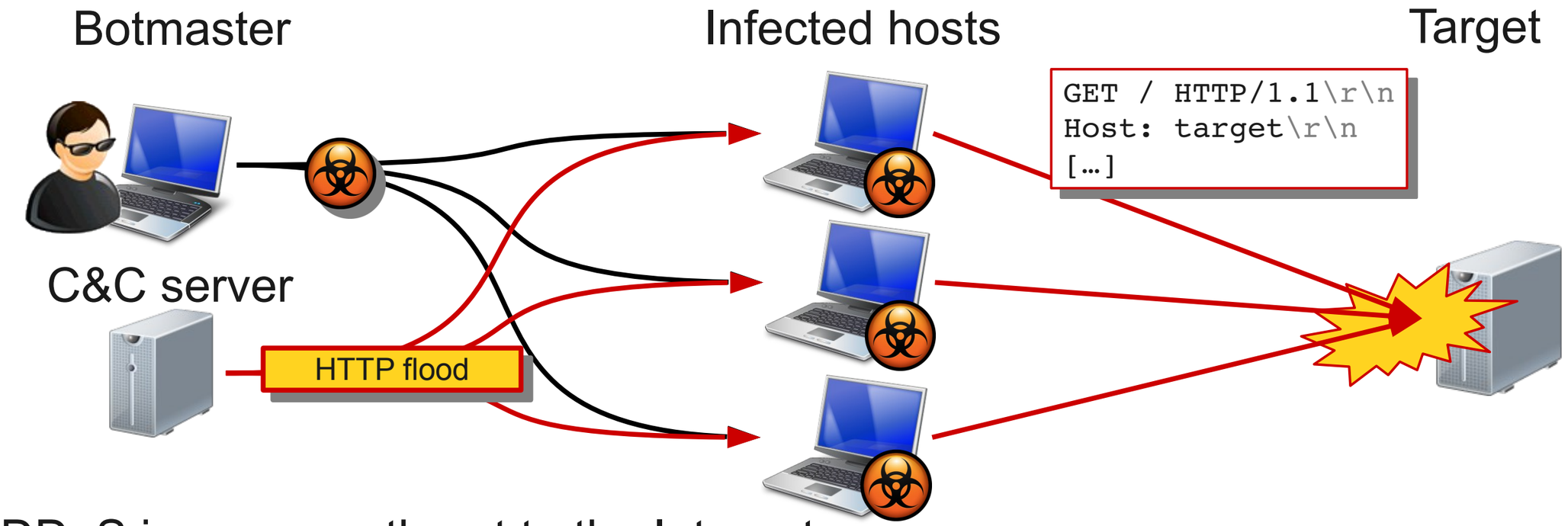
Cashing out the Great Cannon?
On Browser-based DDoS Attacks and Economics

G. Pellegrino[1], C. Rossow[1], F. J. Ryba[2], T. C. Schmidt[3], M. Wählisch[2]

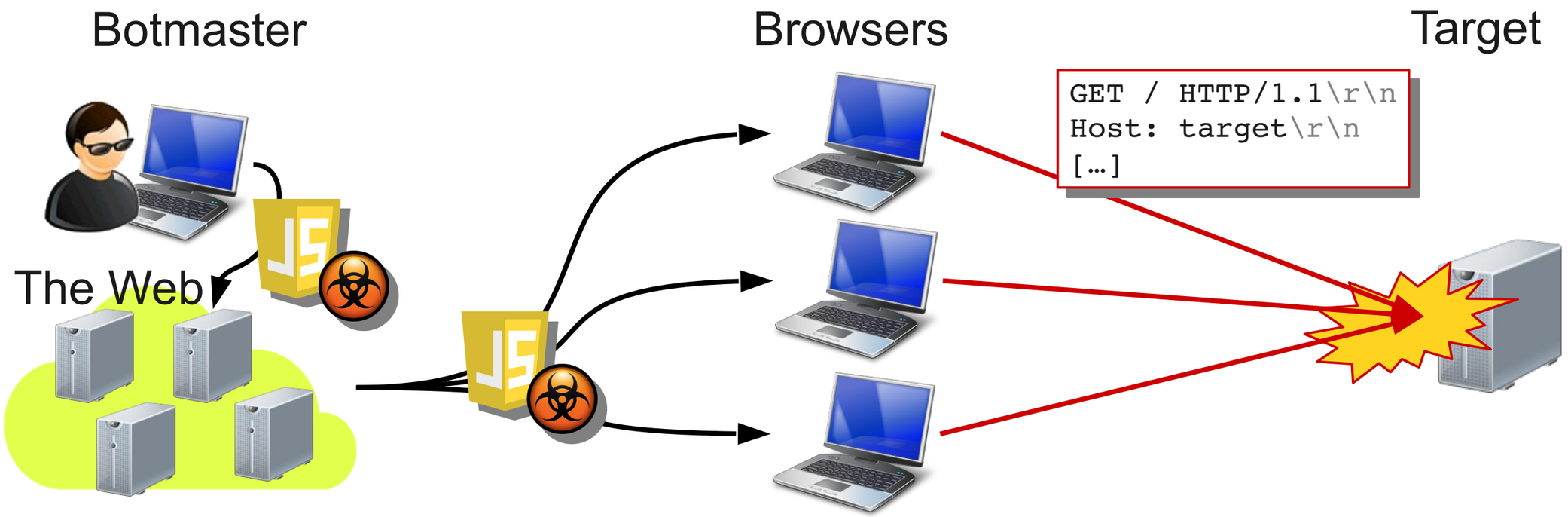
[1]CISPA / MMCI, Saarland University
[2]Freie Universität Berlin
[3]HAW Hamburg

