Don't Trust the Locals: Investigating Persistent Client-Side Cross-Site Scripting in the Wild

Marius Steffens, Martin Johns, Christian Rossow, and Ben Stock

Dimensions of Cross-Site Scripting

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

```
echo "Welcome ".
  $_GET["name"];
```

Reflected XSS

```
mysql_query("INSERT INTO posts ...");
// ..
$res = mysql_query("SELECT * FROM posts");
while ($row = mysql_fetch_array($res)) {
  print $res[0];
}
```

Persistent XSS

```
document.write("Welcome " +
  location.hash.slice(1));
```

DOM-based XSS

Dimensions of Cross-Site Scripting

Server | Client

Reflected

Server:

```
echo "Welcome ".
  $_GET["name"];
```

Client:

```
document.write("Welcome " +
  location.hash.slice(1));
```

Persistent

Server:

```
mysql_query("INSERT INTO posts ...");
// ..
$res = mysql_query("SELECT * FROM posts");
while ($row = mysql_fetch_array($res)) {
  print $res[0];
}
```

Client:

```
localStorage.setItem("name",
  location.hash.slice(1));
// ..
document.write("Welcome " +
  localStorage.getItem("name"));
```

From Persistence to Code Execution

- **Cookies**
  - bound to eTLD+1 or hostname
  - limited character set
    - e.g., no semicolon
    - only 4096 chars
- **Local Storage**
  - bound to an origin
  - at least 5 MB

- **HTML Markup**

```
element.innerHTML = "foobar";
```

- **JavaScript**

```
eval("x = 'foobar'");
```

- **Script source**

```
var script =
document.createElement("script");
script.src="//foobar.script.com";
document.body.appendChild(script);
```

From Persistence to Code Execution

- Cookies
  - bound to eTLD+1 or hostname
  - limited character set
    - e.g., no semicolon
    - only 4096 chars
- Local Storage
  - bound to an origin
  - at least 5 MB

- HTML Markup

```
element.innerHTML = "foobar";
```

- JavaScript

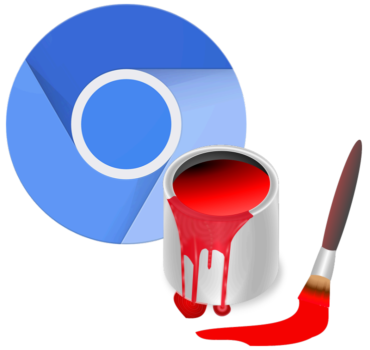
```
eval("x = 'foobar'");
```

- ce

```
c =
document.createElement("script");
script.src="//foobar.script.com";
document.body.appendChild(script);
```

How prevalent is this threat among top sites?

Collection of Flows



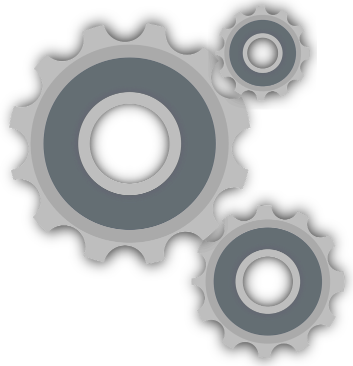
```
key: user_id
value: foo
```

```
<script>
  let stored = localStorage.getItem("user_id");
  eval("user='" + stored + "'");
</script>
```

Automated Exploit Generation

```
<script>
  let stored = localStorage.getItem("user_id");
  eval("user='" + stored + "'");
</script>
```

```
key: user_id
value: foo
```



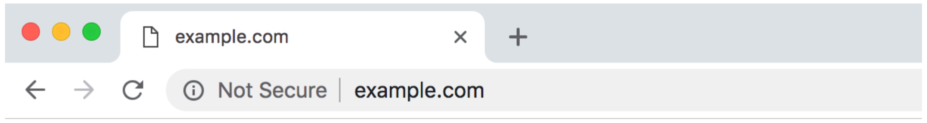
```
key: user_id
value: ';alert('XSS');//
```

