Anonym activities: white- and blackhat applications

Márk Jelasity

The need for hiding

- blackhat
  - avoid the detection of criminal activity
  - hide crucial infrastructure such as “mothership” servers, monitoring and control servers, etc
- whitehat
  - protect privacy
  - fight censorship

Example systems

- BotNets
  - networks of compromised PCs
  - initially IRC-based; now increasingly P2P
  - main servers and operator wants to stay anonym
- Anonym networks
  - Dedicated (closed or open) networks
  - some variation of “mixing” communication so that participants cannot be traced back
  - remailer networks, low latency networks, friends-networks

Outline

- The Storm botnet
  - storm protocols and operation
  - fast-flux dns techniques
- Tor
  - basic idea of onion routing
  - current status and some problems

Storm Botnet

- appeared in 2007 January
- primarily for sending spam
- advanced P2P technology
- size estimated between 500,000 and 50 million
- aggressive measures for protection
  - regular download of updates to prevent reverse engineering
  - DDoS attack against external hosts that attempt to probe its operations

Storm Botnet Technology

- uses overnet protocol, based on the kademlia DHT
  - key space is 128 bit binary (usual DHT design)
  - routing is based on XOR distance
    - **eg $d(001,110)=001\oplus110=111$**
  - for 0<=i<=128 there is a “bucket” of k(=20) addresses that are at distance from $[2^i,2^{i+1})$
  - these buckets are kept fresh from observing traffic (preferring oldest, but live nodes), and proactive lookup if needed
  - lookup uses the 3 closest nodes in parallel