Classical DDoS Botnets



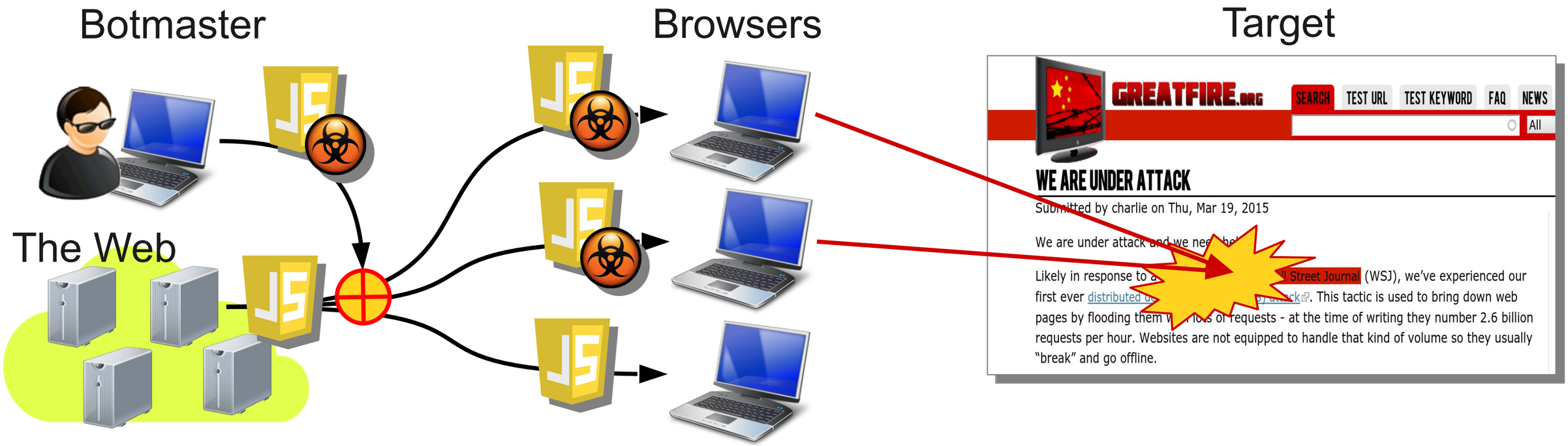
- DDoS is a severe threat to the Internet
- In classical DDoS botnets:
  - Infection-based recruitment (Drive-by download, Browser vulns, ...)
  - Architecture-dependent malware

Browser-based DDoS Botnet



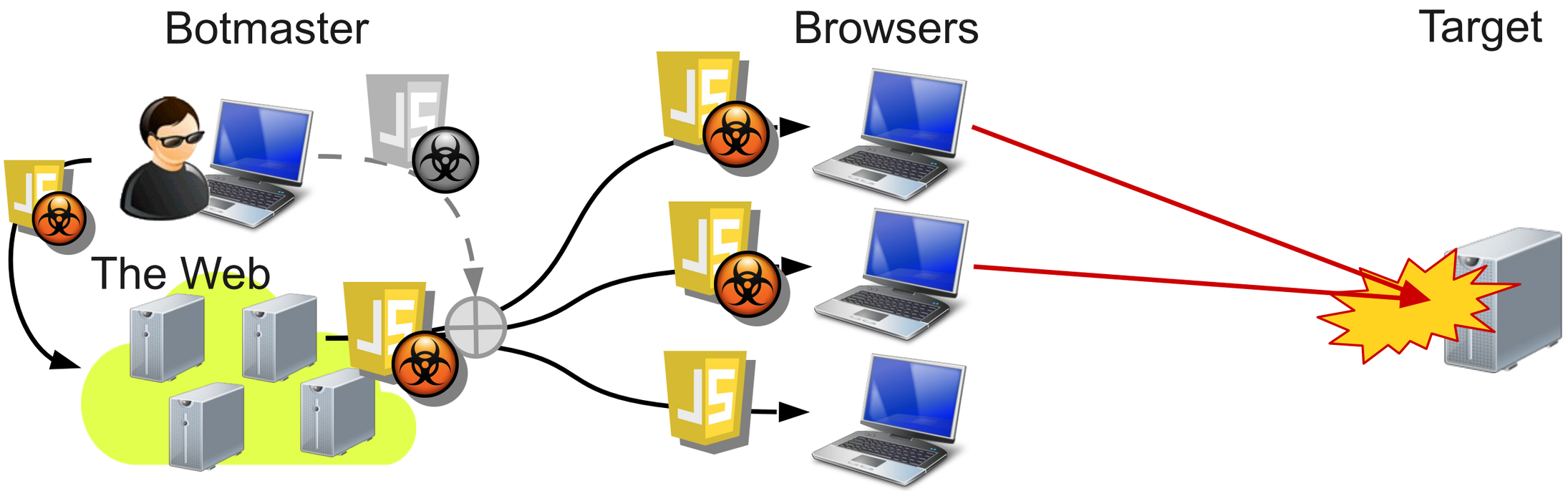
- **Browser-based botnet** a new type of botnet
  - Infectionless bots recruitment
  - Architecture-independent malware (e.g., OSX, Windows, Linux, Android)

