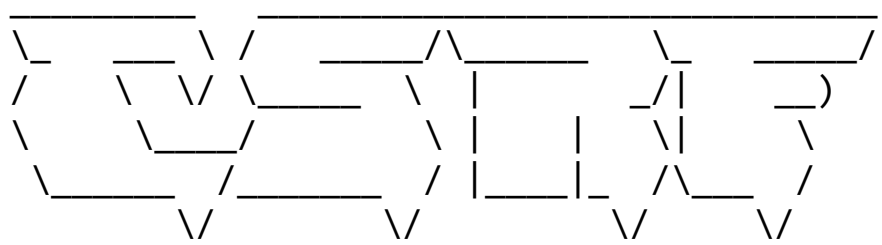
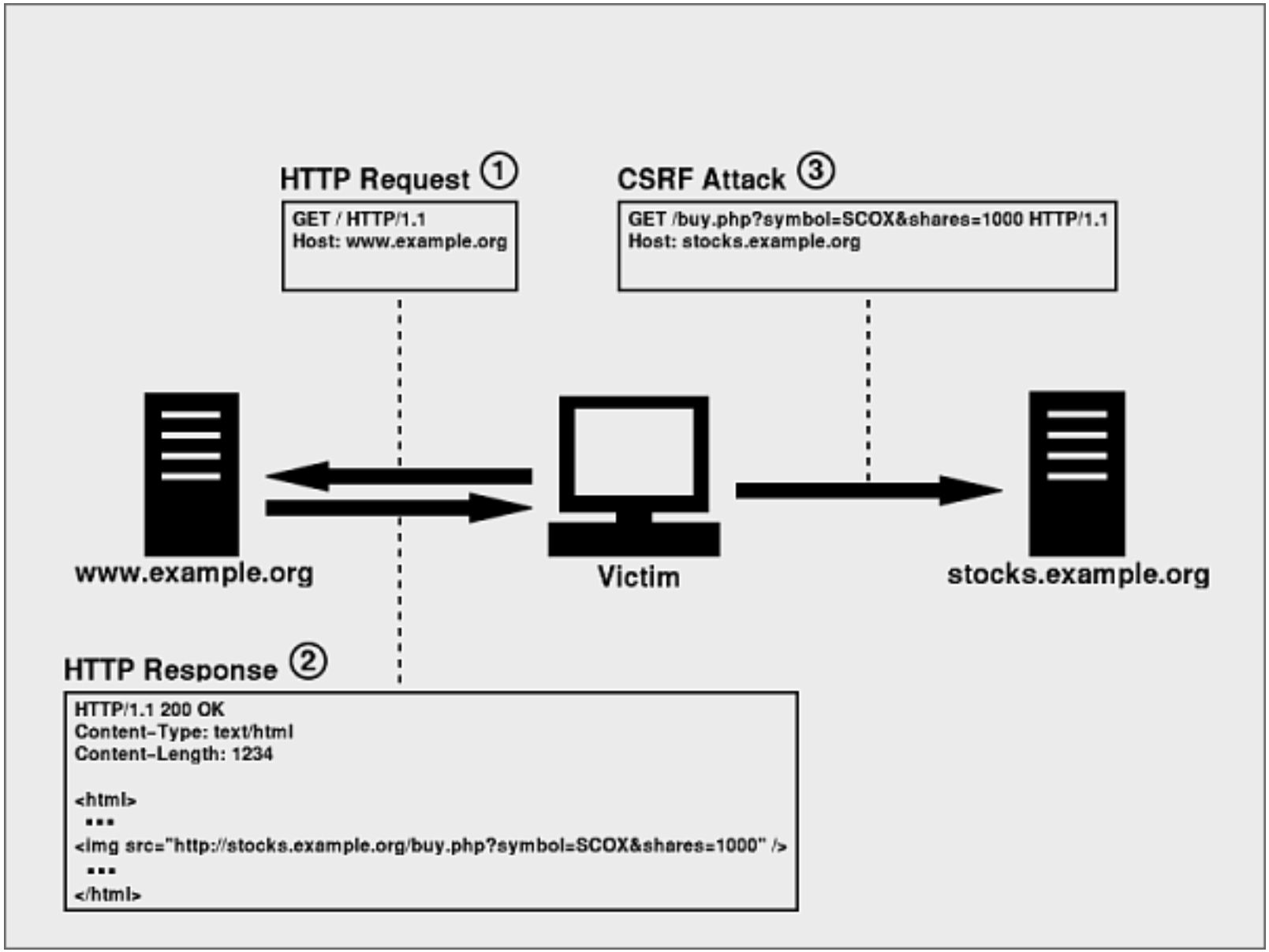


What is it?

