Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Semester Thesis

Design & Implementation of a Rewriting Forward Proxy

March 22, 2005

David Fuchs
fuchsd@student.ethz.ch

Supervision:
Michael Näf

Overview

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Introduction
- The Information Security Lab
- Purpose
- Basic Principle

Design & Specification
- First Approach
- Cryptographic Primitives
- Architecture
- URL rewriting
- Deficiencies

Implementation
- Existing Software
- Details of the underlying Proxy Software
- Plugin Design

Results
- Security Considerations
- Performance
- Future Work

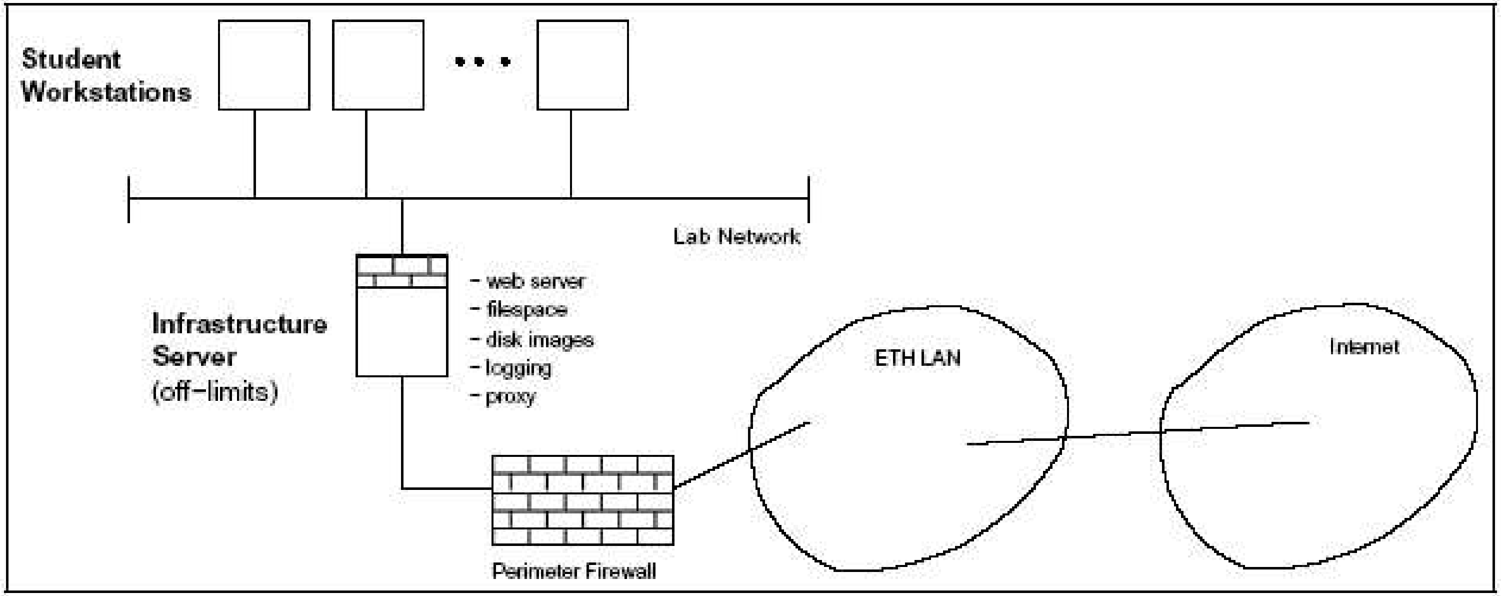
Q&A

Introduction

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

The Information Security Lab

- 11 linux workstations, 1 infrastructure server
- connected to the regular ETH LAN
- traffic from and to the lab reduced to the necessary minimum
- about 1500 HTTP requests per hour
- infrastructure overview:



Introduction

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Purpose

- many "dangerous" tools and techniques are utilized in the lab
  - example 1: crash a web server with specially crafted HTTP requests
  - example 2: inject undesired content into a back-end database with SQL injections
- due to malicious or careless operation, damage might be caused to external resources...
- but basic internet access is an important requirement for the course students.