The Great Cannon



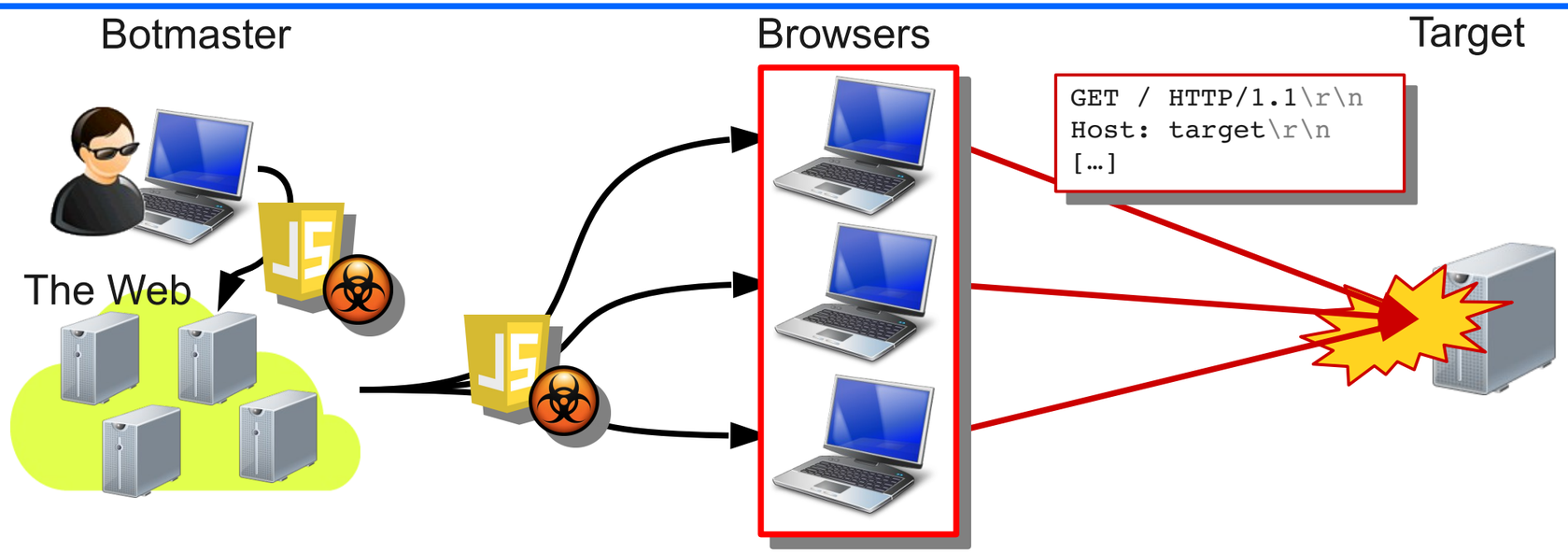
- In March 2015 first browser-based DDoS attacks [CitizenLab]
- Recruitment: Powerful attacker injects JS into HTTP conversations
  - ➔ We envision also less powerful attacker can launch similar attacks

Threat Model



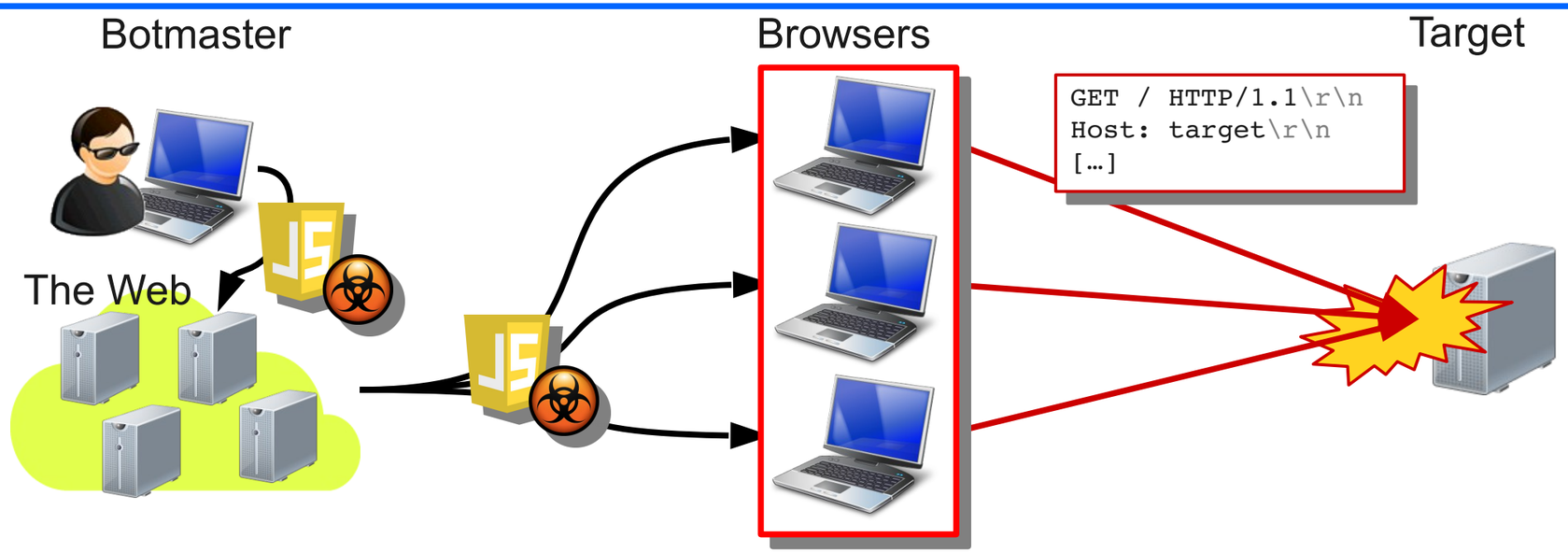
- No control of the network, e.g., no ISP
- Infiltrate JS over the Web, e.g., as advertisement [Grossman]
- Economic incentives

Toward a Great Cannon for Cyber-Criminals?



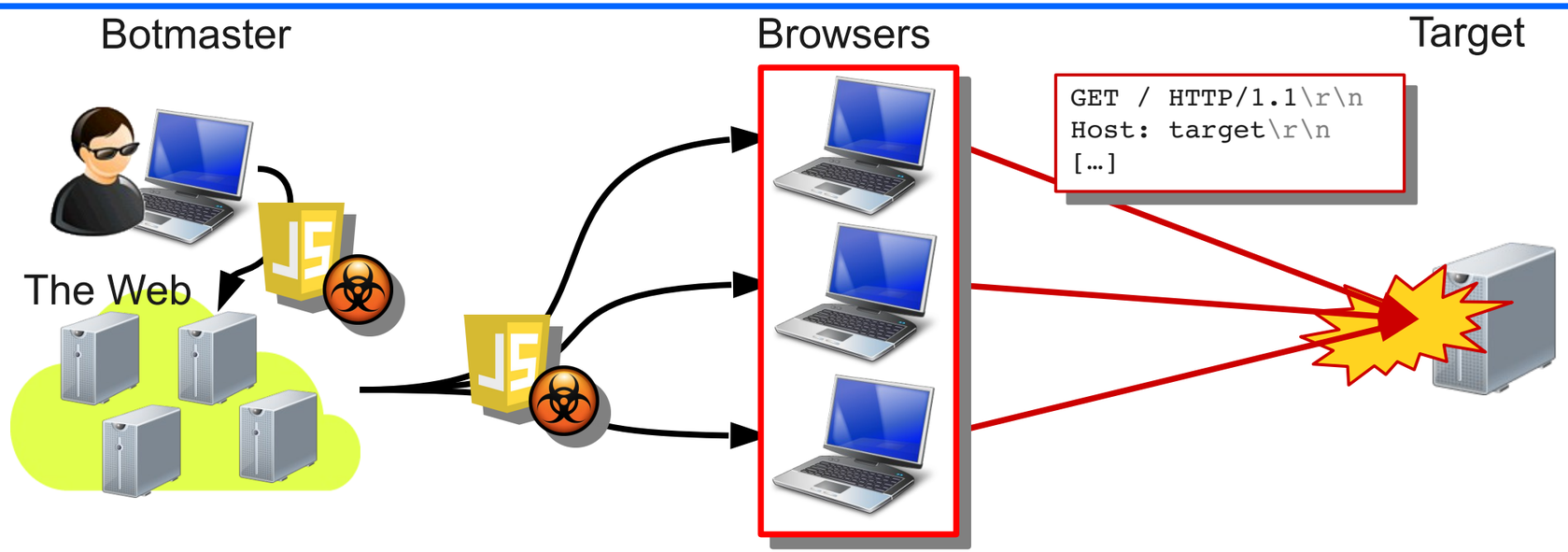
- GC showed that browsers can be used as bots