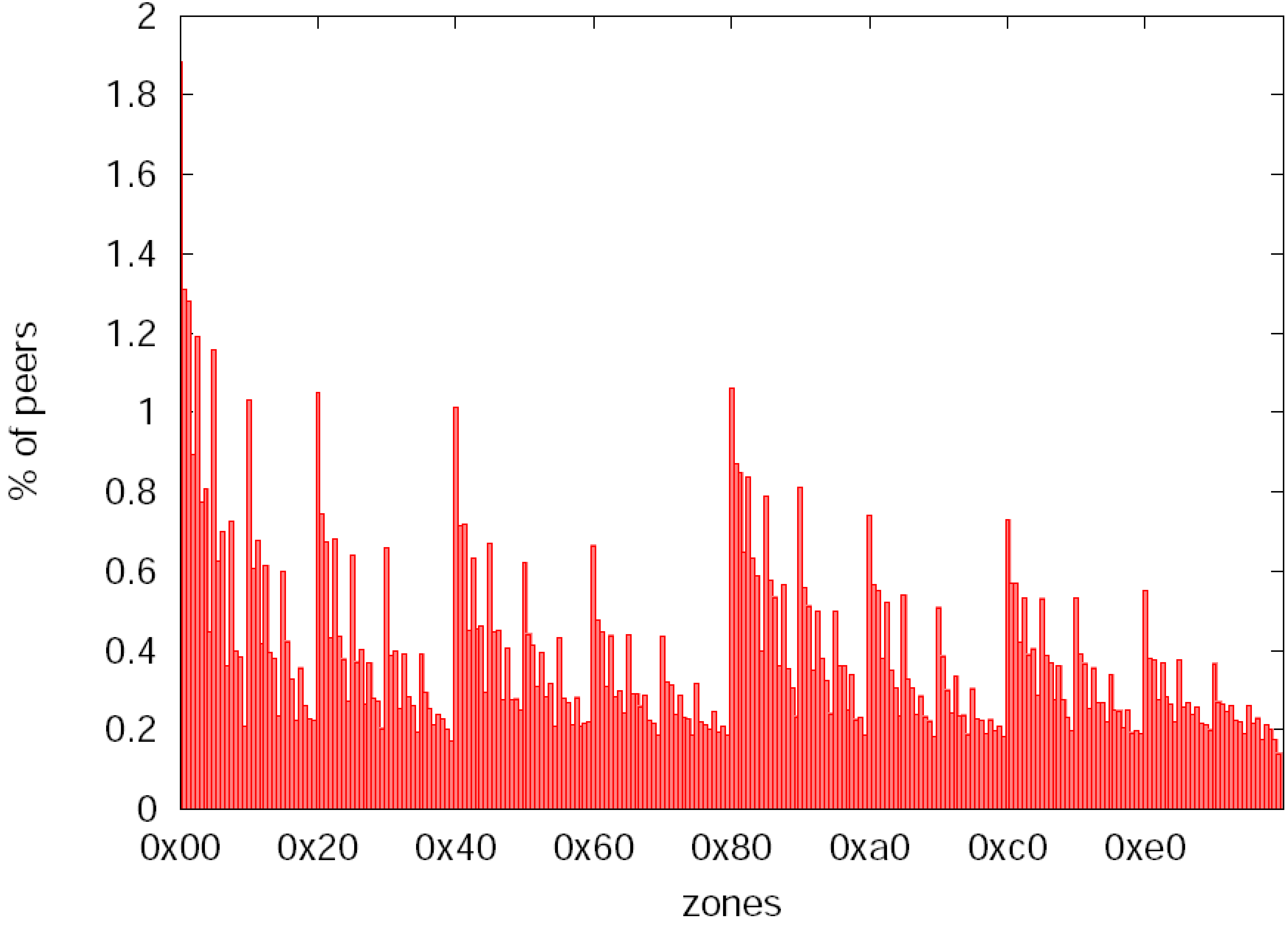
Storm Botnet Technology

- Storm bots periodically search for a given key
  - key is generated using the current date and a random number from [0,31]
  - value of that key contains an encrypted URL
  - which in turn contains new binary updates and other files to download
- for some reason
  - if this lookup fails, bots rejoin the network with new ID and repeat the search
- file sharing networks such as eDonkey can be used to store these keys! (same protocol)

Measurements

- Crawler: kademlia client that
  - performs queries for random keys
  - records node ID, IP and port that is returned
- seed list
  - 400 hard-wired IP-s in the Storm bot binary
  - storm bot run in a honeypot for 5 hours: 4000 peers
- full crawls (entire 128 bit space)
- zone crawl (space with a fixed prefix)
- estimated size: around 500,000