➔ protect external resources from (unintended) malicious activities!!

Introduction

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Basic Principle

➔ no unvalidated client-provided information (query strings, header fields, ...) is forwarded to an external resource

– access is only granted to “known” URLs that have been identified on previously requested HTML pages, or URLs contained in a whitelist.

– passing data via GET or POST is only allowed for selected resources, and data is validated in a strict manner before being sent to an external resource.

Overview

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Introduction
- The Information Security Lab
- Purpose
- Basic Principle

Design & Specification
- First Approach
- Cryptographic Primitives
- Architecture
- URL rewriting
- Deficiencies

Implementation
- Existing Software
- Details of the underlying Proxy Software
- Plugin Design

Results
- Security Considerations
- Performance
- Future Work

Q&A

Design & Specification

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

First Approach

- use an internal data structure to store URLs “seen” so far...
- ...and rewrite URLs to locate them in the data structure when they are requested.
- downsides:
  - is re-writing the URLs really necessary...?
  - memory usage just grows and grows... some sort of garbage collection is required...
  - ... but then, requests may be wrongly denied because the respective URL has already been collected. Only an oracle could prevent this.
  - this is especially a problem for bookmarks
  - restarts even completely destroy the list

➔ rewrite URLs so that fact that proxy has seen them is stored as part of the URL!

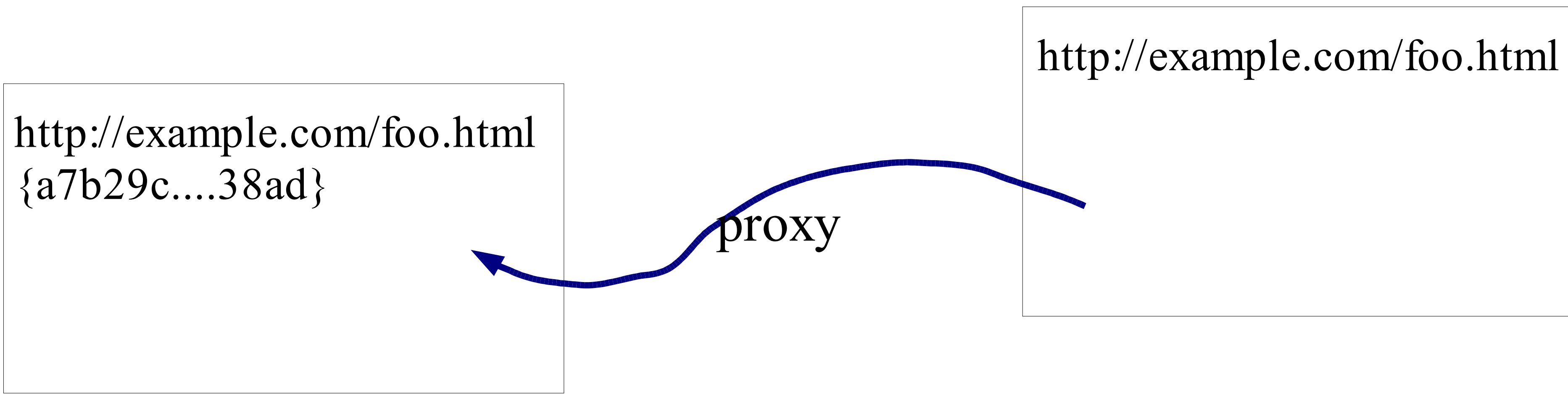
Design & Specification

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Cryptogarphic Primitives

- a HMAC generated by a keyed hash function guarantees message authenticity and integrity

➔ use HMAC tickets to flag URLs as “known”!