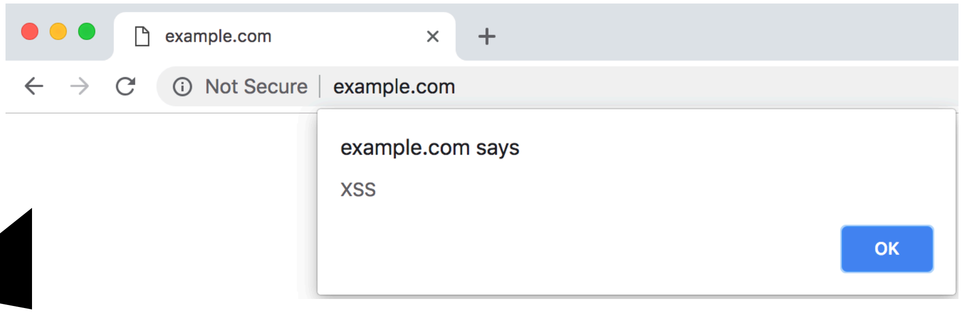
```
eval("user = 'foo'");
```

```
eval("user = '';alert('XSS');//'");
```

Validation of Exploitability

```
key: user_id
value: ';alert('XSS');//
```





example.com says

XSS

Empirical Study

- Found 1,946 out of 5,000 domains making use of stored data in their application
  - 1,645 cookies, 941 localStorage

- Found 418 domains with exploitable data flow
  - 213 (**13%**) cookies, 222 (**24%**) localStorage

Developers put trust into integrity of persisted values

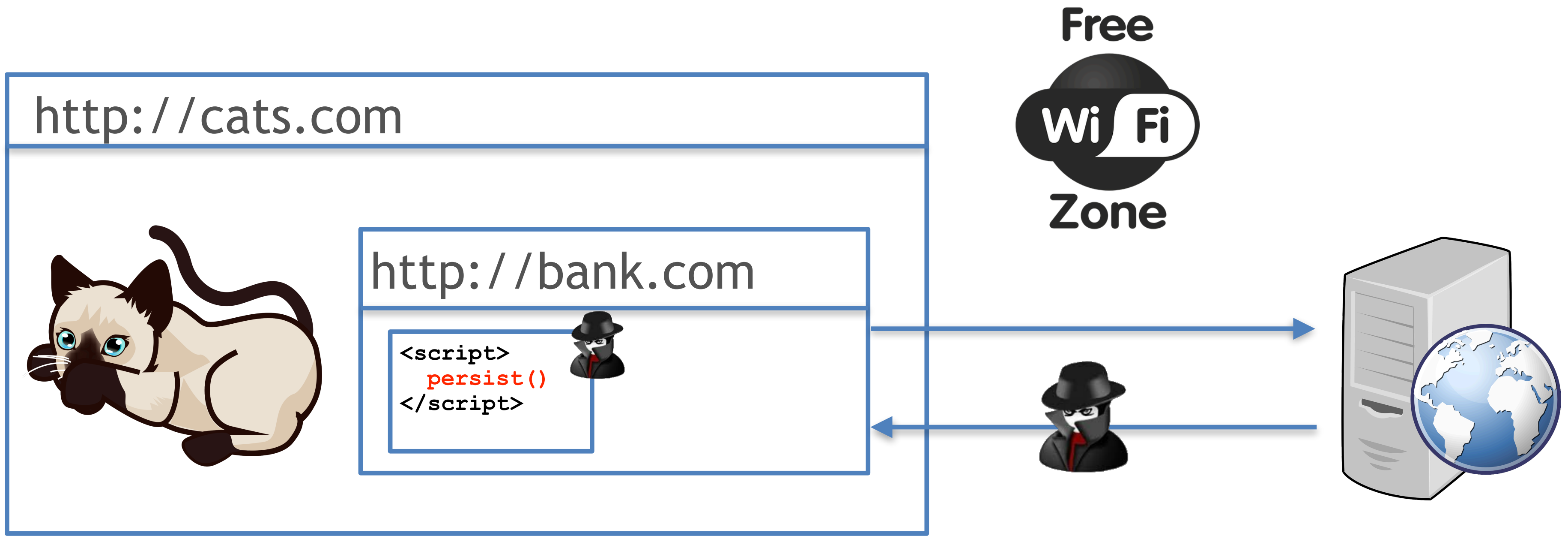
Real-world exploitability?

