Attacked team – Team 4

Domain name: https://neu-csye6225-spring2017-team-4.me/

Attack vector 1:

WMAP Kali Linux vulnerability testing tool

WMAP is a feature-rich web application vulnerability scanner that was originally created from a tool named SQLMap. This tool is integrated with Metasploit and allows us to conduct **web application scanning** from within the Metasploit Framework

```
msf > vulns
[*] Time: 2012-01-16 20:58:49 UTC Vuln: host=172.16.2.207 port=80 proto=tcp name=auxiliary/scanner/http/options refs=CVE-2

msf >
```

With this tool we found that opponent's team was vulnerable for JavaScript based XSS attack (cross site scripting).

Risk factors:
Web applications are always becoming more and more complex. For many, trying to constantly push out new features as quickly as possible is causing security to be put at the back-burner of the development process. This could occur for a number of reasons including small development budgets, tight deadlines, and general unawareness of best security practices to name a few.

The result of not taking security seriously when developing software leads to vulnerabilities which put not only the organizations systems, but also potentially its reputation and customer's personal data at risk. This is largely the case when it comes to web-application vulnerabilities. There are many types of these vulnerabilities, but, for the sake of this article, we will cover a particular type of input validation vulnerability called Cross-Site Scripting (XSS) attacks.

Input validation vulnerabilities occur when user-input that is submitted to a website is not sanitized. There are many types of input validation attacks including (but not limited to) SQL Injection, XSS, File Inclusion (both Local and Remote), and system command injection (or Shell injection). These attacks are heavily used because of how many applications are still vulnerable as well as the ease of exploitation and the immediate impact that exploiting these vulnerabilities has.

At their most basic level, input validation vulnerabilities allow attackers to use specially crafted input to cause the web-application to function differently than how its developer intended. Whether this be the ability to inject SQL commands that are sent unsanitized to a back-end

database (SQL injection), or the ability to leverage file upload capabilities in the application to upload a PHP backdoor (Remote file inclusion), code injection attacks should be taken very seriously. But, let's focus on XSS.

Cross-Site Scripting (XSS) attacks occur when user input is not sanitized and then echoed back to the user somehow. There are three types of XSS attacks: Reflective, Stored, and DOM injection. The two that will be covered here are reflective and stored XSS attacks.

Attack vector 2:

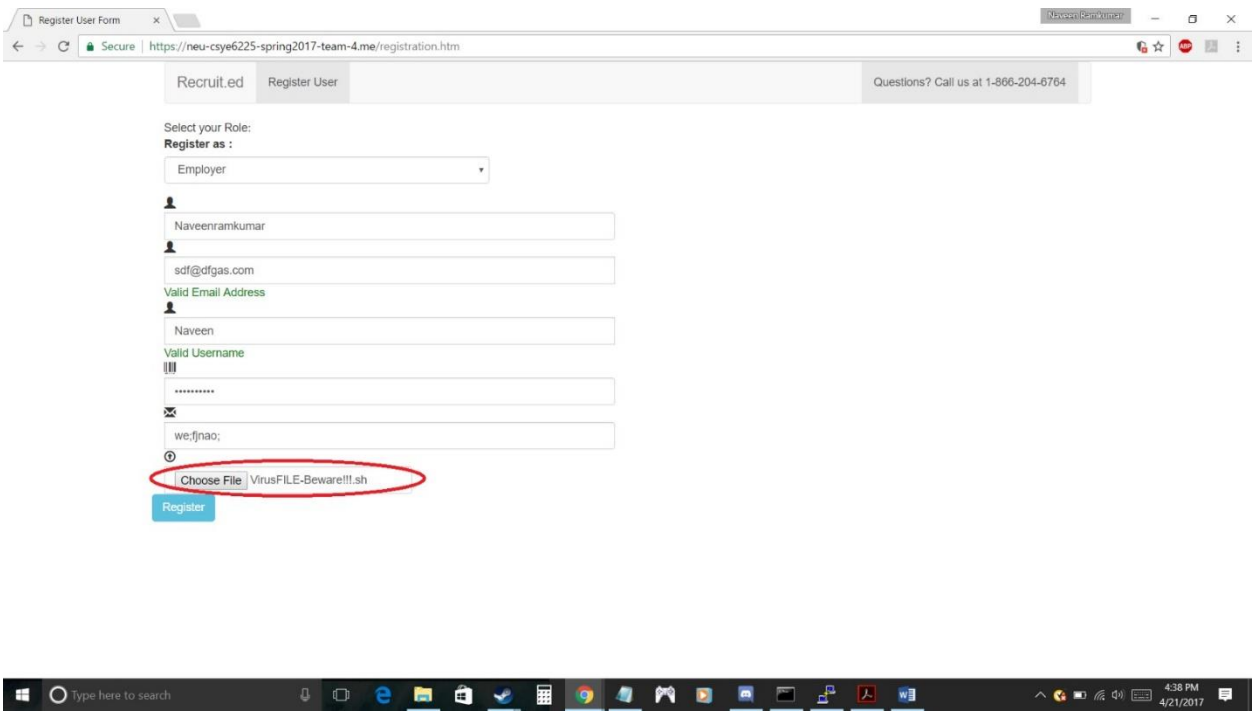
Unrestricted file upload

We observed that our opponent's web application accepts all kinds of files to be uploaded. This is one of the factor that puts the application in significant risk and further exploit it. For part 1 , we tried to upload malicious file.