Uneven distribution of storm bot IDs

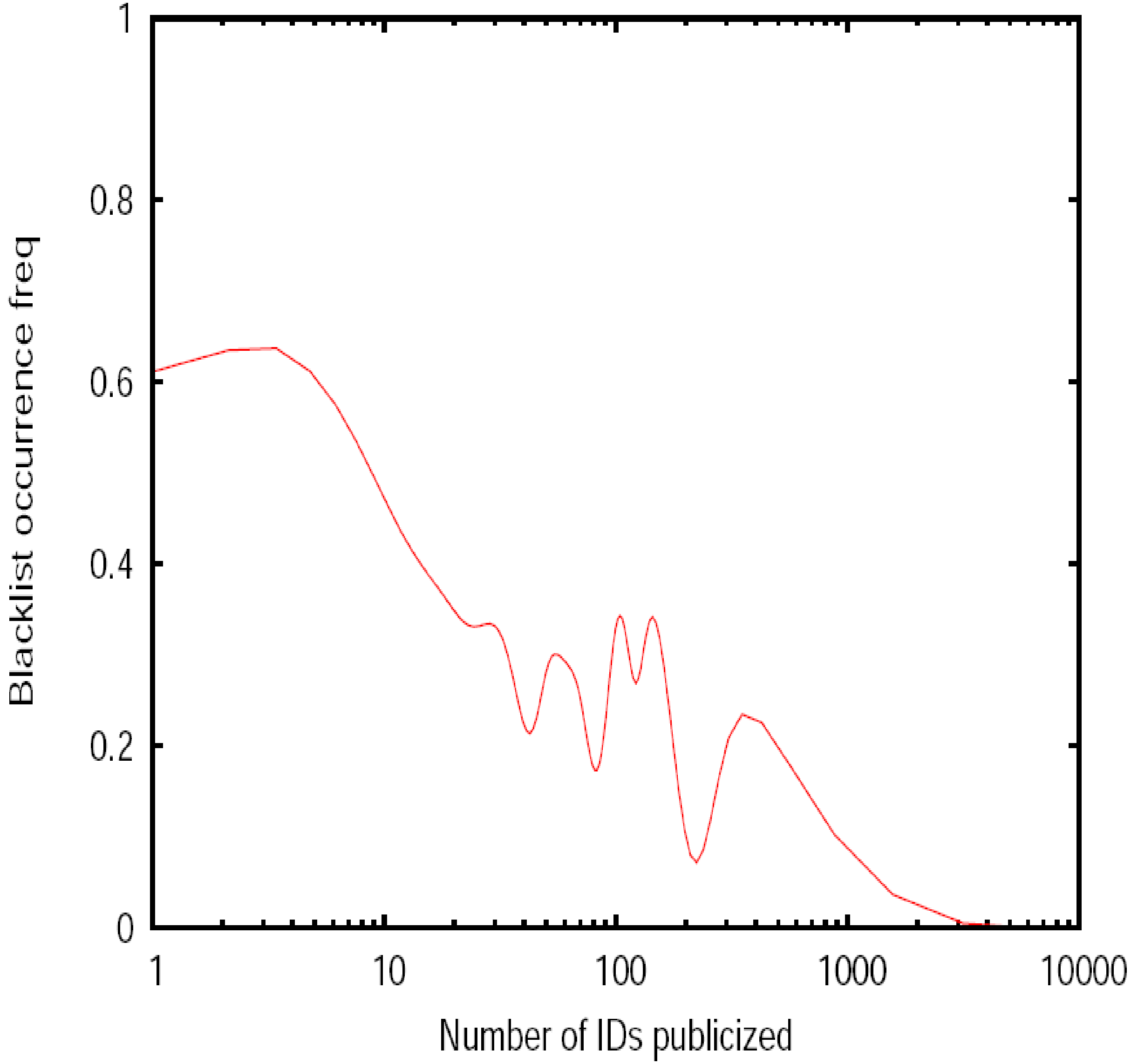


Explanation of uneven distribution: war against the Storm?

- Around 1% of returned IP addresses bogous
- But 45% of unique Ids have one of these addresses
- These IDs are responsible for the non-uniformity of the ID distribution as well
- possible explanation
  - index poisoning
  - we are witnessing efforts to fight the Storm Botnet