- when the URL is later requested, the proxy checks correctness of the attached HMAC ticket.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Architecture (1)

- Tickets: All URLs identified in a fetched HTML document are tagged.
- Rule List: A list of URLs where access is always granted or denied, regardless of the existence of a ticket.
- rule format:
  - URL(s) the rule will be applied to
  - target (allow or deny)
  - name and description for informational purposes
  - optional set of parameters as input for the Data Handlers (see next slide)

Design & Specification

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Architecture (2)

- Data Handlers: For selected resources, clients are allowed to pass values. For those resources, Client Data Handlers identify, parse, and validate client-provided data.
- allowed client-provided data is described by parameters of a rule:
  - parameter name
  - legal range for the according value
  - HTTP method (GET or POST)
  - an optional “required” attribute if the parameter is to be present.
  - ➔ particularly useful for x-www-urlencoded data, less suited or not applicable for other formats

Design & Specification

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Architecture (3)

– sample Rule for Google:

```
<rule target="allow">
   <name>google searches</name>
   <description>this rule allows to search
        google.</description>
   <url>http://www.google.ch/search</url>
   <param method="get">
      <pname>hl</pname>
      <pvalue>^(en|de|fr|it)$</pvalue>
   </param>
   <param method="get" required="required">
      <pname>q</pname>
      <pvalue>^[a-zA-Z0-9\+%]+$</pvalue>
   </param>
   ...
</rule>
```

Architecture (4)

– proxy actions: the first matching rule will be applied.

1. if a matching denying rule is found, the request is blocked.
2. if a matching allowing rule is found, the request is forwarded. The rule's client data handler ensures that client-provided data matches the parameter set defined for the rule. If present, a ticket attached to the URL will be removed.
3. if a valid ticket is present, the request is forwarded. The ticket is removed.
4. the request is blocked.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Architecture (5)

- header fields also contain client-provided information, and may cause harm to an external resource
- possible actions for header fields:
  - forward the header as-is without any checks. This should generally be avoided since it violates the basic rule, but is sometimes required by the RFC.
  - forward the field after having checked its correctness. Correctness can be checked semantically or syntactically.
  - replace the field value provided by the client by a pre-defined default value. This makes sense for values that are known in advance.
  - remove the header field. This is reasonable for fields that are not required by the external resource to fulfil a request.
- trade-off between RFC compliance and requirements...

Design & Specification

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

URL rewriting

– URL format:

```
"http://" <host>[":"<port>] "/" [<path>]
    ["?"<searchstring>]["#"<reference>]
```

– rewritten URL format:

```
"http://" <host>[":"<port>] "/" [<path>]
    ["?"<searchstring>] "{"<ticket>"}" ["#"<reference>]
```

