Blacklisting malicious websites using peer-to-peer technologies.

Bachelor thesis exposé – Philipp C. Heckel – 25 Feb 2009

Table of Contents

1. General Idea

As the Internet has evolved to the powerful network it is today, more and more criminal minds try to exploit it for their needs by poisoning websites with malicious content. Once a website is infected and a user browses the site with a vulnerable web browser, it infects the user's computer and installs malicious software (malware) on it.

There are many different possibilities to protect the users' workstations from being infected, be it periodical software updates or anti-virus software, however most approaches are rather reactive. The approach suggested in this thesis is to prevent users from browsing these sites by providing a service to check a website for malignity *before* the user opens it. Hence if a website has been reported to be malicious, the browser can warn the user and suggest not to visit it.

To provide a system like this, three basic components are required:

- Core System: A publicly reachable database providing a service to query it for the malignity of URLs or domains. This database will be called the *URL blacklist*.
- Client Honeynet: An instance that identifies malicious websites and fills the database with up-to-date information.
- Browser Plug-In: A client for querying the database for the malignity of an URL or domain in form of a browser plug-in.

2. System Design

The thesis' main objective is to design and implement the system core, an interface for the client honeynet to fill the system with information about malicious websites, as well as, an interface for the browser plug-in to use the service.

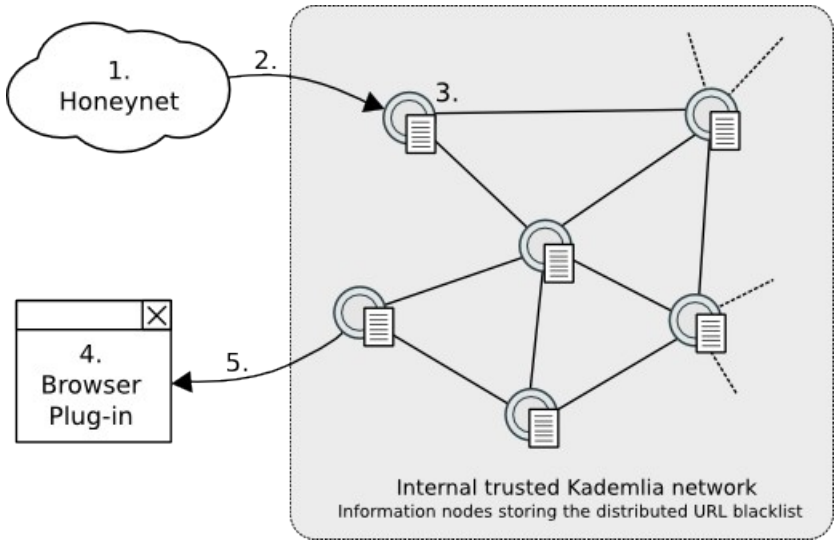

Fig. 1: The client honeynet crawls the world wide web for malicious websites (1) and sends the results to an info node (2) which stores an according entry in the network's DHT (3). When a user attempts to access the site (4), the browser plug-in queries the network (5) and either allows or denies to open it after receiving the result.

2.1. Core System

In contrast to the classic client/server model, the proposed system design uses a peer-to-peer network to reduce the load of single systems and to be more resistant against denial-of-service attacks, or general failures.

The network consists of so called *information nodes* (info nodes) that store the URL blacklist as a distributed hash table (DHT), i.e. each node only holds a small part of the blacklist but is able to locate and retrieve any requested entry. The network design is based on a modified version of the peer-to-peer protocol *Kademlia* and extends it to a fully encrypted public key infrastructure, called *KadS*. The communication between nodes or with outside clients such as the honeynet and the browser plug-in can hence be highly encrypted. To achieve this, each info node is equipped with a distinct key-pair from which its node ID derives and which is signed by a CA certificate known to all clients and nodes. Every node with a signed key-pair is part of the *trusted* internal KadS network and has full access to the DHT.