Toward a Great Cannon for Cyber-Criminals?



- GC showed that browsers can be used as bots
  - However, anecdotal knowledge only [Kuppan, Grossman]

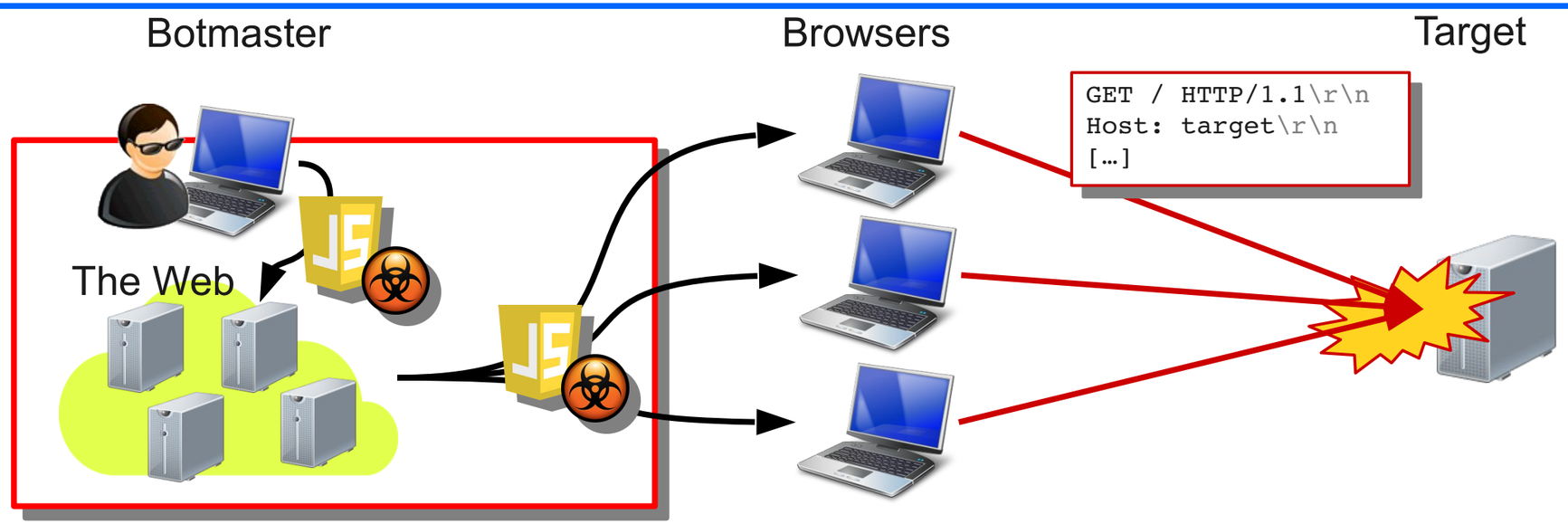
Toward a Great Cannon for Cyber-Criminals?



- GC showed that browsers can be used as bots
  - However, anecdotal knowledge only [Kuppan, Grossman]

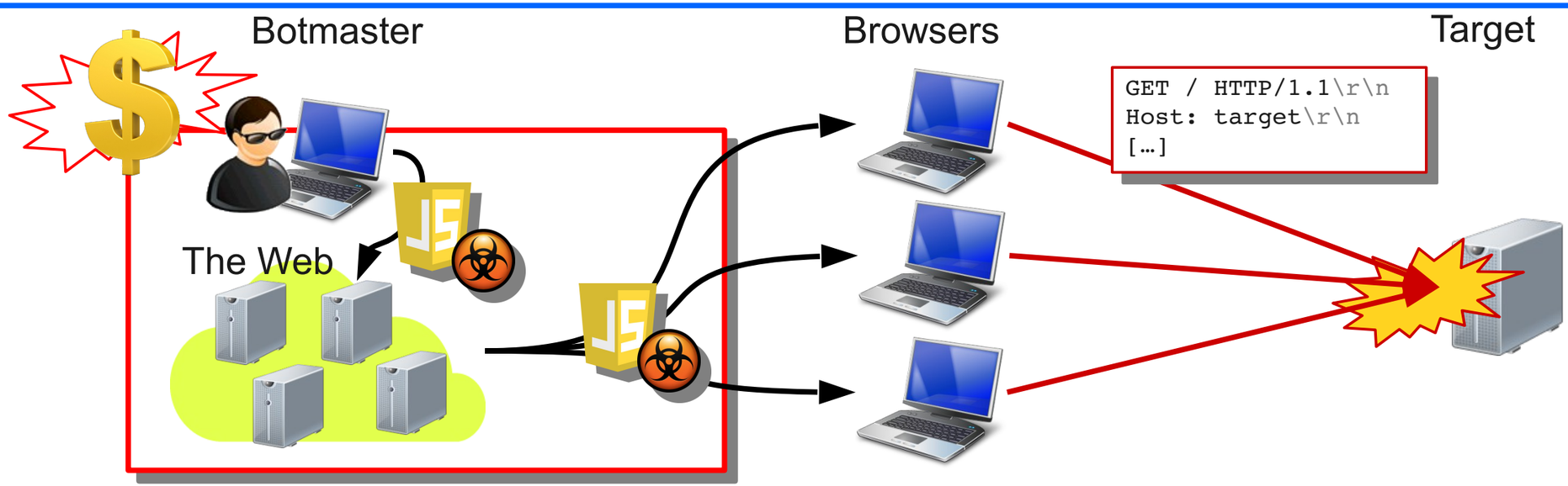
➔ To date, no systematic understanding of browser features to support DDoSes

Toward a Great Cannon for Cyber-Criminals?



