XSS-Worms

Sven Vetsch
Leader OWASP Switzerland Local Chapter
sven.vetsch _at_ disenchant.ch
www.disenchant.ch

OWASP

12.February 2007

XSS-Worms

OWASP Switzerland Local Chapter
Meeting 12. February 2007

Sven Vetsch / Disenchant

About me

- Sven Vetsch
- Security Tester, Analyst and Engineer
- Dreamlab Technologies Ltd.
- Specialized on
  - Web application security
  - Social engineering

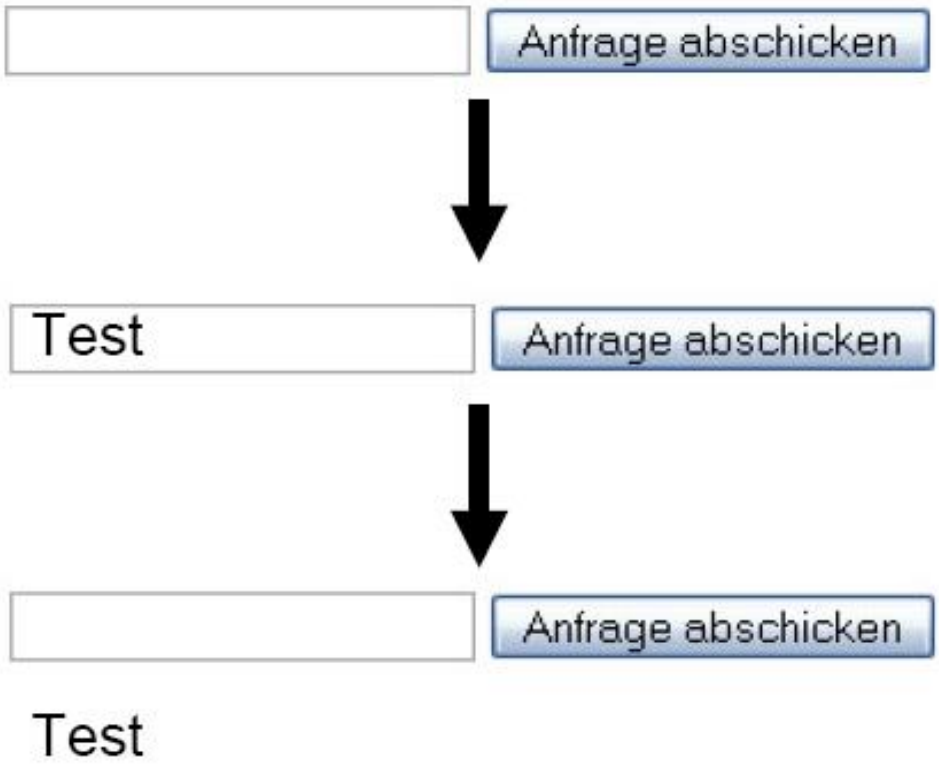
- www.disenchant.ch

Table of Content

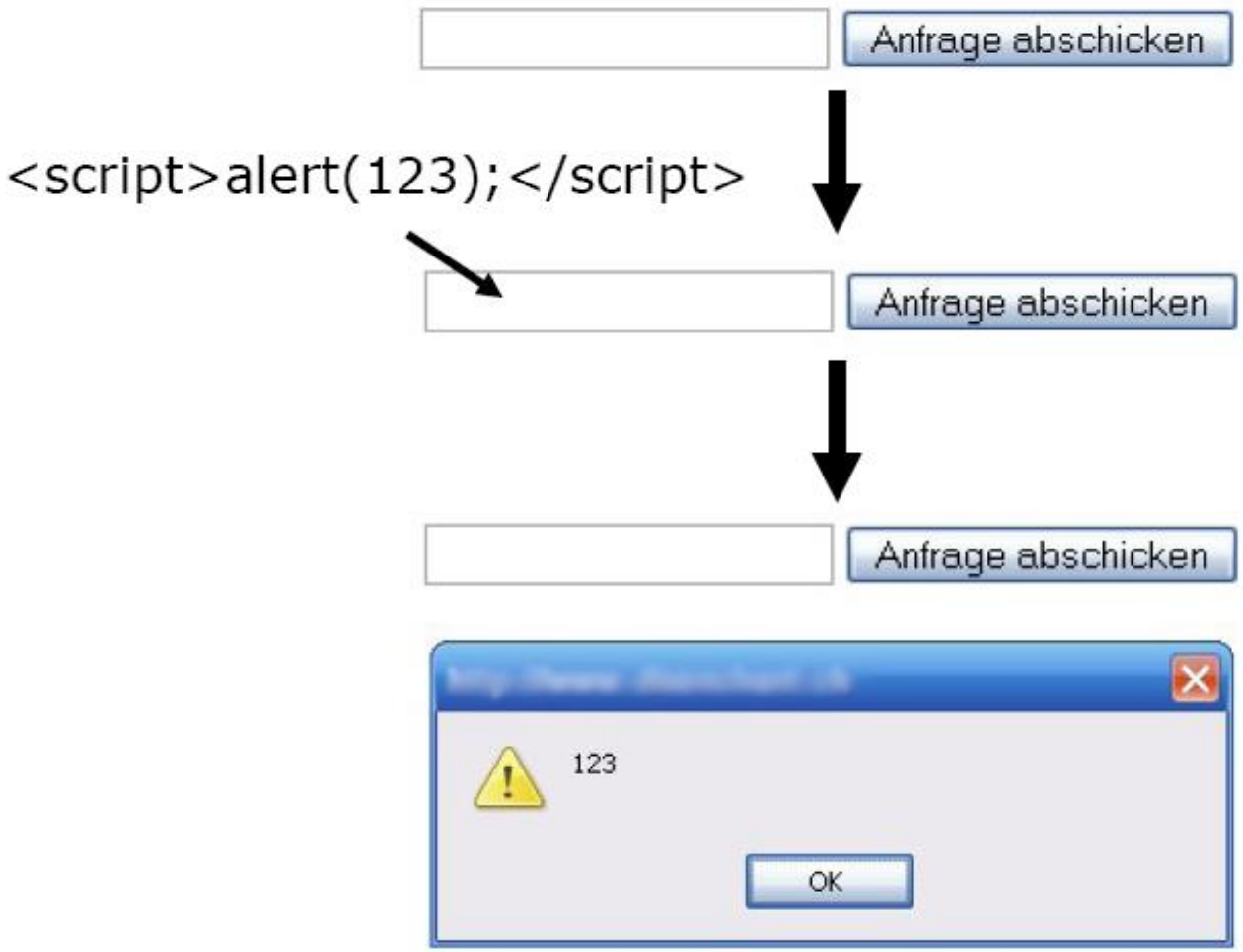
Basics on XSS

- Injection of client side script code or HTML into a third-party Website.
- Operating system independent
  - Sometimes depending on the web browser

Basics on XSS



Basics on XSS



Basics on XSS

- Most websites are/where vulnerable
  - About every second week a new hole in MySpace.com
  - My own “University Experiment”
  - Ebay, Google, Yahoo!, Microsoft, IBM

How XHRs work

- XMLHttpRequest
- Originally developed by Microsoft

How XHRs work

```
var post_data = “Username=Foo&Password=Bar”;
var req = new XMLHttpRequest();
req.open(POST, 'http://host/path/', true);
req.onreadystatechange = function () {
    if (req.readyState == 4) {
        alert(req.responseText);