Risks factors:

- The impact of this vulnerability is high, supposed code can be executed in the server context or on the client side. The likelihood of detection for the attacker is high. The prevalence is common. As a result the severity of this type of vulnerability is high.
- It is important to check a file upload module's access controls to examine the risks properly.
- Server-side attacks: The web server can be compromised by uploading and executing a web-shell which can run commands, browse system files, browse local resources, attack other servers, or exploit the local vulnerabilities, and so forth.
- Client-side attacks: Uploading malicious files can make the website vulnerable to client-side attacks such as XSS or Cross-site Content Hijacking.
- Uploaded files can be abused to exploit other vulnerable sections of an application when a file on the same or a trusted server is needed (can again lead to client-side or server-side attacks)
- Uploaded files might trigger vulnerabilities in broken libraries/applications on the client side (e.g. iPhone MobileSafari LibTIFF Buffer Overflow).
- Uploaded files might trigger vulnerabilities in broken libraries/applications on the server side (e.g. ImageMagick flaw that called ImageTragick!).
- Uploaded files might trigger vulnerabilities in broken real-time monitoring tools (e.g. Symantec antivirus exploit by unpacking a RAR file)
- A malicious file such as a Unix shell script, a windows virus, an Excel file with a dangerous formula, or a reverse shell can be uploaded on the server in order to execute code by an administrator or webmaster later -- on the victim's machine.
- An attacker might be able to put a phishing page into the website or deface the website.
- The file storage server might be abused to host troublesome files including malwares, illegal software, or adult contents. Uploaded files might also contain malwares' command and control data, violence and harassment messages, or steganographic data that can be used by criminal organisations.
- Uploaded sensitive files might be accessible by unauthorised people.

- File uploaders may disclose internal information such as server internal paths in their error messages.

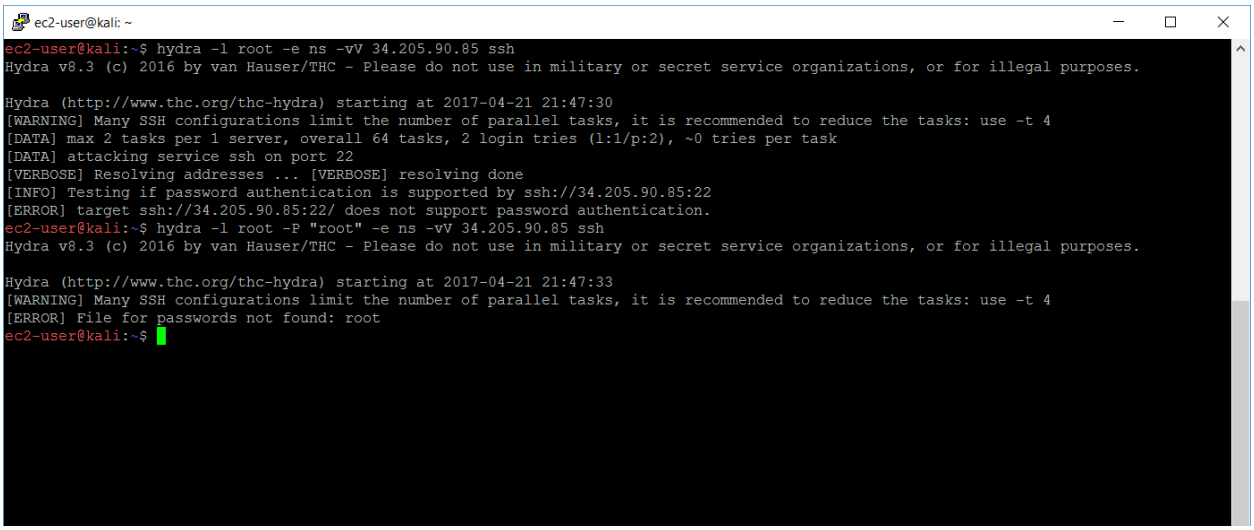


Attack Vector 3:
Additional open ports

Nmap (Network Mapper) is a security scanner, originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich),[2] used to discover hosts and services on a computer network, thus building a "map" of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host(s) and then analyzes the responses.

The software provides a number of features for probing computer networks, including host discovery and service and operating-system detection. These features are extensible by scripts that provide more advanced service detection,[3] vulnerability detection,[3] and other features. Nmap can adapt to network conditions including latency and congestion during a scan. The Nmap user community continues to develop and refine the tool.

Nmap found two open ports: **80** and **22**, so we know that the server has both HTTP and SSH services. At this point, we tried to use Hydra to crack the root password on SSH.


ec2-user@kali: ~
ec2-user@kali:~$ hydra -l root -e ns -vV 34.205.90.85 ssh
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.
Hydra (http://www.thc.org/thc-hydra) starting at 2017-04-21 21:47:30
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 2 tasks per 1 server, overall 64 tasks, 2 login tries (l:1/p:2), ~0 tries per task
[DATA] attacking service ssh on port 22
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://34.205.90.85:22
[ERROR] target ssh://34.205.90.85:22/ does not support password authentication.
ec2-user@kali:~$ hydra -l root -P "root" -e ns -vV 34.205.90.85 ssh
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.
Hydra (http://www.thc.org/thc-hydra) starting at 2017-04-21 21:47:33
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[ERROR] File for passwords not found: root
ec2-user@kali:~$