Infection Vector

- Requirement for successful attack: persisted malicious payload
  - single infection is sufficient
  - extracted on every page load

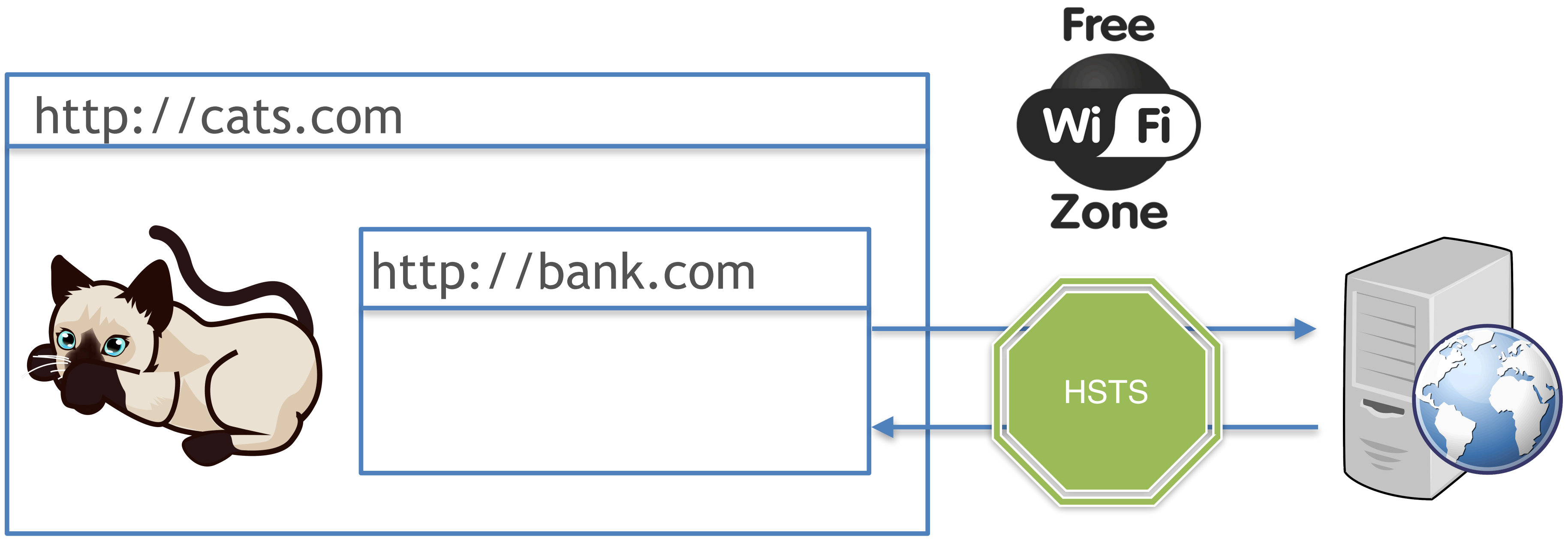
Infection Vector: Network Attacker

- Requirement for successful attack: persisted malicious payload
  - single infection is sufficient
  - extracted on every page load