The proposed KadS network is almost identical to the Kademlia network, i.e. it basically consists of four remote procedure calls (RPC): PING, STORE, FIND_NODE and FIND_VALUE. The major extension to the protocol is that every communication is synchronously encrypted with a session key which is created and exchanged in a handshake procedure when two nodes first meet. That is, a node has a different session key for the connection with each node it knows in the

network. The handshake is similar to the one used in other software except for the fact that both clients exchange their public key and need to verify each others identity. After a successful handshake, two nodes are able to exchange lightweight encrypted messages using the session key and the basic protocol applies [Kad02].

As mentioned above, the actual information about malicious URLs is stored as distributed hash table and contains an entry for every domain analysed by the client honeynet. To find information about a domain inside the network, an info node has to query the network for the according hash key of the domain (e.g. the SHA1 checksum of the domain-string) using the FIND_VALUE remote procedure call. Once the entry has been found and is returned to the information node, it can be forwarded to the client that requested it. An entry mainly consists of the following information:

- Domain: The domain name for this analysis entry.
- Analysis Code: An integer code to quickly identify the analysis status of the domain, e.g.
  - 0 → Nothing known about the domain.
  - 1 → The whole domain seems to be clean, i.e. no reported infection.
  - 2 → Partial domain infection; the attached list shows which paths are affected.
  - 3 → The whole domain has been flagged as *malicious*.
- Path List: An optional list of domain paths or URLs that define it as *clean* or *malicious*, including the expiry date of the entry.

2.2. Interfaces

Both browser plug-in and honeyclients require to interact with the core system. For this purpose, a distinct interface and protocol for each of them has to be designed.

The browser plug-in needs to be able to query the network for vulnerable URLs of a domain before it allows the user to access the site. For this purpose, it connects to one or more information nodes and creates a secure connection. This handshake between the *untrusted* client and the *trusted* information node could for instance be initiated when the browser starts. Almost like a handshake from a browser to a secure web-server, the client plug-in verifies the identity of the information node and creates the session key to use for future messages. A possible request/response cycle from the plug-in's perspective for a query on *example.com* could look like the following listing, assuming that N is an information node and B is the browser plug-in (in this example unencrypted and omitting the handshake to secure the connection):

```
B → N   QUERY       example.com
B ← N   RESULT      domain = example.com
                    analysis code = 2 (partial domain infection)
                    path list =
                        (malicious, expires 25/3/09, bad.example.com/xy.html),
                        (clean, expires 2/3/09, example.com/clean.html)
                        [...]
```

The client honeynet is permanently crawling the World Wide Web to find malicious websites and therefore needs an interface to store the results inside the KadS network. Just like the browser

plug-in, a honeyclient also needs to establish a secure connection to one or more information nodes. In contrast to the browser, it requires to identify itself to the information nodes because it attempts to store or update information about a domain or URL. Hence it seems useful to either also equip honeyclients with a key-pair signed by the CA certificate, or to introduce a different authentication mechanism so that each honeyclient can prove that it is allowed to perform CRUD (Create, Read, Update, Delete) operations on the DHT of the KadS network.

3. Scope of work

The thesis will split in three main parts. First, it will concentrate on the idea of the proposed system and discuss the general approach of the blacklisting service by looking into existing similar solutions and the used technology to implement them. This particularly includes the comparison of peer-to-peer solutions to classic client/server models and the discussion about advantages and disadvantages of both approaches with special focus on the proposed service. Furthermore, it will introduce the Kademlia protocol, look into its strengths and weaknesses and describe the required changes to be made for the URL blacklisting service.

The second part of the thesis will focus on the implementation of the service, how the design specification has been realised and which parts had to be added or changed. Furthermore, it will explain the resulting code and pull together all parts of the system.

The last part will review the results and discuss potential improvements such as performance or other applications of the system or protocol.

A. References

[Kad02] Kademlia: A Peer-to-peer Information System based on the XOR Metric
Original protocol definition, New York University, 2002