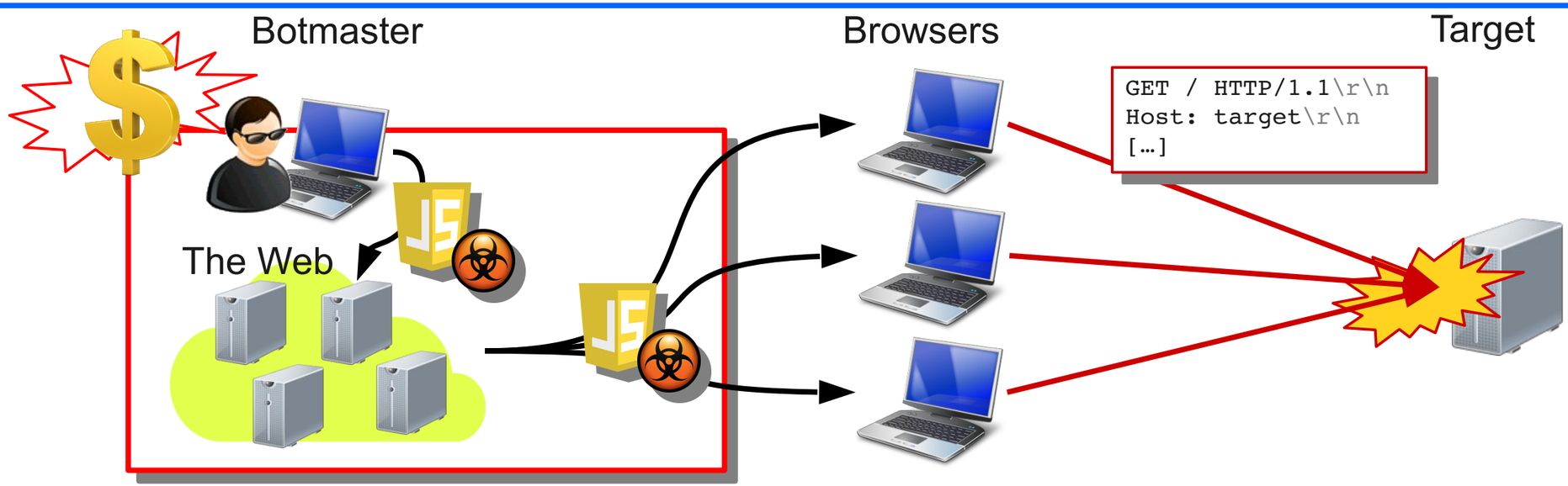
- Promising for less powerful attackers, i.e., criminals with economic incentives

Toward a Great Cannon for Cyber-Criminals?



- Promising for <u>less powerful attackers</u>, i.e., criminals with economic incentives
  - However, little is known about recruitment techniques and costs

Toward a Great Cannon for Cyber-Criminals?



- Promising for <u>less powerful attackers</u>, i.e., criminals with economic incentives
  - However, little is known about recruitment techniques and costs

➔ Hard to assess if criminals will jump on the wagon of GC-like attacks

Contents

- Review browser features
- Browser features in DoS attacks
- Cost estimation and comparison

Browser Features

Classical DDoS bots: Yoddos/DirtJumper

- Supports different DDoS attacks
  - TCP, UDP, and HTTP based flooding
- And attack variants:
  - HTTP reqs. with no `recv()`
    - Via TCP FIN or RST
  - HTTP custom `Host` and `Referer`
    - Bypass filters

Yoddos Attack Commands (Source [Welzel])

Cmd ID	Functionality	Target
0x00000001	UDP with raw socket. `rand()` spoofed IPs	host/IP
0x00000002	Same as `0x00000001`	host/IP
0x00000004	Same as `0x00000001`, single thread	host/IP
0x00000008	UDP with raw socket. Spoofed IPs	host/IP
0x00000010	Same as `0x00000008`	host/IP
0x00000020	TCP msgs with `\%d<<<<<I@C<<<<<\%s!`	host/IP
0x00000040	UDP with rnd data and msg lengths	host/IP
0x00000080	TCP with rnd data and msg lengths	host/IP
0x00000100	UDP with rnd data but structured message	host/IP
0x00000200	TCP with rnd length for each message	host/IP
0x00000400	`connect()` 200 sockets (only once)	host/IP
0x00000800	`connect()` 200 sockets (continuously)	host/IP
0x00001000	HTTP, Host and Referer fixed, no `recv()`	URL
0x00002000	HTTP, path is `/`, no `recv()`, no Referer	host/IP
0x00004000	HTTP, `no recv()`, varies path to fetch	URL
0x00008000	HTTP, `InternetOpenA()`	URL
0x00010000	Custom UDP/TCP data from C&C server	host/IP

Web Browsers as DDoS bots

- Offer communication APIs
  - e.g., XMLHttpRequest, WebSocket, and Server-Sent Events
- Other DoS-enabling JS APIs
  - Image and WebWorker APIs
- However, less flexible
  - No direct access to TCP/UDP
    - restricted to extensions...
  - No IP spoofing
- Reviewed 4 APIs ...

