Nethemba

all about security

Web Application Attacks in practice

or how the modern real application attack looks like

Nethemba s.r.o., Michalská 3, 821 03 Bratislava, +421 905 400542, +420 910 800 955, info@nethemba.com,

Contents

- Achieving anonymity
- Username enumeration, password bruteforcing
- XXE/XML Injection
- 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)
- Use Tor/I2P anonymization networks
- Hack any Internet vulnerable server (there are millions)
- Use anonymous shell accounts (freeshell.eu)

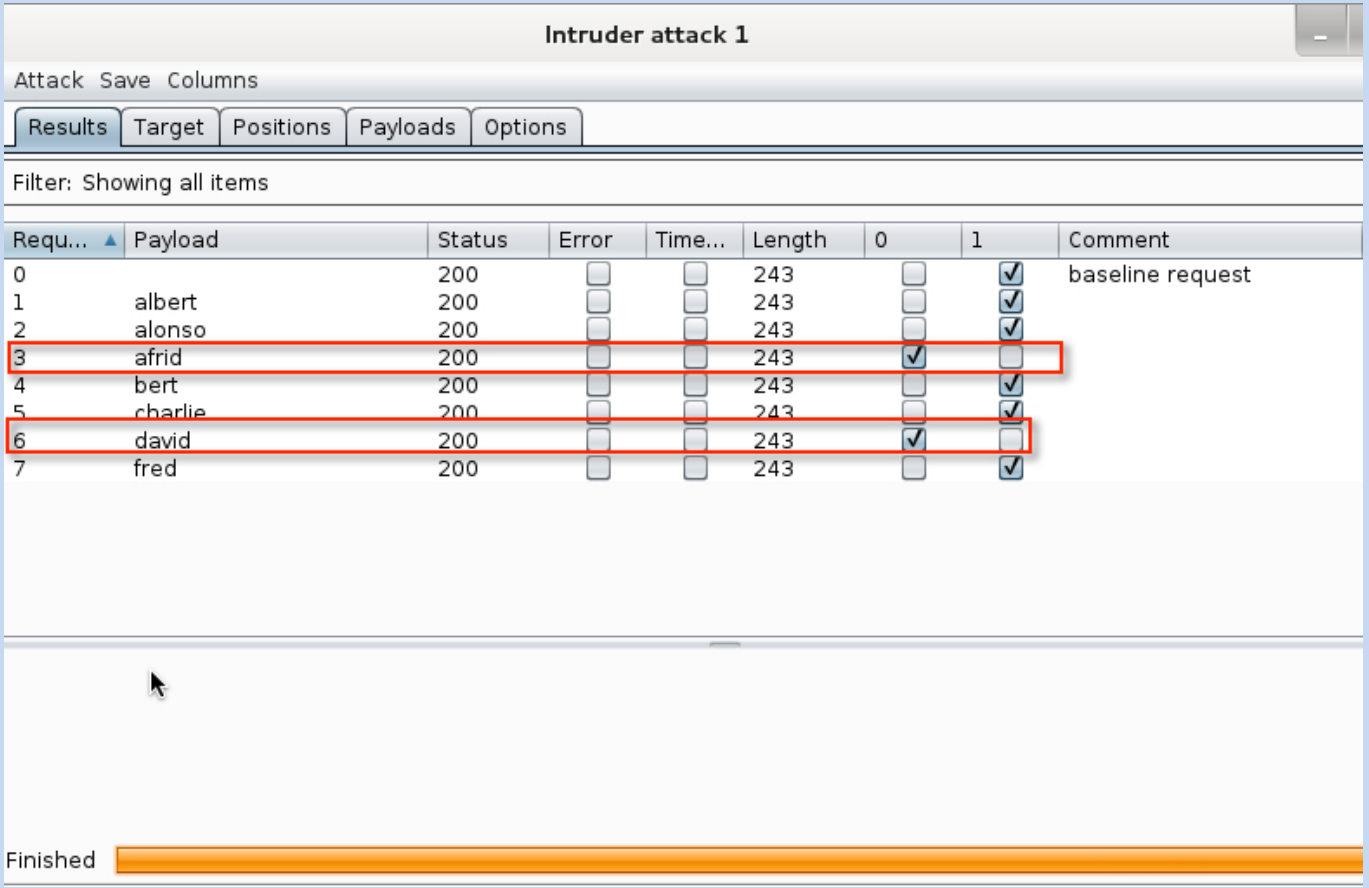
The attacker has to be aware of

- Sanitizing its “browser footprint”
- DNS leaks (using TOR internal DNS resolver)
- Traffic analysis (**T**raffic **C**onfirmation 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)
- **Note:** We have no credentials, no logins, no passwords - let's find existing ones using the wordlist of commonly used English usernames

Username enumeration II



Password enumeration

- We have valid logins, so can brute-force valid passwords now
- The problem is the account is blocked for a define time period after 3 incorrect passwords, so we will use in-breadth search instead of in-depth search (most applications are unable to detect it) – login1, password1, login2, password2, login3, password3, etc.)