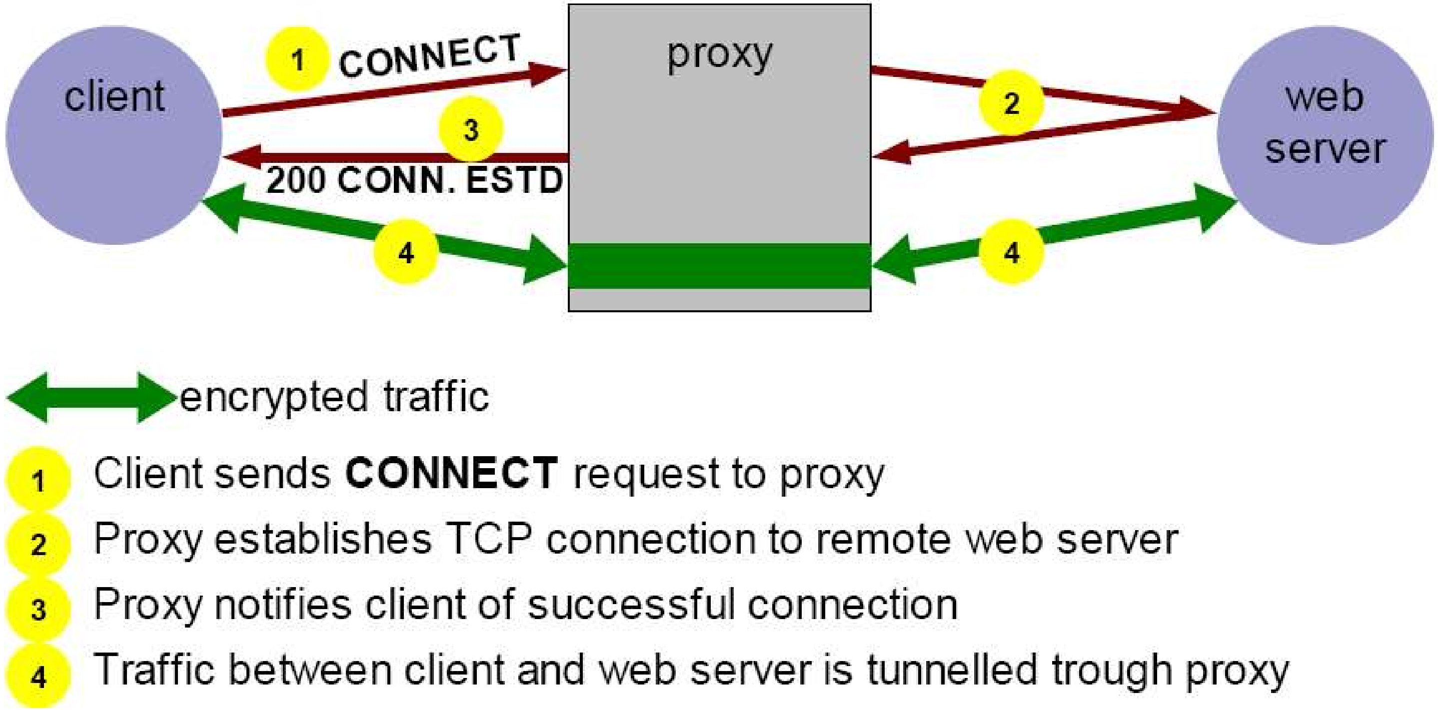
– “{“ and “}” have to be encoded. Other marker strings or even no markers could also be used. Using markers is convenient since not all hash functions produce output of the same lenth.

Design & Specification

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Deficiencies (1)

- HTTPS with “normal” proxies: the CONNECT mechanism
  - arbitrary TCP connections can be tunneled!
  - parsing of incoming HTML documents not possible!

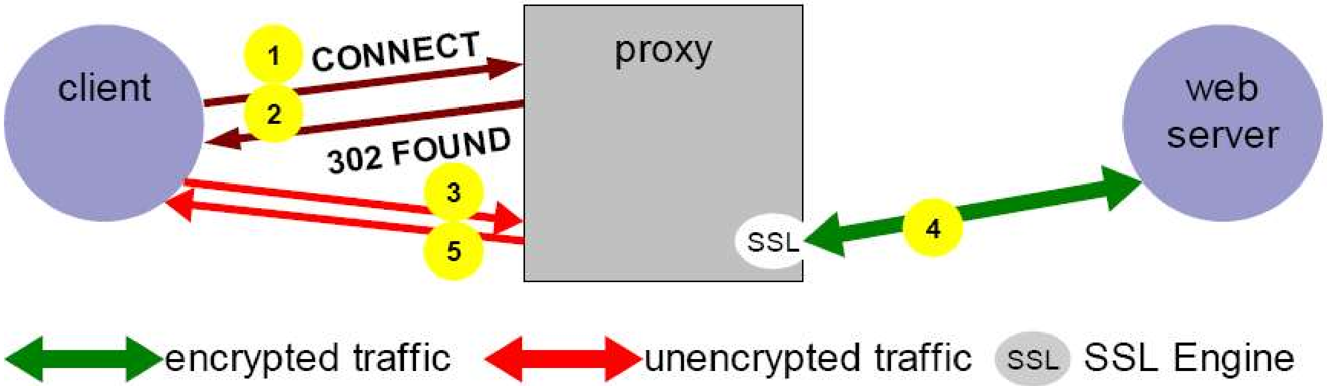


- solutions on following slides were not implemented for reasons of time.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Deficiencies (2)

- HTTPS solution 1: HTTPS between proxy and server
  - connection between client and proxy not secure
  - user agents may refuse Redirections for CONNECT requests
  - not user friendly