Yoddos Attack Commands (Source [Welzel])

Cmd ID	Functionality	Target
0x00000001	UDP with raw socket. `rand()` spoofed IPs	host/IP
0x00000002	Same as `0x00000001`	host/IP
0x00000004	Same as `0x00000001`, single thread	host/IP
0x00000008	UDP with raw socket. Spoofed IPs	host/IP
0x00000010	Same as `0x00000008`	host/IP
0x00000020	TCP msgs with `\%d<<<<<I@C<<<<<\%s!`	host/IP
0x00000040	UDP with rnd data and msg lengths	host/IP
0x00000080	TCP with rnd data and msg lengths	host/IP
0x00000100	UDP with rnd data but structured message	host/IP
0x00000200	TCP with rnd length for each message	host/IP
0x00000400	`connect()` 200 sockets (only once)	host/IP
0x00000800	`connect()` 200 sockets (continuously)	host/IP
0x00001000	HTTP, Host and Referer fixed, no `recv()`	URL
0x00002000	HTTP, path is `/`, no `recv()`, no Referer	host/IP
0x00004000	HTTP, `no recv()`, varies path to fetch	URL
0x00008000	HTTP, `InternetOpenA()`	URL
0x00010000	Custom UDP/TCP data from C&C server	host/IP

XMLHttpRequest API (1/4)

- Send HTTP requests to arbitrary targets
- Restrictions:
  - SOP and CORS, but HTTP requests are sent anyway

```
var target = "http://target/";
var xhr    = new XMLHttpRequest();
xhr.open("GET", target);
xhr.send();
```

Send HTTP request

Yoddos Attack Commands (Source [Welzel])

0x00001000	HTTP, Host and Referer fixed, no `recv()`	URL
0x00002000	HTTP, path is /, no `recv()`, no Referer	host/IP
0x00004000	HTTP, no `recv()`, varies path to fetch	URL
0x00008000	HTTP, `InternetOpenA()`	URL

XMLHttpRequest API (2/4)

- Send HTTP requests to arbitrary targets
- Restrictions:
  - SOP and CORS, but HTTP requests are sent anyway

```
var target = "http://target/";
var xhr    = new XMLHttpRequest();
xhr.open("GET", target);
xhr.send();
```

Send HTTP request

Yoddos Attack Commands (Source [Welzel])

0x00001000	HTTP, Host and Referer fixed, no `recv()`	URL
0x00002000	HTTP, path is /, no `recv()`, no Referer	host/IP
0x00004000	HTTP, no `recv()`, varies path to fetch	URL
0x00008000	HTTP, `InternetOpenA()`	URL

XMLHttpRequest API (3/4)

- Send HTTP requests to arbitrary targets
- Restrictions:
  - SOP and CORS, but HTTP requests are sent anyway
- Additional behaviors:
  - Partial control over the TCP socket life-cycle → no `rcvd()`

```
var target = "http://target/";
var xhr    = new XMLHttpRequest();
xhr.open("GET", target);

setTimeout(function() {
  xhr.abort();
}, 10);

xhr.send();
```

RST after 10 ms

Yoddos Attack Commands (Source [Welzel])

0x00001000	HTTP, Host and Referer fixed, no `recv()`	URL
0x00002000	HTTP, path is /, no `recv()`, no Referer	host/IP
0x00004000	HTTP, no `recv()`, varies path to fetch	URL
0x00008000	HTTP, `InternetOpenA()`	URL

XMLHttpRequest API (4/4)

- Send HTTP requests to arbitrary targets
- Restrictions:
  - SOP and CORS, but HTTP requests are sent anyway
- Additional behaviors:
  - Partial control over the TCP socket life-cycle → no `rcvd()`
  - Set/modify request headers
    - Except for `Host` and `Referer` (and others)

```
var target = "http://target/";
var xhr    = new XMLHttpRequest();
xhr.open("GET", target);

setTimeout(function() {
  xhr.abort();
}, 10);

xhr.send();
```

RST after 10 ms

Yoddos Attack Commands (Source [Welzel])

~~0x00001000~~	~~HTTP, Host and Referer fixed, no recv()~~	~~URL~~
~~0x00002000~~	~~HTTP, path is /, no recv(), no Referer~~	~~host/IP~~
0x00004000	HTTP, no recv(), varies path to fetch	URL
0x00008000	HTTP, InternetOpenA()	URL
~~0x00010000~~	~~Custom UDP/TCP data from C&C server~~	~~host/IP~~

Web Sockets (1/2)

- Extension of HTTP
  - Establish full-duplex stream-oriented client-server communication channel via the **WebSocket Handshake protocol**
    - Based on a HTTP request/response pair

```
var target = "ws://target/";
var ws = new WebSocket(target);
```

WebSocket Handshake

Yoddos Attack Commands (Source [Welzel])

0x00001000	HTTP, Host and Referer fixed, no `recv()`	URL
0x00002000	HTTP, path is /, no `recv()`, no Referer	host/IP
0x00004000	HTTP, no `recv()`, varies path to fetch	URL
0x00008000	HTTP, `InternetOpenA()`	URL

Web Sockets (2/2)

- Extension of HTTP
  - Establish full-duplex stream-oriented client-server communication channel via the **WebSocket Handshake protocol**
    - ➔ Based on a HTTP request/response pair
- Additional behaviors:
  - ➔ Partial control over the TCP socket life-cycle → no `rcvd()`
  - No access to request headers

```
var target = "ws://target/";

setTimeout(function () {
  ws.close();
}, 10);

var ws = new WebSocket(target);
```

RST after 10ms

Yoddos Attack Commands (Source [Welzel])

~~0x00001000~~	~~HTTP, Host and Referer fixed, no recv()~~	~~URL~~
~~0x00002000~~	~~HTTP, path is /, no recv(), no Referer~~	~~host/IP~~
0x00004000	HTTP, no recv(), varies path to fetch	URL
0x00008000	HTTP, InternetOpenA()	URL

API Evaluation

Aggressiveness

API	Browser	AVG Reqs/s	MAX Reqs/s
XMLHttpReq.	Chrome	1,005	1,886
	Firefox	2,165	**2,892**
WebSocket	Chrome	34	73
	Firefox	**0**	**0**
Server-Sent Evts	Chrome	210	941
	Firefox	258	1,907
Image	Chrome	84	109
	Firefox	751	1,916