    }
};
req.send(post_data);
```

How XHRs work

```
var post_data = “Username=Foo&Password=Bar”;
var req = new XMLHttpRequest();
req.open(POST, 'http://host/path/', true);
req.onreadystatechange = function () {
    if (req.readyState == 4) {
        alert(req.responseText);
    }
};
req.send(post_data);
```

How XHRs work

```
var post_data = “Username=Foo&Password=Bar”;
var req = new XMLHttpRequest();
req.open(POST, 'http://host/path/', true);
req.onreadystatechange = function () {
    if (req.readyState == 4) {
        alert(req.responseText);
    }
};
req.send(post_data);
```

How XHRs work

```
var post_data = “Username=Foo&Password=Bar”;
var req = new XMLHttpRequest();
req.open(POST, 'http://host/path/', true);
req.onreadystatechange = function () {
    if (req.readyState == 4) {
        alert(req.responseText);
    }
};
req.send(post_data);
```

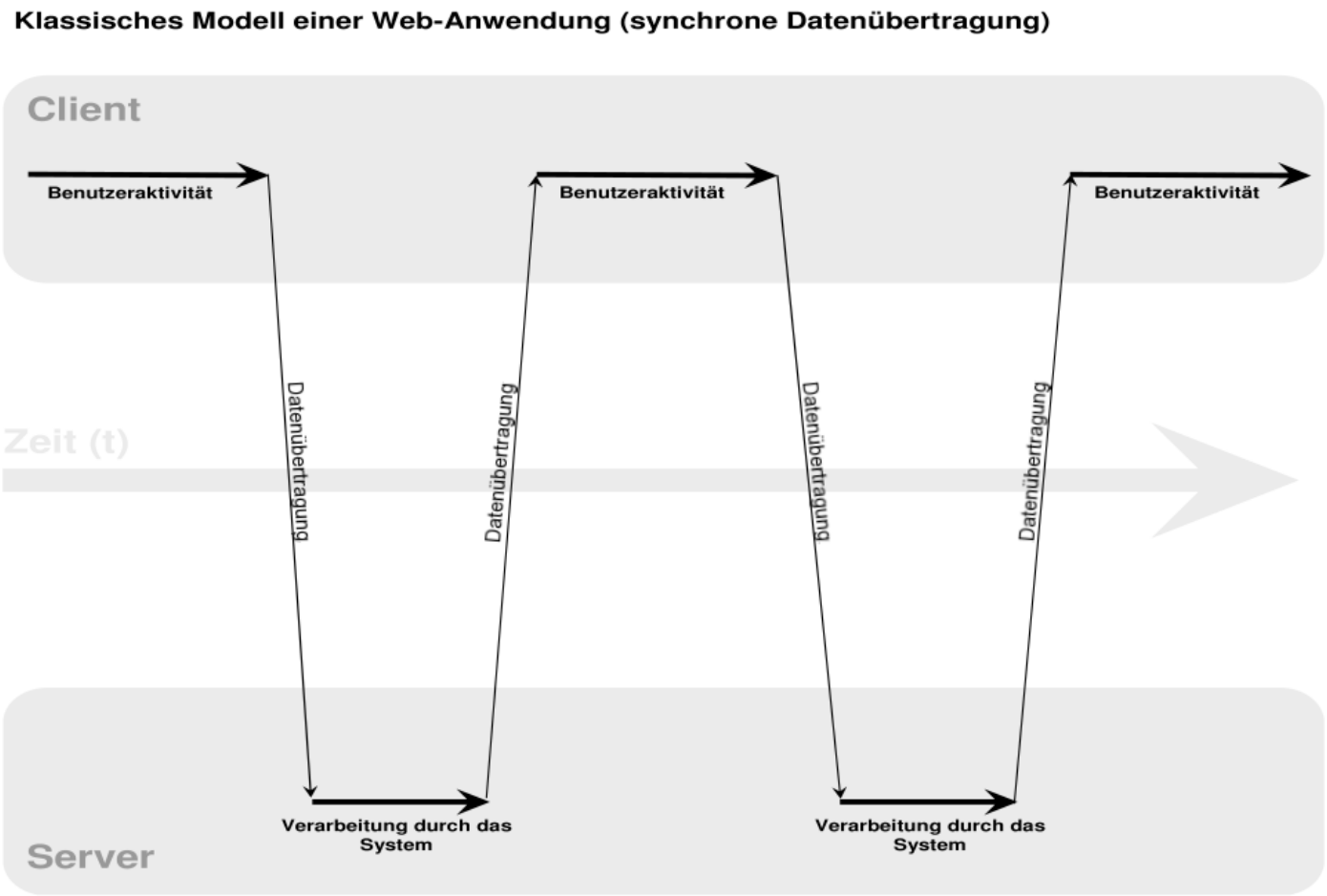
How XHRs work

```
var post_data = “Username=Foo&Password=Bar”;
var req = new XMLHttpRequest();
req.open(POST, 'http://host/path/', true);
req.onreadystatechange = function () {
    if (req.readyState == 4) {
        alert(req.responseText);
    }
};
req.send(post_data);
```