We are 'in', looking for vulnerabilities

- We have revealed the valid combination of login and password and are 'in'
- Unfortunately no SQL/LDAP injections vulnerabilities, nor classical XSS vulnerabilities have been revealed
- But it seems the Java server allows XXE/XML injection

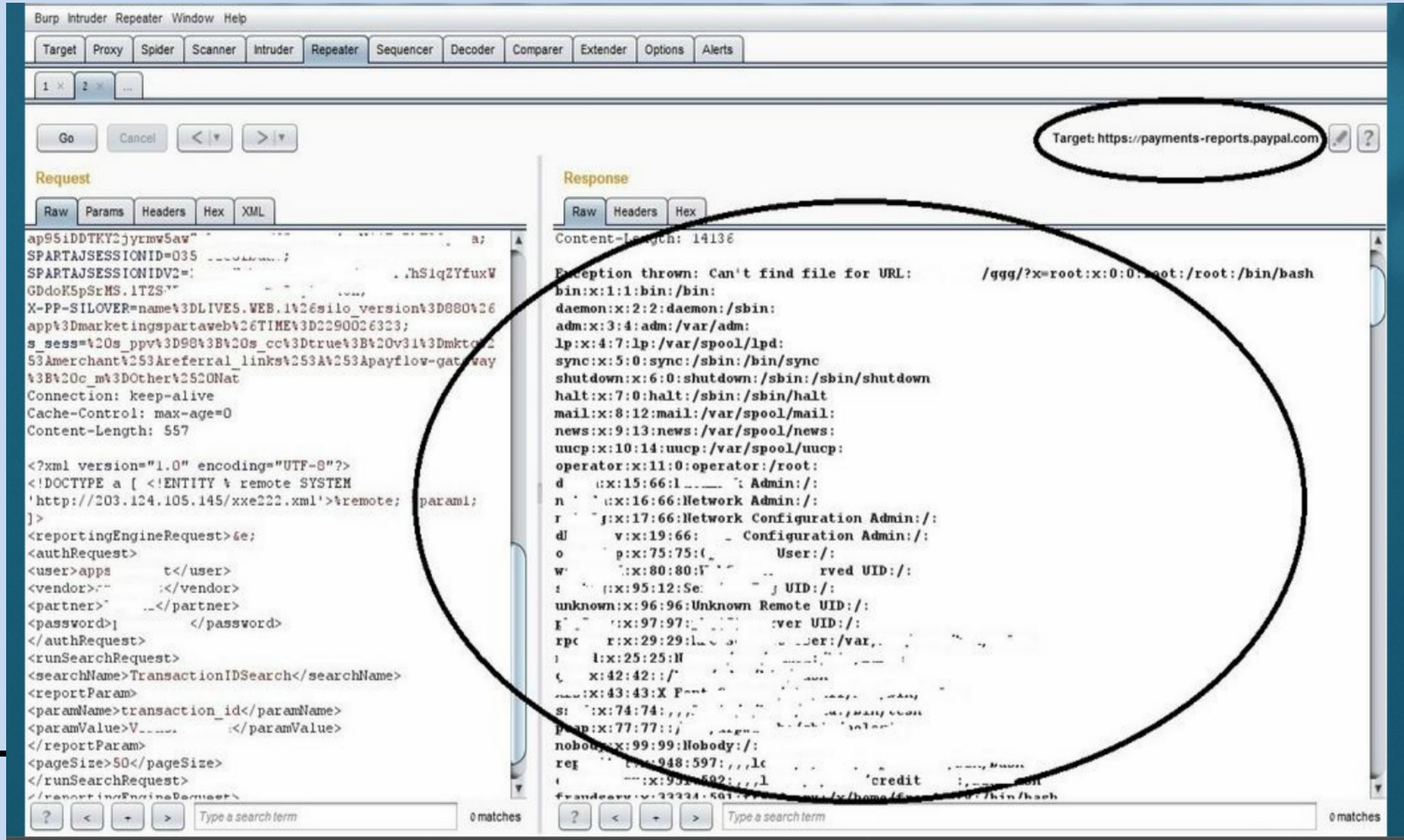
XXE/XML Injection

- 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>
```

Nethemba

all about security



XXE/XML Injection

Potential Impact:

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

The attacker's interests

The attacker is **mainly interested in**:

- Any sensitive information (credit card numbers, 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)
- Offline or online rainbow tables (for all hashes which do not use salt including DES, LM, NTLM, MD5, SHA1, SHA256, SHA512, ...)

Exploiting of local system

- Possibility to use own SQL statements (**execute privileges** or **reading/writing local files** or **run own system commands**) or make TCP/Unix sockets (SSRF) almost always leads to gaining the local system user access (apache, www-data, webuser, ..)
- Consequently, exploiting the kernel is obviously easy if the system is unpatched for few months (there are many public available local root exploits for recent Linux kernels)

Nethemba

all about security

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 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, ...)

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, or sold as a part of botnet on the hacker's blackmarkets

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

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/php5/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