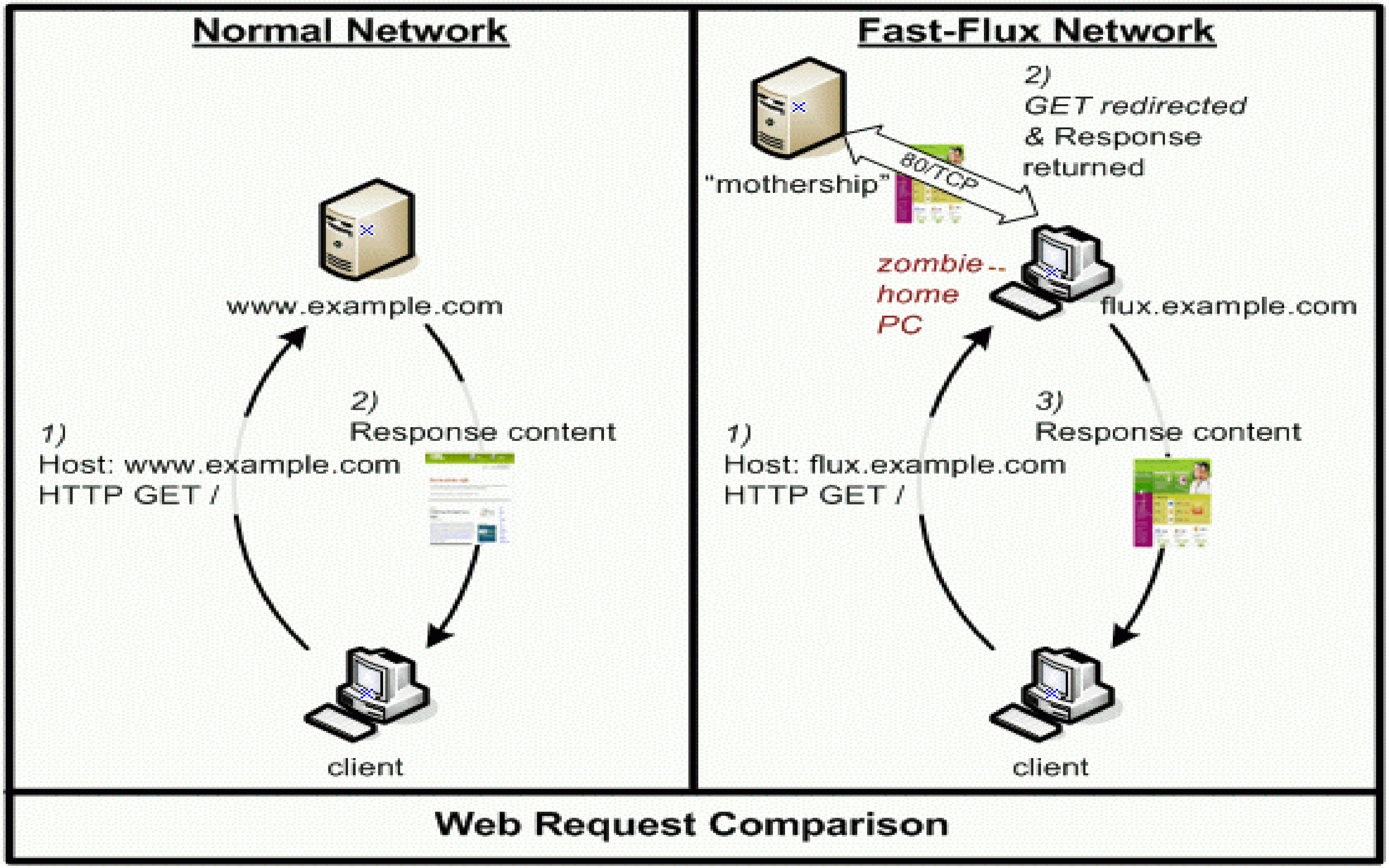
More observations of the war?