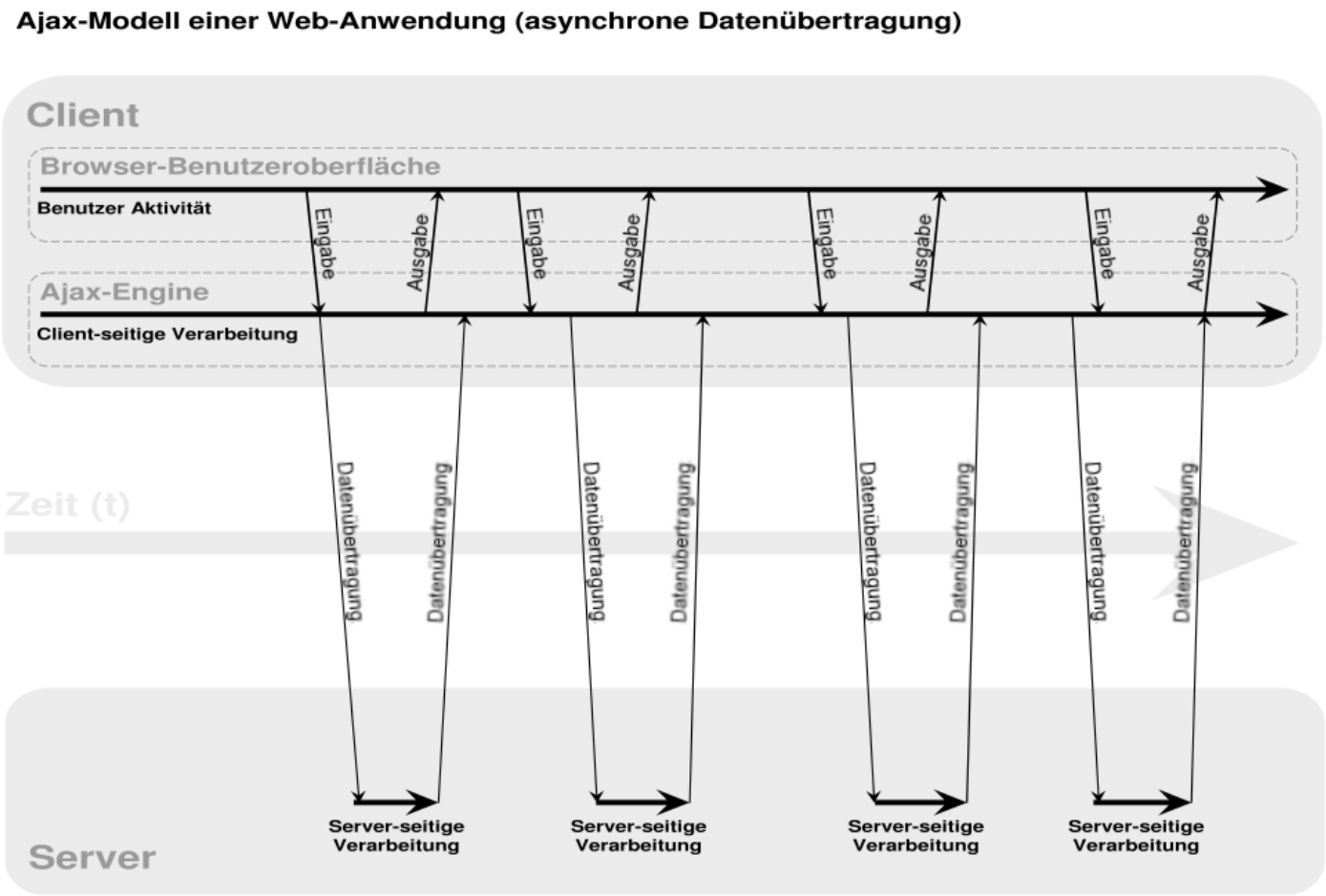
How XHRs work

```
var post_data = “Username=Foo&Password=Bar”;
var req = new XMLHttpRequest();
req.open(POST, 'http://host/path/', true);
req.onreadystatechange = function () {
    if (req.readyState == 4) {
        alert(req.responseText);
    }
};
req.send(post_data);
```

How XHRs work



How XHRs work



Famous XSS-Worms

- Samy
- 13. October 2005
- First XSS-Worm
  - Not the first webbased worm!
- Infected MySpace
- Over 1'000'000 infections in 24h
- MySpace.com had to shut down their servers for cleaning up

