Chair for Network Architectures and Services
Department of Informatics
TU München – Prof. Carle

Peer-to-Peer Systems and Security IN2194

Chapter 2 Security 2.3 Resistance

Dipl.-Inform. Heiko Niedermayer
Christian Grothoff, PhD
Prof. Dr.-Ing. Georg Carle

Overview

- Motivation
- Censorship-Resistance
  - Freenet
  - Freenet 0.7 / Darknets
- Bot networks
  - Fast Flux, Domain Flux
  - Conficker

Censorship-Resistance

Censorship-Resistance

Motivation

Security and Peer-to-Peer – a chance?

- Can we ***use Peer-to-Peer networks for achieving security?***
- But what is good about Peer-to-Peer….?
  - No single node is responsible for the network.
    → All nodes can be replaced.
  - Nodes all over the world
    → No local attacker can physically shutdown the entire network.
  - You can do it on your own.
    → Existance of network may be unknown to others.
  - …

→ Use it for Denial-of-Service-resistance? Cencorship-resistance?

Censorship-resistance

Eternity Service

- Ross Anderson proposed the idea of an Eternity Service in 1996, the network equivalent to the Gutenberg press that made replication of books easy and knowledge available
  – for many and for long / forever.
- An eternity service needs to witstand the most extreme attacks just like the original Internet was planned witstand nuclear warfare.

Censorship-resistance

- A censorship-resistant system should disable a reasonable attacker to prevent the access to a certain piece of information (*item*).
- The attacker should also not be able to destroy the piece of information.

Censorship-resistance – Requirements

Security for Nodes / User

- Requests
  - A user requesting a critical item may also have to fear attacks against her.
    → So, censorship-resistant systems need to enable users to request items without fear of attack.
    - Consequence: A node/user should be able to deny that it was her requesting an item.
- Serving a request
  - A node that serves or forwards a reply needs to be protected.
    - It should be able to deny being the server or author for an item.
- Storage
  - Users may also like to censor items themselves (= attacker) or should be protected as they provide storage for the service.
    - A node/user should be able to deny that she knew what was stored.

Plausible deniability (Security Goal, deutsch: Glaubhafte Abstreitbarkeit)

- An entity can claim that it is „innocent“ with high-enough probability despite being involved (e.g. relaying an unknown item instead of being the source).

Claimed! (→„innocent action“) *Denied! (→„bad action“)*

Freenet

Freenetproject.org

Freenet

- By Ian Clarke, Oskar Sandberg, et. al.
- A censor-resistant, secure, distributed, and unstructured Peer-to-Peer network.
- Versions
  - 0.5 current implementation of „classic freenet“ idea
  - 0.7 new version in 2006 with new network strategy (combining darknet idea with small-world graph)

Freenet – Identification of data/information

Any information in Freenet is referenced through its hash key.

Hash keys (Global Unique IDs = GUIDs) in Freenet

- Content Hash Keys (CHK)
  - CSK = hash(content)
  - Unique identification of a content / file → for low-level data-storage
  - Data referenced by CHK cannot be updated.
- Signed Subspace Keys (SSK)
  - Private namespace that can be read by any user with knowledge of public key, but only written by its owner with her private keys.
  - Private key (→ signature) can be used to update the content stored at a SSK.
  - SSK for files is hash of public key and file name (item name).
  - Used for user and meta information.
    - E.g. pointers to CHKs with content, pointers to chunks of a file

Data Security / Confidentiality

- Data is recommended to be encrypted with a key only known to legitimate users. Keys should be passed to users together with passing the GUID.

Freenet – Join / Routing (classic, Freenet 0.5)

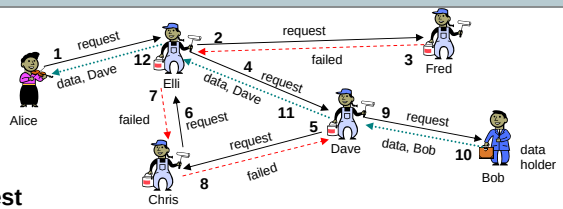
Node Join

- Node creates Public/Private key pair
  - Public key is identity of a node.
  - GUID of node is different from public key and used to determine which IDs of data to store.
- Sends an announcement to a known node (rendezvous node)
  - The announcement has a TTL and is sent to a random next hop at each hop.
  - All nodes that received the announcement randomly create a GUID for the node which determines its responsibility for keys.