- Firefox shows a more aggressive behavior
- 18x faster than prior tests: ~170 XHR reqs/s [Kuppan]

Aggressiveness

API	Browser	AVG Reqs/s	MAX Reqs/s
XMLHttpReq.	Chrome	1,005	1,886
	Firefox	2,165	**2,892**
WebSocket	Chrome	34	73
	Firefox	**0**	**0**
Server-Sent Evts	Chrome	210	941
	Firefox	258	1,907
Image	Chrome	84	109
	Firefox	751	1,916

Browser	Workers	AVG Reqs/s
Chrome	0	1,359
	2	966
	3	689
Firefox	0	1,456
	2	2,424
	3	2,616

- Firefox shows a more aggressive behavior
- 18x faster than prior tests: ~170 XHR reqs/s [Kuppan]

Aggressiveness

API	Browser	AVG Reqs/s	MAX Reqs/s
XMLHttpReq.	Chrome	1,005	1,886
	Firefox	2,165	**2,892**
WebSocket	Chrome	34	73
	Firefox	**0**	**0**
Server-Sent Evts	Chrome	210	941
	Firefox	258	1,907
Image	Chrome	84	109
	Firefox	751	1,916

Browser	Workers	AVG Reqs/s
Chrome ↓	0	1,359
	2	966
	3	689
Firefox ↑	0	1,456
	2	2,424
	3	2,616

- Firefox shows a more aggressive behavior
- 18x faster than prior tests: ~170 XHR reqs/s [Kuppan]

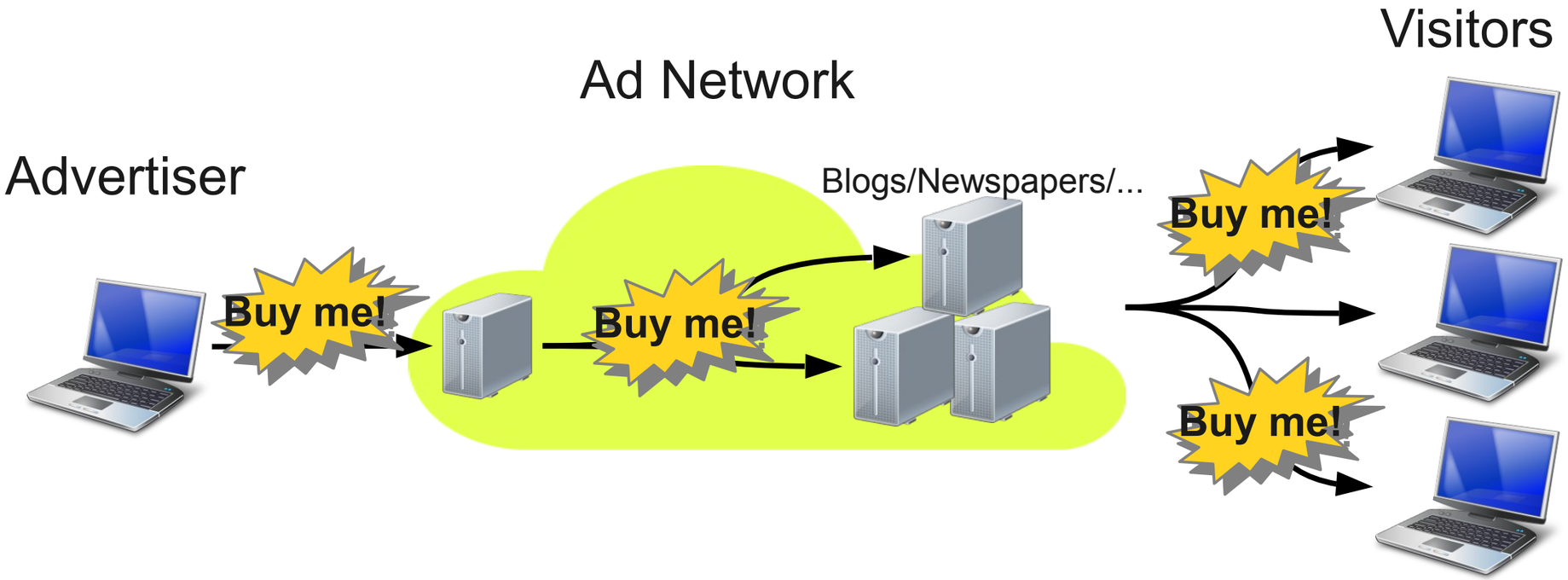
➔ ~3,000 reqs/s is a severe threat

Bot Recruitment and Cost Estimation

Recruitment Technique

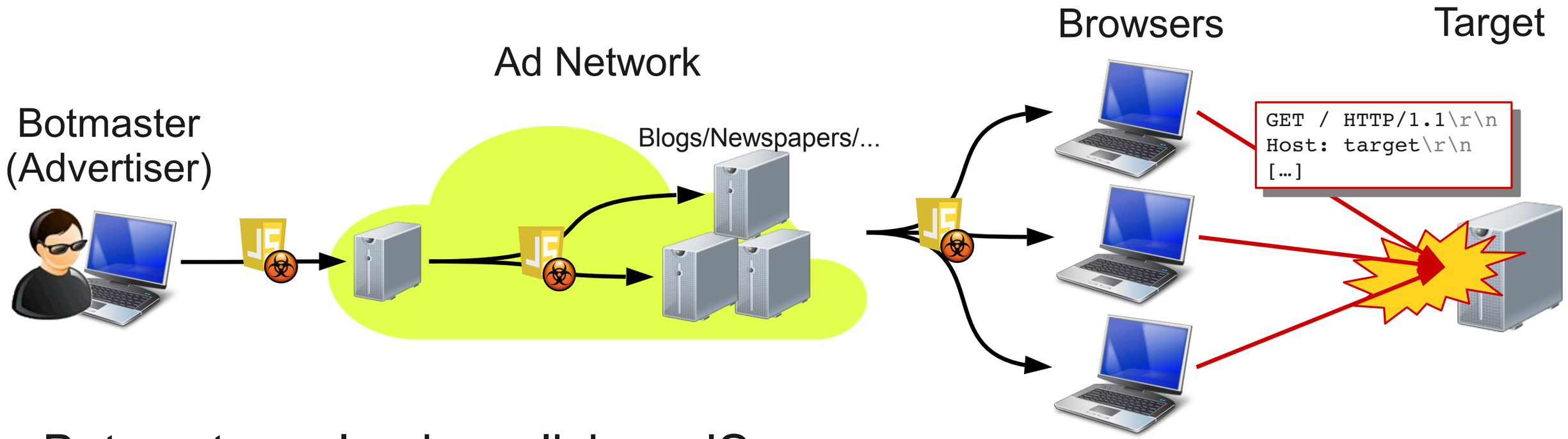
- Cost depends on the recruitment technique
- Techniques
  1. **Ad networks**
     - Malicious JS as advertisment
  2. Typosquatting
     - Registration of domain misspellings
  3. Machine-generated visits
  4. Web application hijacking
     - Using vulns to spread malicious JS, e.g., Stored XSS