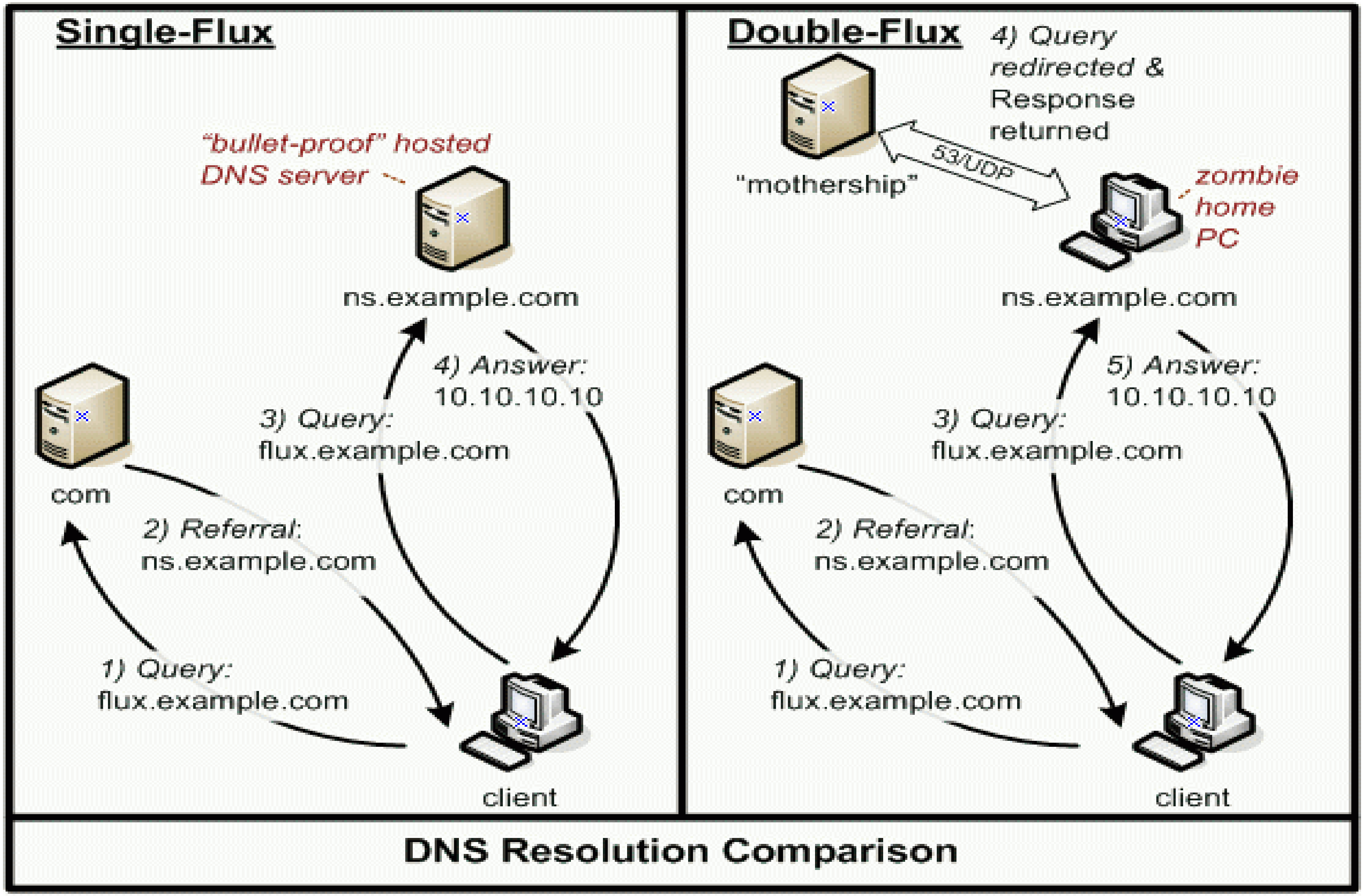
- interestingly: some non-bogous IP addresses are associated with many IDs
- these tend to be those that are not on spam blacklists...



Fast-flux service technology

- Storm was observed to use it since 2007 June
- another step to use the BotNet as a supercomputer!
- basic idea is to very quickly change the DNS record for a given hostname so that each time a different bot serves it
  - this website can host the content the spam messages point to: scam, phishing, illegal content, etc
  - remotely similar to round robin DNS, but emphasis is obfuscation not only load balancing, and actual content is often served by a single “mothership” where bots forward to


Normal Network
www.example.com
1)
Host: www.example.com
HTTP GET /
2)
Response content
client
Fast-Flux Network
"mothership"
2)
GET redirected
& Response
returned
80/TCP
zombie -- home PC
flux.example.com
1)
Host: flux.example.com
HTTP GET /
3)
Response content
client
Web Request Comparison


Single-Flux
"bullet-proof" hosted DNS server
ns.example.com
com
4) Answer: 10.10.10.10
3) Query: flux.example.com
2) Referral: ns.example.com
1) Query: flux.example.com
client
Double-Flux
4) Query redirected & Response returned
"mothership"
53/UDP
zombie home PC
ns.example.com
com
5) Answer: 10.10.10.10
3) Query: flux.example.com
2) Referral: ns.example.com
1) Query: flux.example.com
client
DNS Resolution Comparison