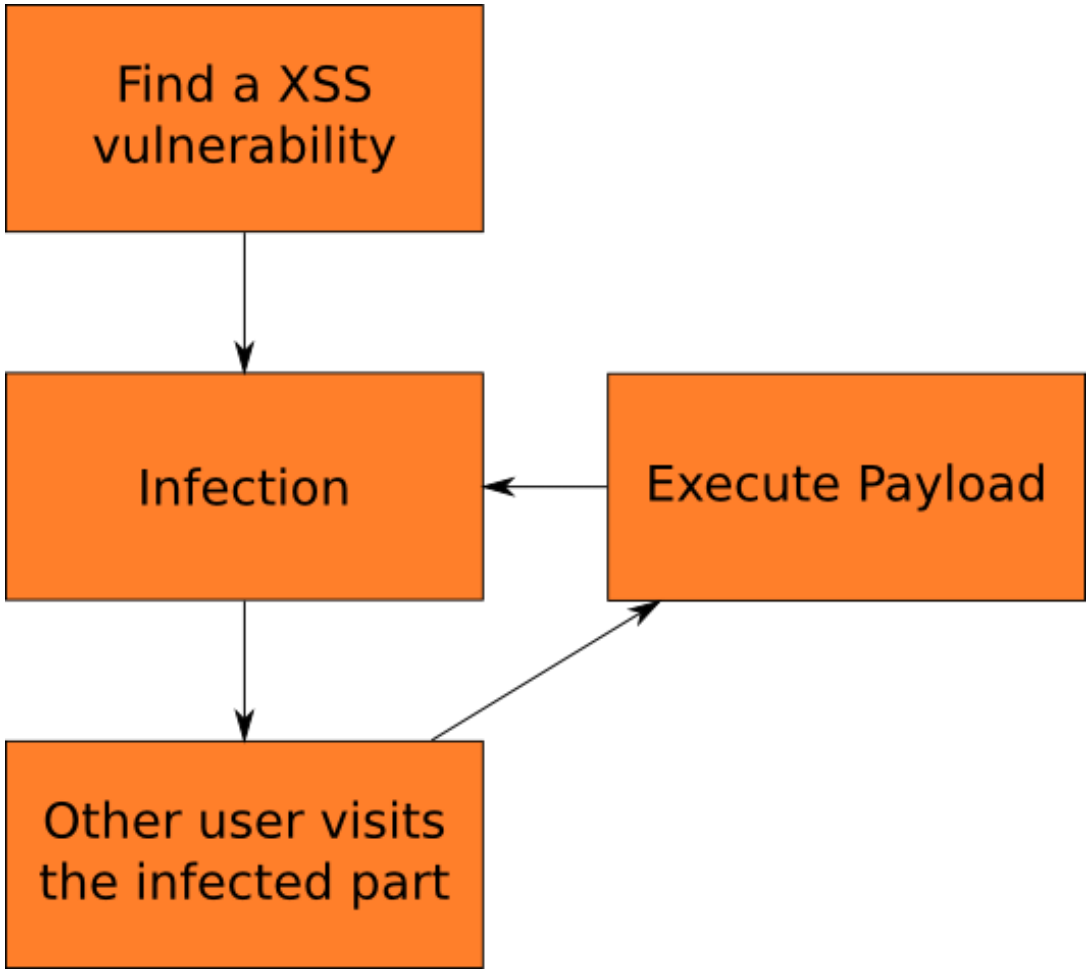
Famous XSS-Worms

- Yamanner
- Infected Yahoo! Mail
- Nobody knows how many mail addresses were stolen for spamming purposess

Anatomy of XSS-Worms

- Mix of XSS and XHR
  - Other technologies are used if necessary
- Self-propagating XSS
- Spreads very fast
  - Because of high user interaction

Anatomy of XSS-Worms



The full risk

- Problems
  - The Web 2.0 concept allows different webapplications to interact with each other
  - Users have no chance do defend such worms
  - No antivirus software helps
  - Awareness of XSS-Worms is very low
  - Increasing knowledge in Javascript

The full risk

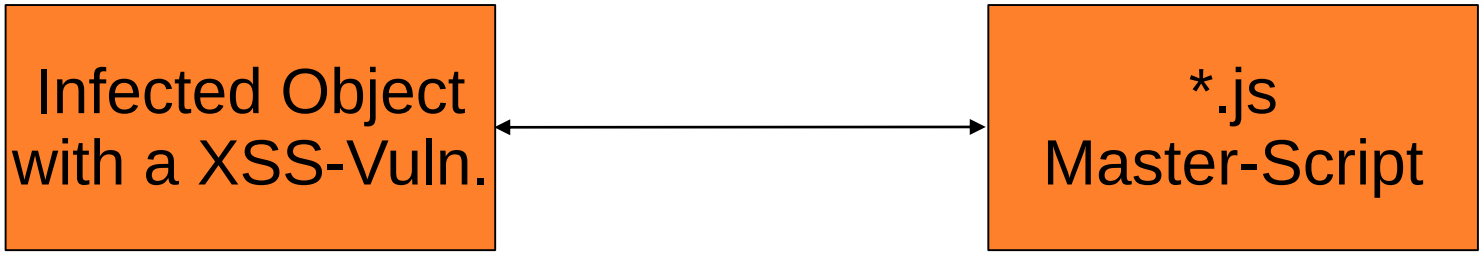
- Modify a web application over the DOM tree
- Phishing
- Hijacking accounts
- Create “real” botnets
- Spamming
- DDOS
- SEO-Hacking
- …