Routing

- The next hop is determined according to GUID and the target ID. The hop with the closed GUID to the target ID is selected.
- When processing requests for items close to its GUID, a node learns about a node closer to the item with similar GUID and will preferable connect to them.

Freenet – Data Request and Maintenance



Data Request

- The request is forwarded to node with closest GUID. In case it reports a failure, the 2nd best (step 4 and step 9 in example), 3rd best, etc. is chosen.
- Request
  - Message has their own message ID, so predecessor is known and loops (step 6/7 in example) can be detected.
  - TTL (Time To Live Counter) limits the number of hops and is decremented at each hop.
- Reply
  - Send the message with data and source locator (for connecting to nodes with close GUID data, see last slide) to the predecessor in the path.
  - The predecessor caches the data with higher probability the closer it is to the target. The predecessor may also alter the source (step 11 in the example, from Bob to Dave).
- Cache Maintenance
  - If new file arrives, delete least popular cache entries if necessary.

Freenet – Inserting Data

Inserting Data

- Determine GUID for file (CHK, SSK)
- Perform lookup for GUID like in request.
- IF file with GUID found THEN insert fails.
- IF no file with GUID found THEN
  - „All Clear“ message is sent to source.
  - Source sends the data down the path.
  - Each node on the path
    - Verifies and stores the data.
    - Links the data to current source mentioned in the packet and with some probability changes the source entry in the packet to itself.

Freenet – Basic Security Concept

Requesting an item

- Originator of request not clear as predecessor could be requestor or relay (TTL may leak some information though).
- Unless the observer has meta-knowledge, the requested GUID and item appear to be random.

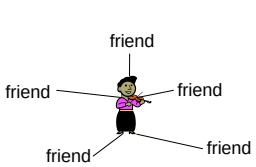
Serving an item

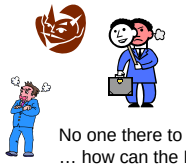
- Reply messages are altered occasionally on the path, so that it is not clear if the node mentioned in the reply message is source or relay.
- Unless the observer has meta-knowledge, the served GUID and item appear to be random.

Storing an item

- CHK=hash(data), data encrypted with unknown key.
- For a node that is simply storing the data the data item looks like a random GUID with random data.
  - Node cannot know what it is. Thus, it can deny knowledge and it cannot censor data it does not like.

Darknets





Darknet

- A darknet is a private network in which a user only connects to other users she fully trusts (friends).
- Usually, darknets are small with only few participants that form a closed group and thus, a *closed network*.
  → remember this strategy for securing a P2P network
- Darknets are not necessarily Peer-to-Peer as one peer could act as server or hub.

Freenet 0.7 (Darknet approach)

Freenet 0.7

- Build a global darknet.
- *No links to any other node than our predefined friends!*
  - No „learning“ algorithm for connecting to new nodes after replies.
    - Thus, no need to propagate source to optimize graph (cluster close GUIDs).
- The darknet is expected to be a small-world graph.
- Routing Optimization
  - While the graph is fixed, Freenet 0.7 optimizes the GUID embeddings of the nodes.
  - At first, each node is again assigned a GUID (circular ID space).
  - Nodes randomly contact other nodes (via their neighbors). Two nodes switch their GUIDs when the switch helps to reduce the product of their edge distances (objective for optimization).

- Problem: Can nodes differentiate between correct and wrong IDs of nodes in some hop distance? No → Possiblity to attack routing.

Bot networks

Bot networks

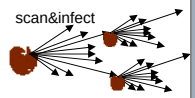
Virus / Worm / Bot

Virus

- Malware that attaches itself to a host software or script.
- If the host is executed, the malware is executed.

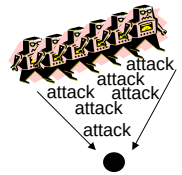
Worm

- Self-replicating malware.
- Uses network to distribute itself.
  - Exploits weaknesses in operating system or networked software.