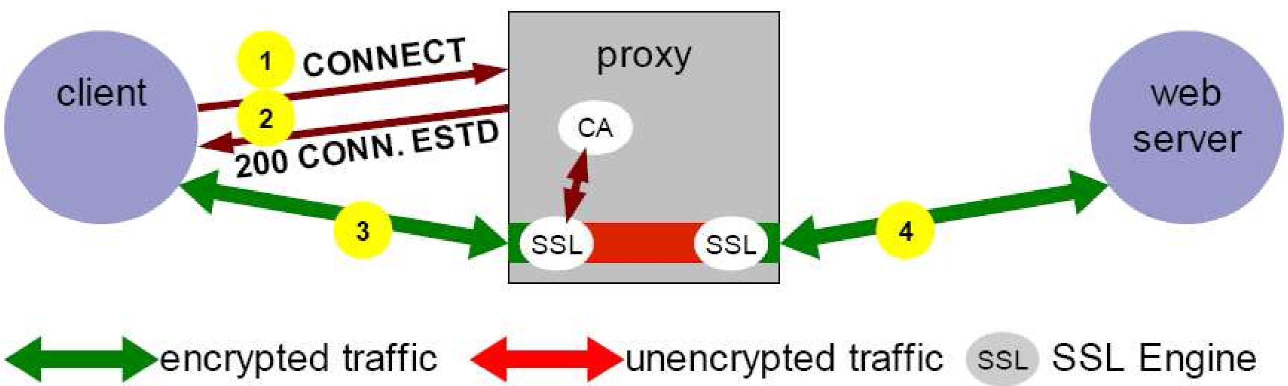
1. Client sends **CONNECT** request to proxy
2. Proxy answers with a **REDIRECT** message
3. User enters HTTPS URL into a form which is submitted to the proxy via insecure HTTP
4. Proxy establishes secure connection to remote web server and forwards request as with normal HTTP.
5. The web server's reply is decrypted and sent to the client via insecure HTTP.

Design & Specification

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Deficiencies (3)

- HTTPS solution 2: Split secure connection at proxy
  - Man-in-the-Middle-Attack! Can be made transparent for client with Certification Authority as part of the proxy.
  - no secure end-to-end semantics!
  - expensive cryptographic operations (DoS attacks...)



1. Client sends **CONNECT** request to proxy
2. Proxy notifies client of successful connection
3. Client establishes secure connection to proxy instead of remote web server and sends request
4. Proxy establishes secure connection to remote web server and forwards request as with normal HTTP. Data from both directions is decrypted, processed and encrypted again

Design & Specification

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Deficiencies (4)

- proxying FTP: FTP uses an URL scheme very similar to HTTP
- many HTTP proxies support “pseudo-FTP” where FTP requests from the client are sent to the proxy via HTTP, and the FTP-server's replies are translated into HTML by the proxy. This mechanism could be extended to include the HMAC tickets in the translated HTML reply.
- but this is not “real” FTP! Web browsers will understand it, but most FTP clients won't.
- I therefore suggest the use of a secure FTP proxy. If FTP access to external resources is not needed, the FTP protocol can be blocked entirely.

Design & Specification

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Deficiencies (5)

- missing links: the proxy may fail to identify links on an incoming page for several reasons:
  - invalid HTML syntax, e.g. missing double qoutes around the HREF element. Example for horrible HTML: Google. This can be handled to some degree by fault-tolerant parsing.
  - URLs not contained in source but generated dynamically, e.g. by JavaScript. This seems almost impossible to handle, but these cases are seldom.
- direct access to web pages not possible, form submission not possible
  - not design flaws, this follows from the requirements

Overview

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Introduction
- The Information Security Lab
- Purpose
- Basic Principle

Design & Specification
- First Approach
- Cryptographic Primitives
- Architecture
- URL rewriting
- Deficiencies

Implementation
- Existing Software
- Details of the underlying Proxy Software
- Plugin Design