Webbased Dynamic Botnets

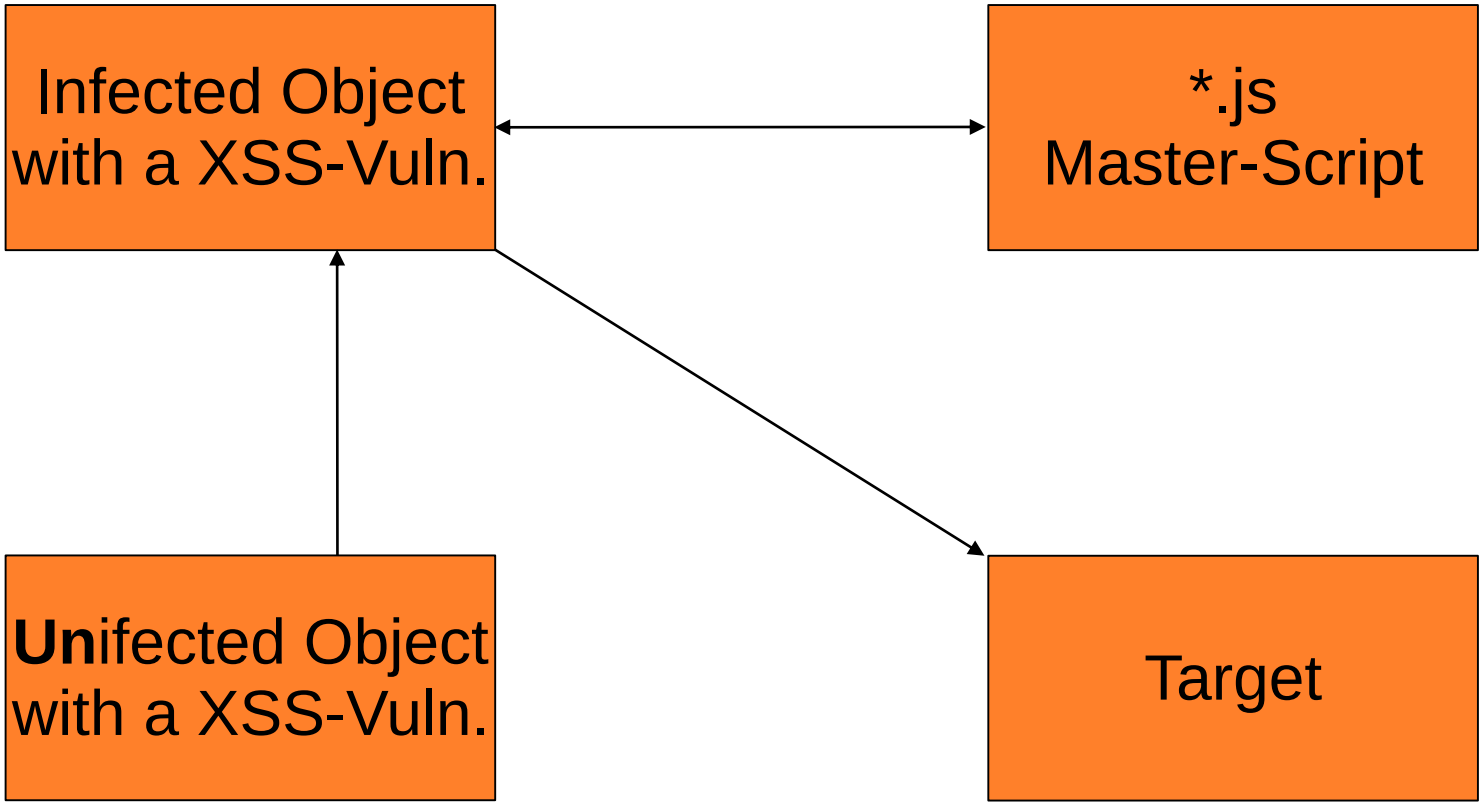
- Single Application Botnet
- Multi Application Botnet

