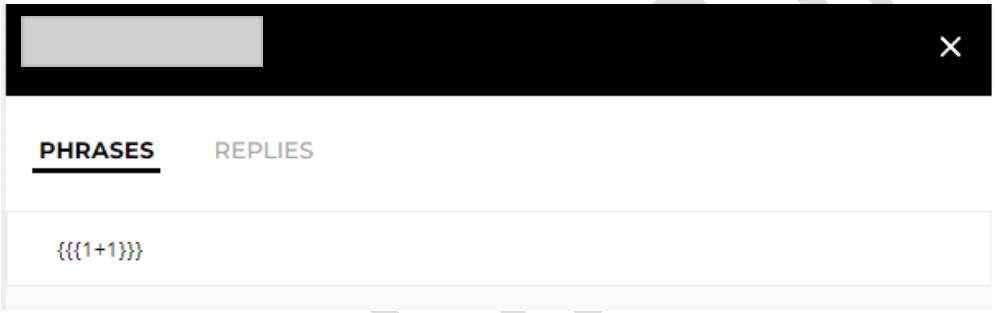
   But we could bypass some restrictions like a “double bracket”:

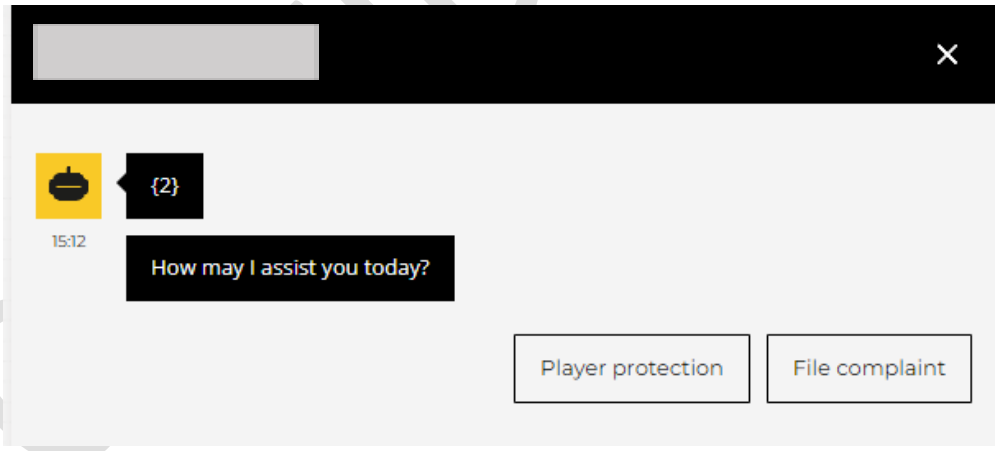


   Application deleted our double bracket payload



An attacker can bypass that restriction using a “triple bracket”.





In the future, you can use complex restrictions on important functionality.

2.9. Risk Rating

During the test, our team found high and critical-level security vulnerabilities. "*****" application has passed penetration testing check with a 5/10 score. The overall risk identified to ***** is **Medium**.

Appendix C: About ESKA

We are the providers of external and internal network penetration services, which could help reveal vulnerabilities before “real” hackers do. All this in a controlled and secure framework and without exploiting the security gaps found, so you could see the holes in your cybersecurity and fill them with the modern cybersecurity tools – no one unwanted could ever get in.

A week rarely goes by without reports of attacks on sensitive systems. It results in financial damage, and the reputation and trust of customers and partners’ crumble.

To protect yourself against attacks, adequate countermeasures must be taken at different levels. Well-trained employees and processes that also take IT security into account are essential for effective protection. However, above all, the security check through a penetration test by an independent third party is an effective means.

So, what is exactly a penetration test? A penetration test is an authorized, planned, and a simulated cyber-attack on a company or a public sector institution. The aim is to identify and eliminate previously unknown points of attack before hackers can use them to steal intellectual property or other sensitive data or otherwise damage an organization.

During the penetration test, trained testers attempt to attack your IT systems using the methods of criminal hackers to determine the vulnerability of systems, after which appropriate protective measures can be taken.

There are two types of businesses:

- those that have already been hacked
- those that will be hacked once

To effectively protect yourself against hacker attacks, penetration tests can give a clear picture of the system’s security situation.

If you would like to discuss your penetration testing needs, please contact us at office@eska.global