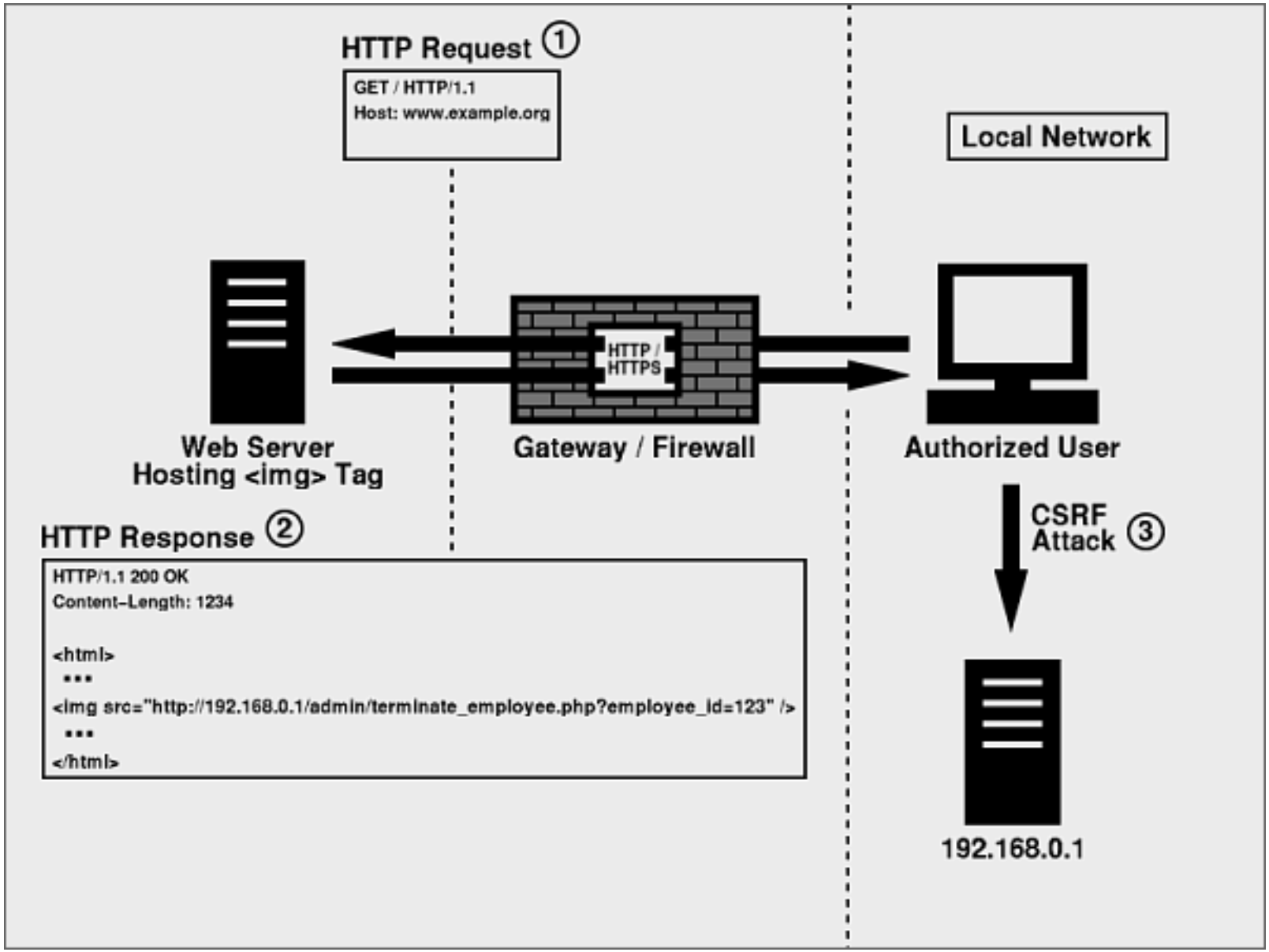
- Cross-Site Request Forgery
- An attack that tricks the victim into loading a page that contains a malicious request
- The attacker can make the victim perform actions that they didn't intend to, such as logout, purchase item, change account information, retrieve account information, or any other function provided by the vulnerable website.

How does it work?

- Attacker tricks victim into browsing to their site
- Malicious site makes a request in the background
- Browsers will automatically include with such requests any credentials associated with the site, such as the user's session cookie, basic auth credentials, IP address, or Windows domain credentials
- Therefore, if the user is currently authenticated to the site, the site will have no way to distinguish this from a legitimate user request.


HTTP Request ①
GET / HTTP/1.1
Host: www.example.org
CSRF Attack ③
GET /buy.php?symbol=SCOX&shares=1000 HTTP/1.1
Host: stocks.example.org
www.example.org
Victim
stocks.example.org
HTTP Response ②
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1234
<html>
...

...
</html>


HTTP Request ①
GET / HTTP/1.1
Host: www.example.org
Local Network
HTTP / HTTPS
Web Server Hosting <img> Tag
Gateway / Firewall
Authorized User
CSRF Attack ③
HTTP Response ②
HTTP/1.1 200 OK
Content-Length: 1234
<html>

</html>
192.168.0.1

Code example

- A bank executes a trade using the following request:

```
GET http://bank.com/transfer.do?acct=BOB&amount=100 HTTP/1.1
```

- An attacker hides the following in their website:

```
<img src="http://bank.com/transfer.do?acct=MARIA&amount=100000" width="1"
height="1" border="0">
```

Note

- CSRF does not (generally) give attacker access to body of the returned request
- It is a one-way attack (for the most part)

Ineffective Prevention

- Secret Cookie
  - Why doesn’t this work?
- Only accept POST requests
  - Why doesn’t this work?
- URL rewriting
  - Why doesn’t this work?
- Checking referrer header
  - Why doesn’t this work?

Effective Prevention

- Synchronizer Token Pattern
  - Send a token with every form/page and re-validate on form submission
- Double Submit Cookies
  - Cookie is set that matches a secret value in the form
  - Cookie value and form value are matched
- Challenge-Response
  - CAPTCHA
  - Re-authentication
  - One-time token
- No XSS Vulns!!

Resources

- https://www.owasp.org/index.php/Testing_for_CSRF_(OWASP-SM-005)
- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

Demos

- Effective CSRF prevention
- Ineffective CSRF prevention