Results
- Security Considerations
- Performance
- Future Work

Q&A

Implementation

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Existing Software

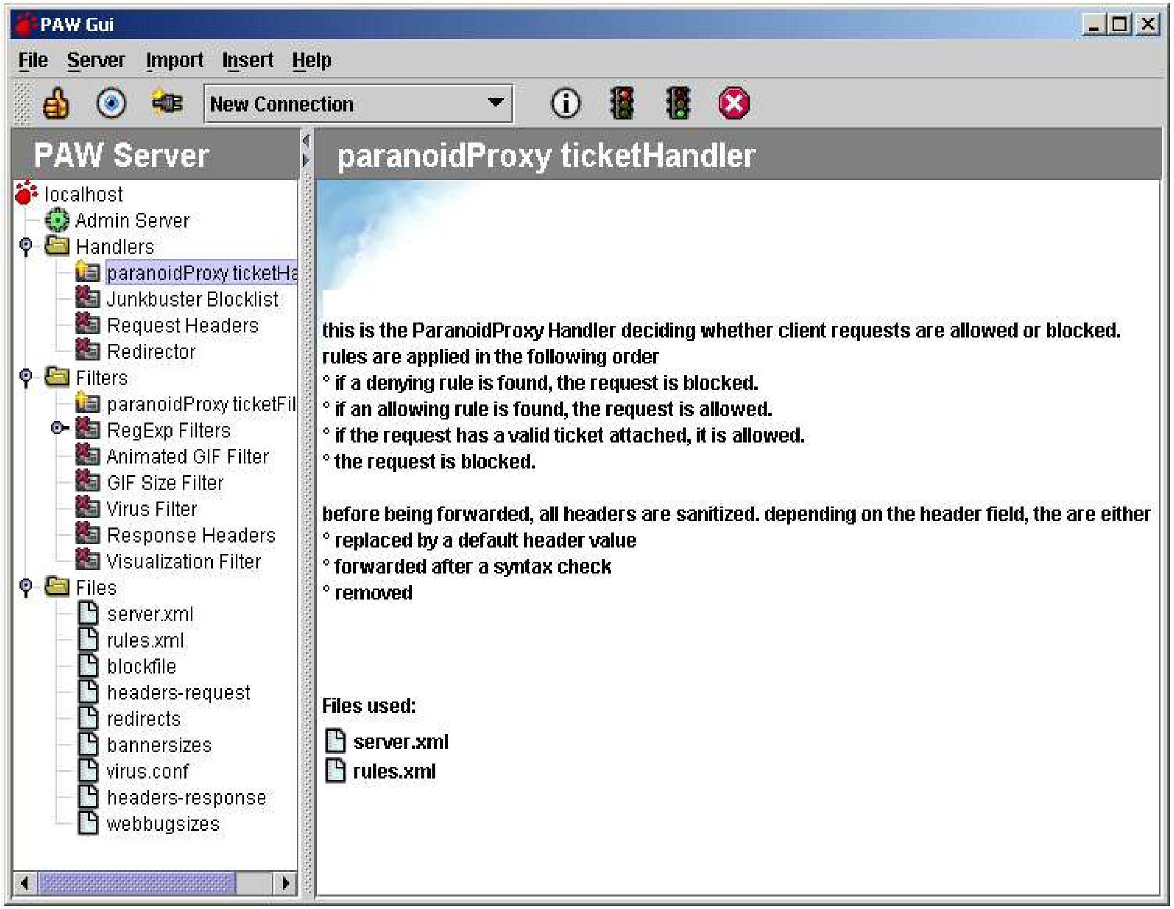
- requirements in order of assigned importance:
  - flexibility and Modularity of the Software
  - code Safety: type safety, garbage collection, etc.
  - documentation
  - stability and Performance
  - license and Source: product should be free and, if possible, open-source
  - ease of use
  - portability
- I began with evaluating Apache's mod_proxy and Squid...
- ...but soon realized that understanding and altering the C-based source will be very tedious and time-consuming
- Java on the other hand offers object oriented programming, type safety, automatic garbage collection...