Use cases of fast flux

- Point is to hide the mothership behind a disposable redirection layer of compromised home PCs
- Mothership takes much longer to detect and shut down
- fast flux combined with P2P technologies are very powerful
  - need active and proactive collaboration with ISP-s: traffic filtering, sending and detecting probe packets with an intrusion detection system (IDS)

Tor

- Can provide anonymity for both clients and servers (the latter using the ".onion" domain)
- So called "onion" routing
- Originally funded by US Naval Research Lab
  - To provide protection for negotiators, agents, etc
  - but if only the Navy uses it, everyone knows it's the Navy: so it went public...
- Later taken over by Electronic Frontier Foundation (EFF)
- Currently a few thousand nodes

Tor Technology

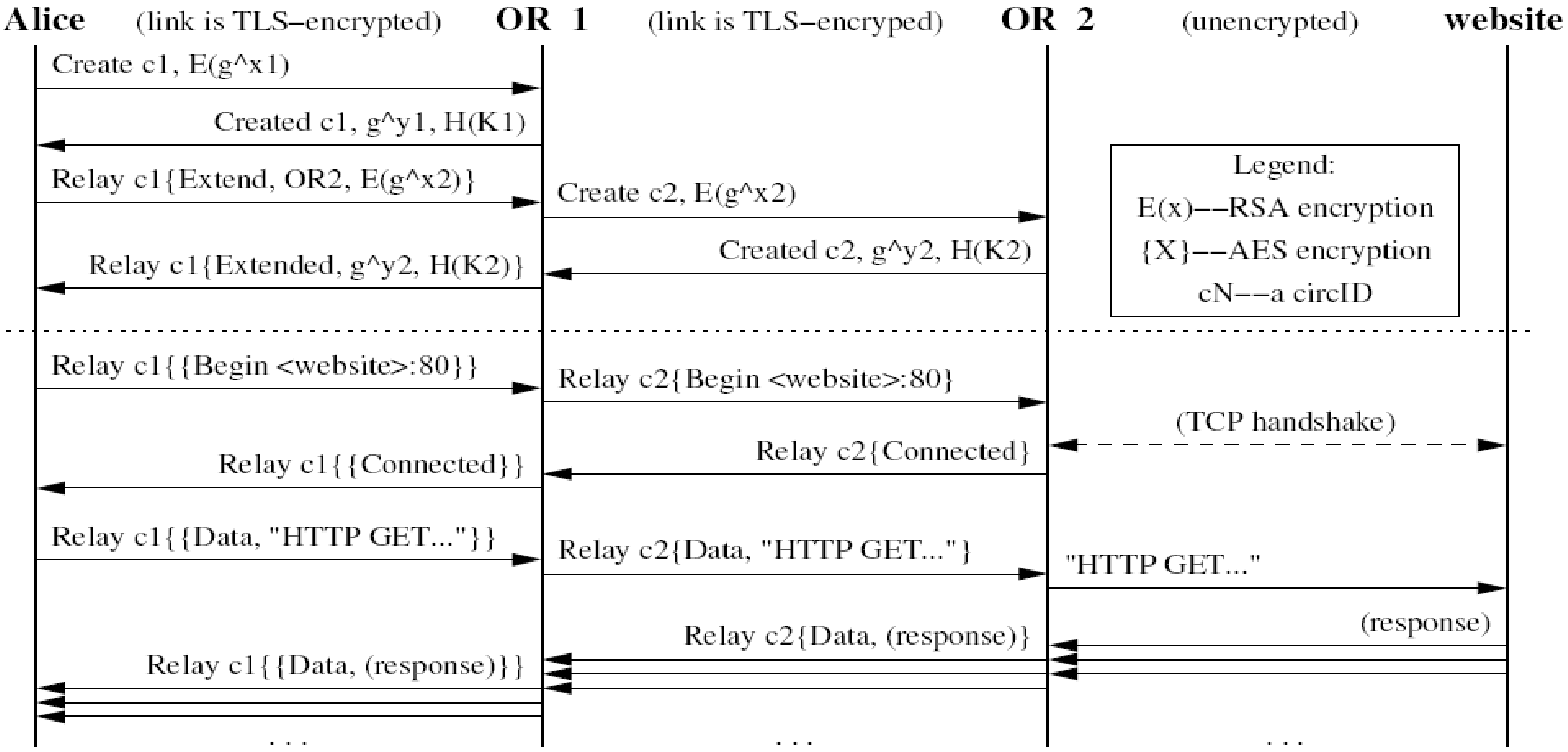
- mix based technology
  - messages are relayed by nodes while each use layers of public-key cryptography
- two kinds of mixing approaches
  - high latency: resilient to global traffic analysis but not interactive (no browsing or shell)
  - low latency: good for interactive applications but vulnerable to traffic correlation analysis (between entry points and exit points)
  - Tor is low latency