- Master servers can also be applications with XSS vulnerabilities

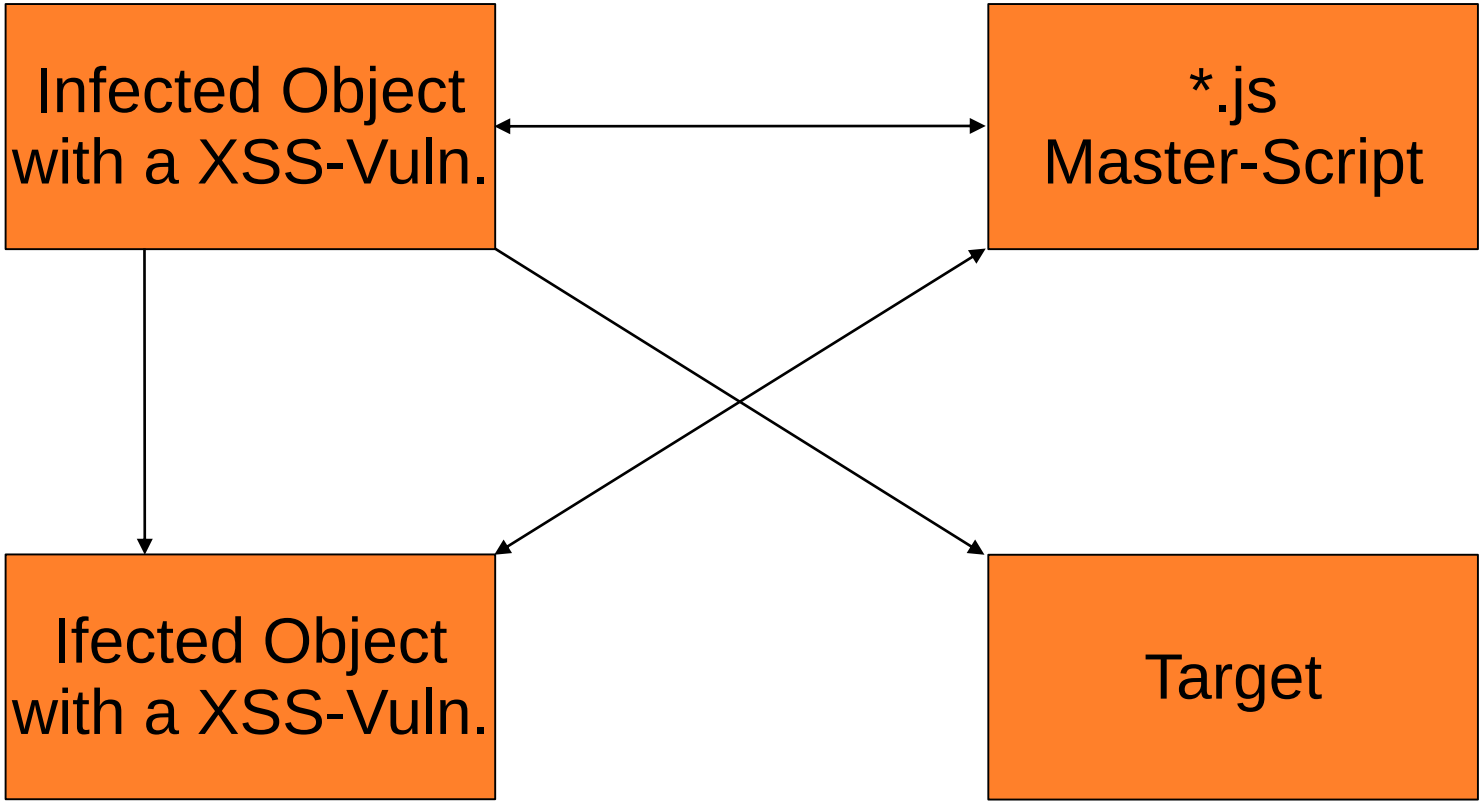
Webbased Dynamic Botnets



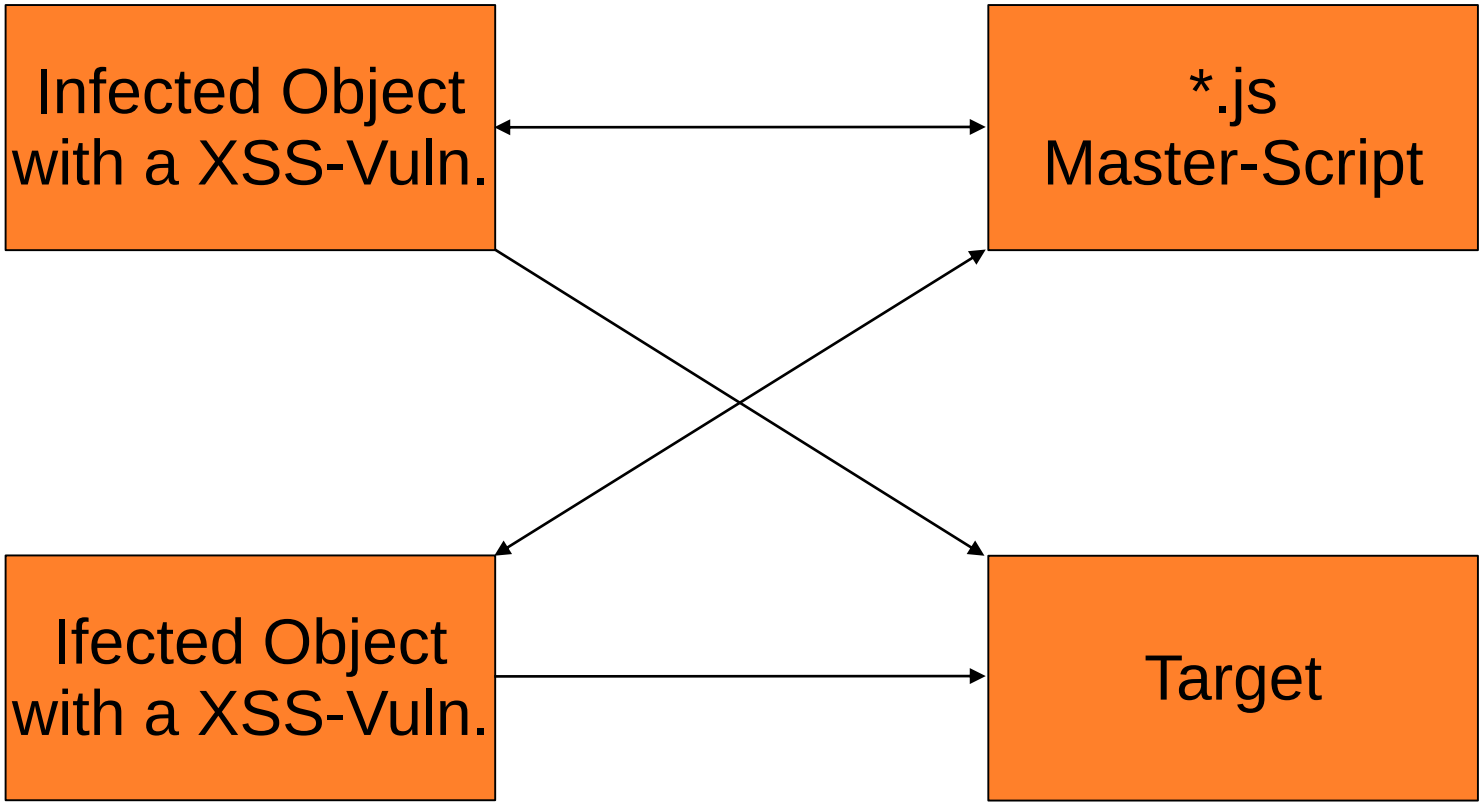
Webbased Dynamic Botnets



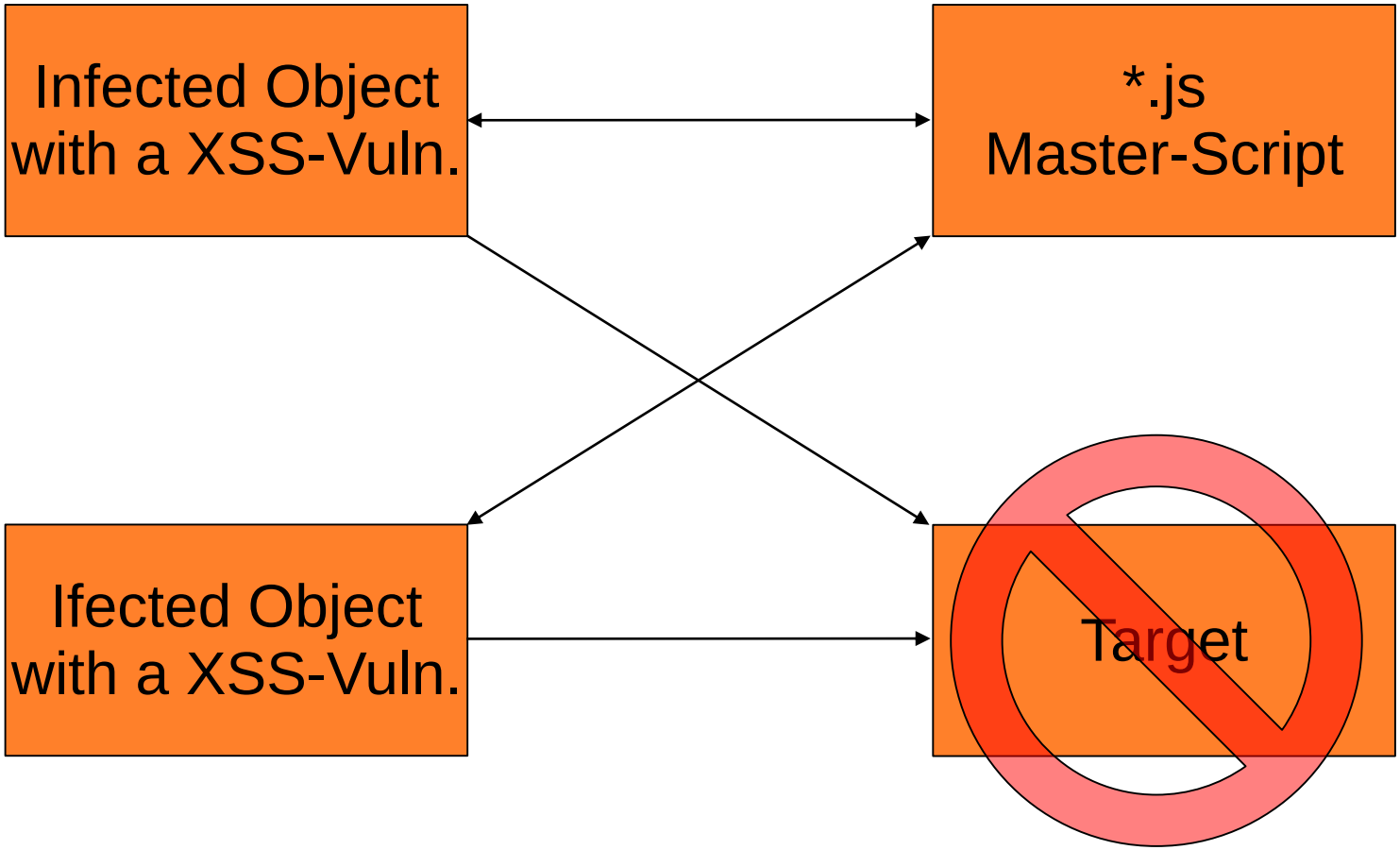
Webbased Dynamic Botnets



Webbased Dynamic Botnets



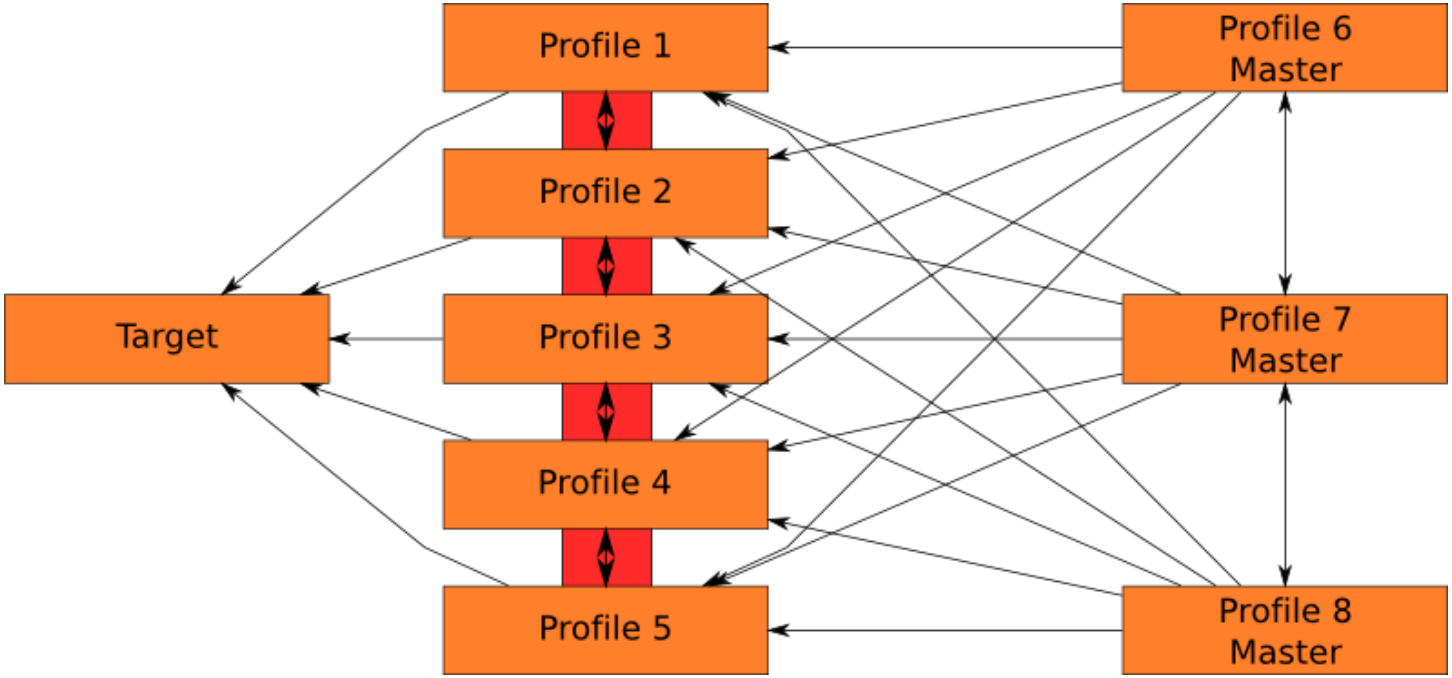
Webbased Dynamic Botnets



Webbased Dynamic Botnets

- Single Application Botnet
  - Only one webapplication is used
  - Big community sites