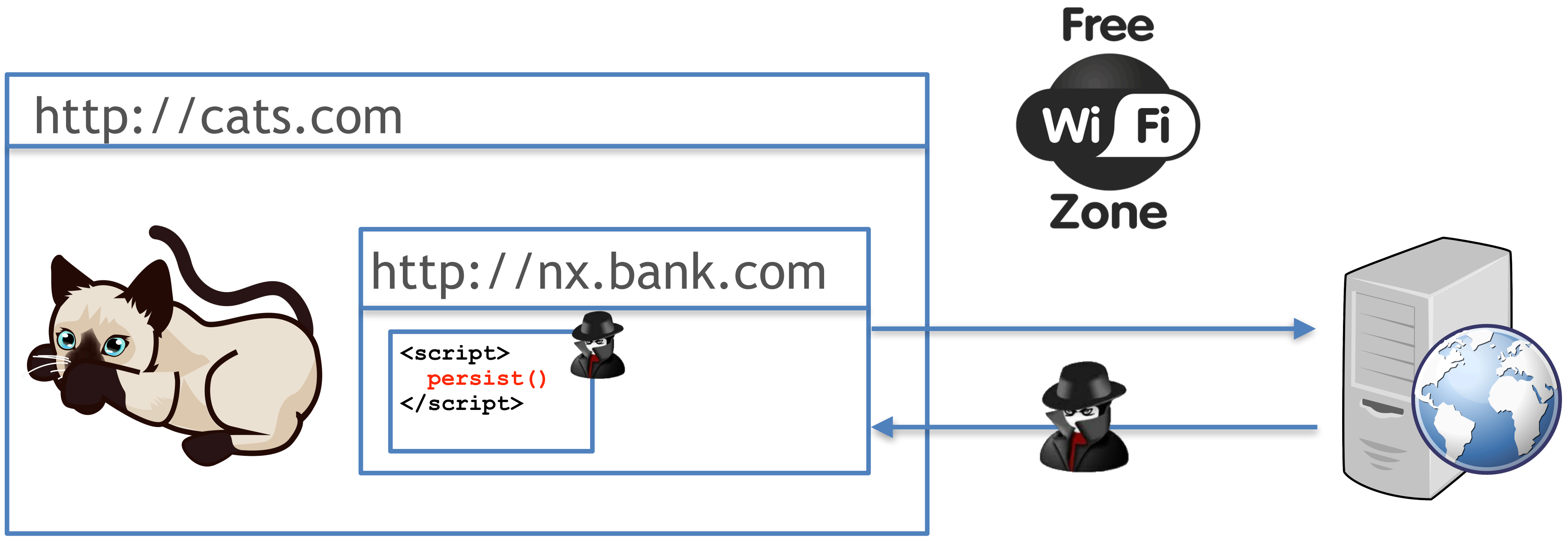
Infection Vector: Network Attacker

- Requirement for successful attack: persisted malicious payload
  - single infection is sufficient
  - extracted on every page load



Infection Vector: Network Attacker

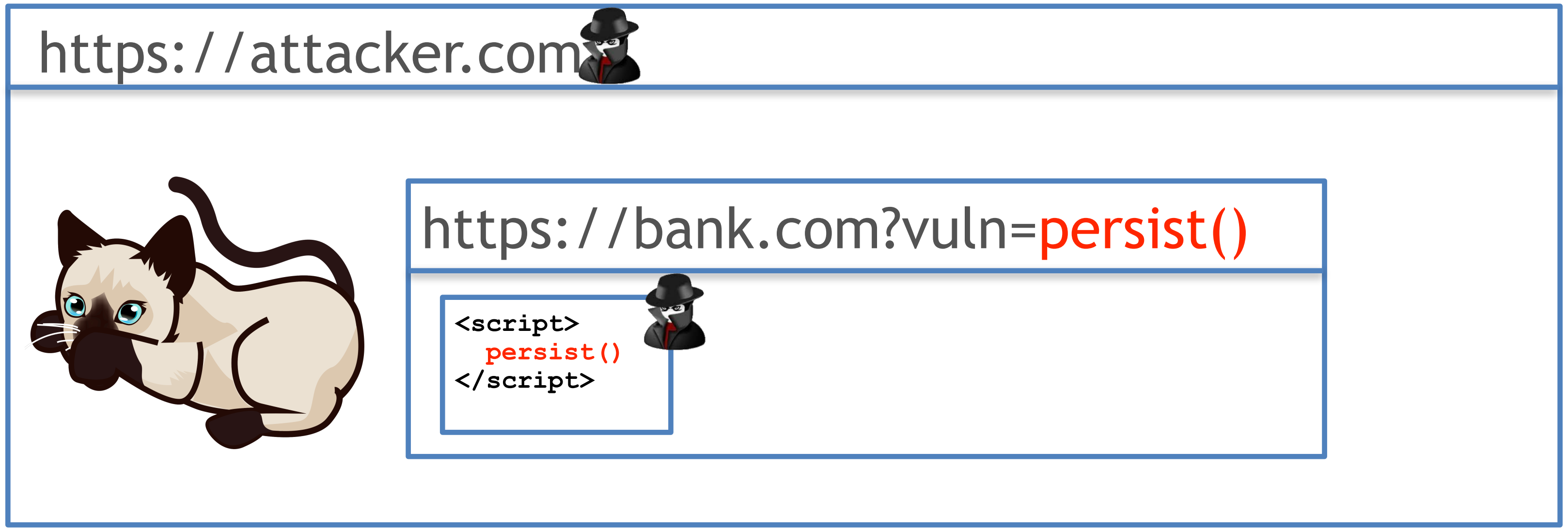
- Requirement for successful attack: persisted malicious payload
  - single infection is sufficient
  - extracted on every page load



```
<script>
  persist()
</script>
```