Ad Networks



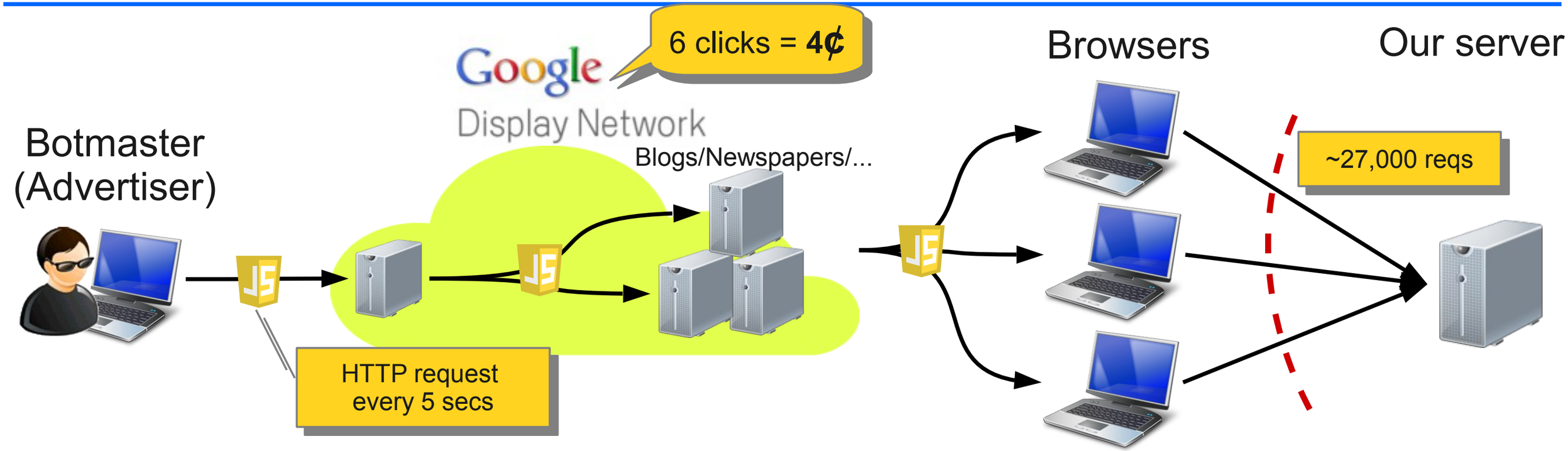
- Advertiser uploads Ad into an Ad Network
- Ad Network distributes Ads to Publishers then to Visitors

Ad Networks



- Botmaster uploads malicious JS
- Ad Network distributes malicious JS
- Attack launched by displaying the Ad

Ad Networks: Cost Estimation



- Google Display Network (May 10-17, 2015)
- Ad: ping our servers every 5 seconds
- Cost per day: **2.4¢**

Ad Networks vs Classical botnets

- Estimation as combination of prior studies (i.e., [Caballero, Rossow])
- *Pay-per-Install*: malware installation from $6 to $140 for 1000 infections [Caballero]
  - 0.6¢ and 14¢ per bot
- *Zeus infiltration 2013*: Bots stay up in ~20 days and online for ~11h a day [Rossow]
  - ➔ Cumulative online time 10 days
- Cost per day between **0.06¢** and **1.4¢** (vs. **2.4¢** of browser-based botnet)

Conclusion

Conclusion

- Systematically reviewed browser features for DDoS attacks
  - Interesting firepower w/ variety of behaviors
  - However, less flexibility wrt. classical bots
  - New rich set of APIs in the near future

- Estimated costs of browser- vs classical botnets
  - slightly higher

Limitations and Future work

- Cost: PPI vs 1 Ad Network
  - Use larger dataset and other Ad Networks
  - Explore other recruitment techniques, e.g., Typosquatting
  - Reduce the cost, e.g., less attractive ads
- Delay between Ad upload and view
  - Bot control/usability, e.g., C&C servers and responsiveness
- Botnet size less predictable
  - Study properties and influence

Takeaway

- Browser-based DDoS botnets are a severe threat
- Costs are comparable, however less flexibility
- We do ongoing research on this topic

References

[CitizenLab] "*China's Great Cannon*", B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Scott-Railton, R. Deibert, V. Paxson. url: https://citizenlab.org/2015/04/chinas-great-cannon/

[Kuppan] "*Attacking with HTML5*", L. Kuppan, Presentation at Black Hat USA 2010

[Grossmann] "*Million Browser Botnet*", J. Grossmann and M. Johansen, Presentation at Black Hat USA 2013

[Akhawe] "*Towards a Formal Foundation of Web Security*", D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, D. Song, CSF'10

[Caballero] "*The Commoditization of Malware Distribution*", J. Caballero, C. Grier, C. Kreibich, and V. Paxson, Usenix Security Symposioum 2011

[Rossow] "*P2PWNED: Modeling and Evaluating the Resilience of Peer-to-Peer Botnets*", C. Rossow, D. Andriesse, T. Werner, B. Stone-Gross, D. Plohmann, C.J. Dietrich, H. Bos, IEEE S&P 2013

[Welzel] "*On Measuring the Impact of DDoS Botnets*", A. Welzel, C. Rossow, H. Bos, EuroSec'14