Tor Technology

- Tor is TCP level approach
  - can support any application over TCP without modifying the application client or server
  - uses IP without modification as well
- Application layer often reveals the client (eg http get, cookies, etc)
  - use Tor along with application filters such as Privoxy (privacy proxy for browsing)

Tor technology

- fully connected overlay network: TLS (SSL) connection to all other routers
- on the client end:
  - the client is talking to the Tor proxy that implements the SOCKS interface
  - the Tor exit builds an unencrypted TCP connection to the server
  - between the Tor proxy and exit onion routing



Alice (link is TLS–encrypted) **OR 1** (link is TLS–encryped) **OR 2** (unencrypted) **website**

Create c1, E(g^x1)

Created c1, g^y1, H(K1)

Relay c1{Extend, OR2, E(g^x2)}

Create c2, E(g^x2)

Created c2, g^y2, H(K2)

Relay c1{Extended, g^y2, H(K2)}

Legend:
E(x)––RSA encryption
{X}––AES encryption
cN––a circID

Relay c1{{Begin <website>:80}}

Relay c2{Begin <website>:80}

(TCP handshake)

Relay c2{Connected}

Relay c1{{Connected}}

Relay c1{{Data, "HTTP GET..."}}

Relay c2{Data, "HTTP GET..."}

"HTTP GET..."

(response)

Relay c2{Data, (response)}

Relay c1{{Data, (response)}}

. . . .

- the client never uses its public key
- onion: layers of AES encryption (a symmetric key encryption) based on secret key negotiated with Diffie Hellman during the circuit building

Problems: last step

- link between Tor exit and service is unencrypted
  - people hosting Tor exits can see all traffic (but not the origin)
- Dan Egerstad: collected high value corporate and government email addresses
  - arrested in October 2007!
  - Egerstad says
    - **traffic to these email accounts probably originated from spies and not original owners**
    - **web traffic is mostly porn...**

Other problems

- DNS leak
  - resolving DNS requests is still direct
  - latest version includes DNS resolver (understands .onion domain as well)
- traffic analysis
  - techniques exist that capture correlated traffic without global knowledge
- misuse
  - bittorrent clients often support Tor: huge traffic
  - criminals wanting to avoid detection

Some refs

- Papers this presentation used material from

Patrick Gray. The hack of the year. The Sunday Morning Herald, November 2007.

Sandeep Sarat and Andreas Terzis. Measuring the storm worm network. Technical Report 01-10-2007, Hopkins InterNetworking Research Group, Johns Hopkins University, 2007.

The Honeynet Project and Research Alliance.
Know your enemy: Fast-flux service networks, 2007.

Roger Dingledine, Nick Mathewson, and Paul Syverson.
Tor: the second-generation onion router. In Proceedings of the 13th USENIX Security Symposium (SSYM'04), pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

- The course website
  - http://www.inf.u-szeged.hu/~jelasity/p2p/