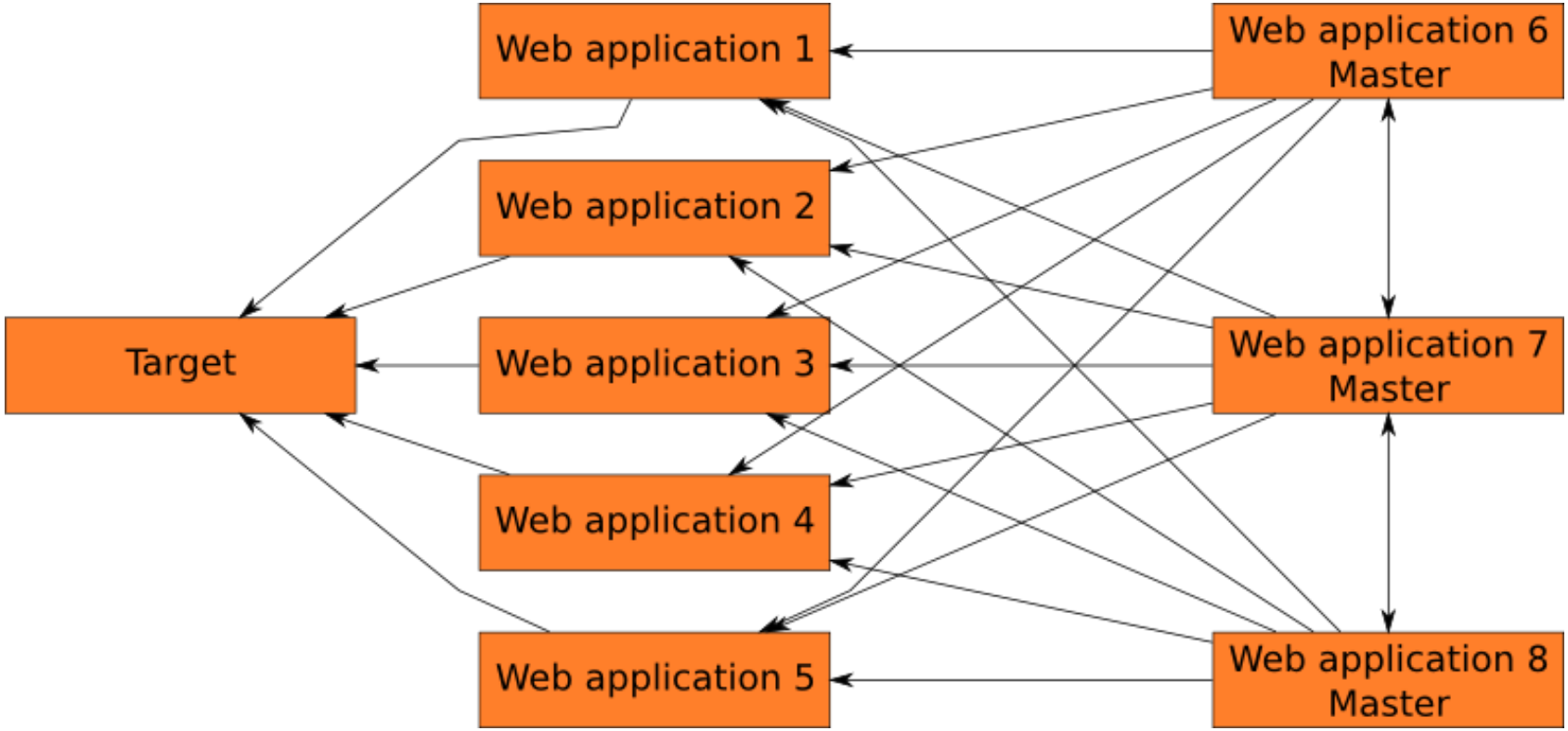
Webbased Dynamic Botnets



Webbased Dynamic Botnets

- Multi Application Botnet
  - Different webapplications of different size
  - Every infected application is independent of others
  - An attacker (normaly) has to infect every involved web application on it's own
  - From my point of view, this kind of botnet can't be stopped with any know technique today.

Webbased Dynamic Botnets



Countermeasures

- Patching and Anti-Virus
- Corporate Web Surfing Filters
- Security Socket Layer (SSL)
- Two factor authentication
- Stay away from questionable websites

Countermeasures

- Protection against XSS-Worms means protection against XSS
- Whitelisting
- Setting defaults wherever possible
- Filter EVERY input
- Santisize input wherever possible

- Users can only disable Javascript
  - Not practicable for todays web applications

Questions

Ask and I'll try to give you an answer :)

Thanks for your attention

sven.vetsch _at_ disenchant.ch