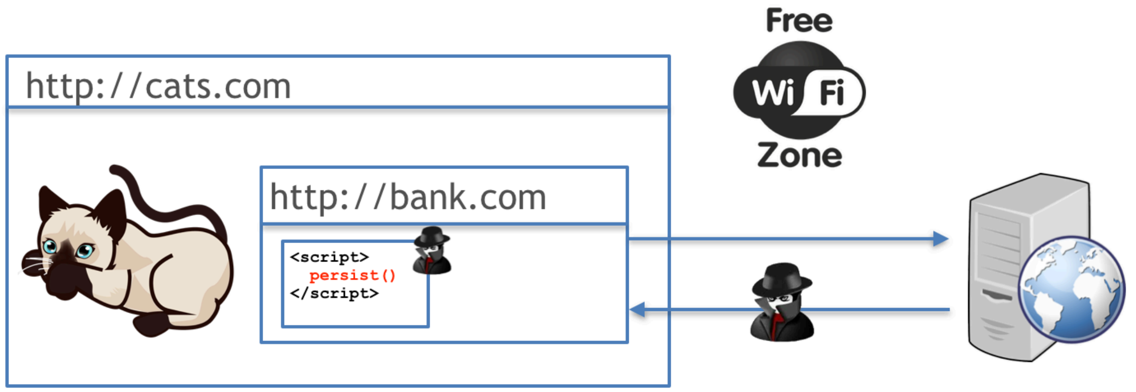
Infection Vector: Web Attacker

- Requirement for successful attack: persisted malicious payload
  - single infection is sufficient
  - extracted on every page load

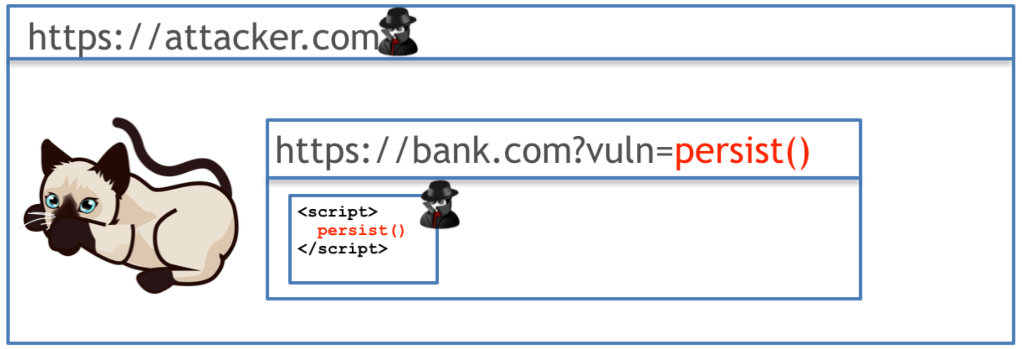


```
<script>
  persist()
</script>
```

Real-World Impact of Vulnerabilities



```
<script>
  persist()
</script>
```



```
<script>
  persist()
</script>
```

- 293 domains for Network attacker
  - lack of HTTPS
  - no includeSubdomains

- 65 domains for Web attacker
  - reflected CXSS in same origin
  - lower bound

Types of Information Stored

Unstructured Data (214)

No code/apparent structure

Context-aware sanitization

Structured Data (108)

JSON/JS Objects

JSON.parse

Code Caching (101)

HTML/JS code

Check integrity/Service Workers

Configurations (28)

Hostnames

Whitelists

Summary & Conclusion

- Persistent Client-Side XSS
  - One-time infection vectors to gain permanent foothold
  - Hard to detect since only client shows signs of infection

- Conducted the first large-scale analysis of persistent client-side XSS
  - found 1,946 domains using persistence in their application
  - 418 domains with exploitable data flow to sink

- Real-world attacker models to provide lower bound on exploitability
  - 293 domains for Network Attacker
  - 65 domains for reflected client-side XSS

Thank you for the attention.
Questions?