Bot

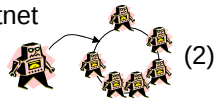
- Program to automate task, here: focus on malicious bots.
- Often self-replicating malware.
  - Like worms, but more passive as bots do not want to be detected.
- Bots (also called Zombies) are used to launch distributed attacks.
  - Distributed Denial-of-Service (DDoS), Spam, …



Botnets are Business

(1) Botnet master (herder) uses a weakness / hole to enter other computers.

(2) Infected computers register themselves in the botnet and wait for commands of their master.



(3) Someone pays the herder to use some of the bots for attacks, spam, or hosting of illegal sites (4).

- Contrary to worms, the botnet is not the attack itself. Its use is to launch subsequent attacks. Botnets are used for cybercrime and cyberwar.

Bots and Botnets

A bot needs

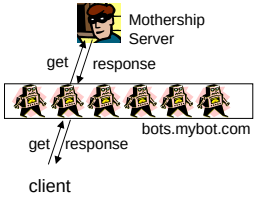
- Mechanisms to infect other computers
  - Exploit weakness in software, protocols, passwords, …
  - Method to scan for computers with weakness
- Methods to register bot and maintain botnet
- Command & Control interface
  - For the owner (herder) and for users of the botnet.
- Attack code
  - For DDoS, Spam, Phishing, etc.
- Code for self-protection
- Download and Update of bot software

Fast Flux Networks

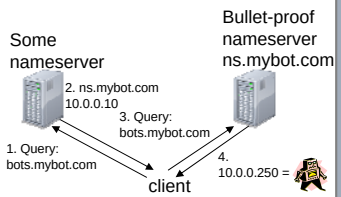
Fast Flux

- Botnet uses DNS for registration of bots, communication, or hosting.
- Idea
  - Map a domain to a set of bots that operate as proxies to herder (mothership)
  - Round robin of addresses with short TTL of DNS Ressource Records.
    - e.g. valid for 3 min..
- Single-Flux
  - A name server that is bullet-proof is used to manage a third-level-domain.
  - The DNS resolution will return a different bot everytime.

Fast Flux Hosting



Single Flux DNS



Domain Flux

Observation

- Fast Flux relies on the cooperation of nameservers.
- What if domains get dropped or access to the nameserver gets blocked?

⇒ Change domain names regularily.

Domain Flux (Random Domain Names)

- Current bots like Conficker change the domains frequently according to a pseudo random number generator.
- Bots connect to some of the random domain names within a botspecific time-interval.
- Advantages
  - Before a provider recognizes the abuse the botnet shifts to another domain.
  - If a domain is shut down, the bots are not lost, but connect to a new one.

Security Concepts for bots

Communication and Orders

- Public key cryptograhpy instead of passwords for orders.
- Messages and payload protected with HMAC and symmetric encryption.
- Signature of code and updates.

Self-Protection

- Root kit and other local obfuscation → against detection
  - Stop scanners, operating system updates, …
- Avoid IP ranges of OS and security companies → against analysis
- Avoid attacking own computer
  - e.g. Conficker does not start when Ukranian keyboard is used.
- Some bots patch the exploited weakness.

Conficker

Conficker

- Botnet that appeared late 2008, size estimation > 10 million computers.
- Modular structure
  - Infection only contains a basic bot.
  - Further modules downloaded from botnet, e.g. a scareware module in April 2009.
- Self-Protection
  - Cryptography
    - Updates and orders are secured with 4096 bit RSA signatures (OpenSSL library).
    - MD6 und RC4 for message protection.
  - Analysis
    - Obfuscation of code to fight analysis.
    - Blacklisting of IP ranges from security companies, Microsoft, etc.
  - Stops windows update, virus scanners, safe-mode, … (varies from version to version)
- Infection
  - MS08-067 server service vulnerability, but also uses other ways to spread like USB drives and network shares.
- Changing domain names
  - Conficker.C 50000 per day from 110 tld suffixes. A bot randomly checks 500 of them.
    - Current date determined from standard websites with time information.
    - Collides with existing 150-200 websites per day.