Implementation

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Details of the underlying Software (1)

– PAW: Pro-Active Web Proxy
  + easy extension through plug-in classes
  + well documented
  + written entirely in Java
  + GNU GPL licensed
  + Administration via GUI or XML config files
  - Not widely used, so bugs might go un-noticed for a long time



Implementation

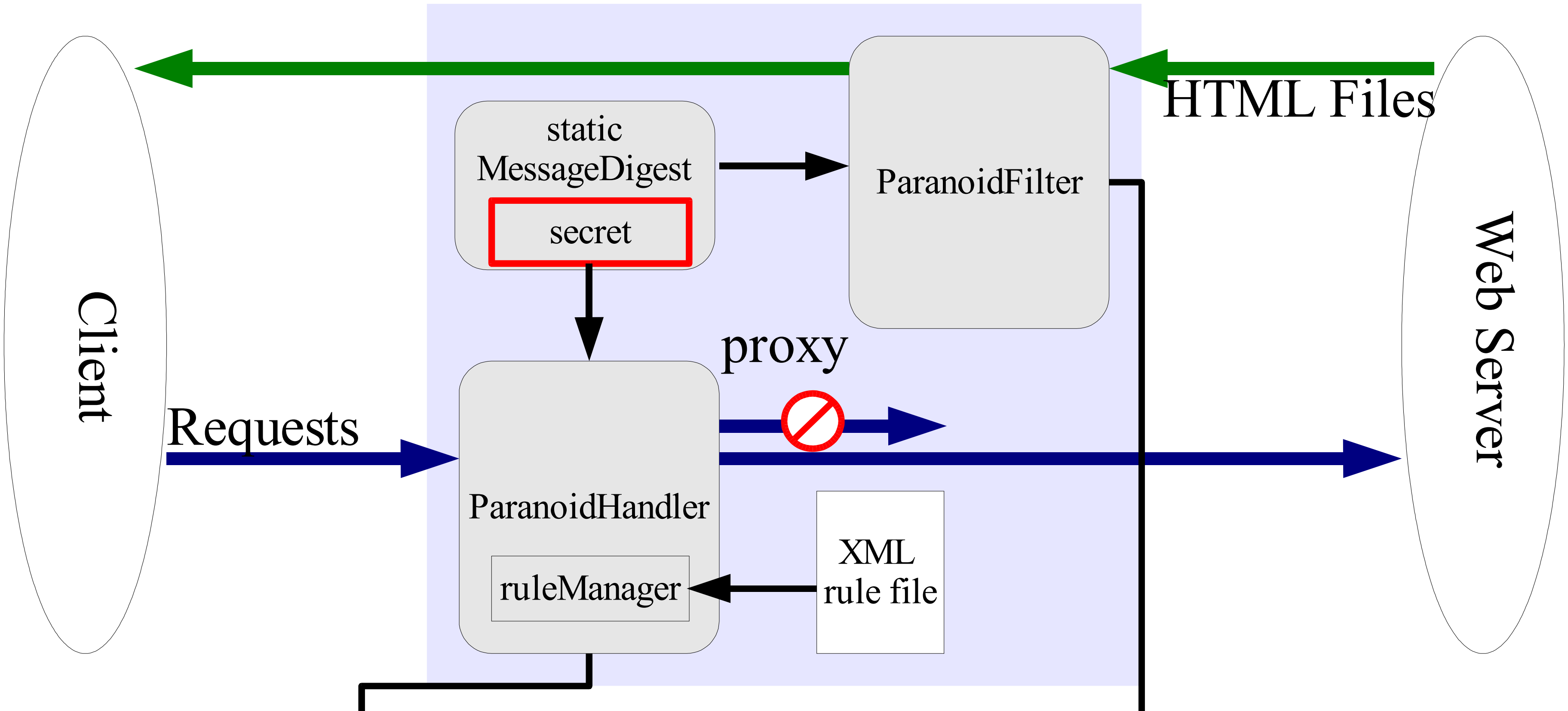
Details of the underlying Software (2)

- PAW is based on SUN's Brazil Framework
- Brazil began as an URL based interface for smartcards
  - evolved into a complete HTTP stack with a small footprint
  - completely in Java
  - two most important entities: Server and Request
  - Server handles Requests with plug-in Handler classes
    → very flexible architecture
  - many Handlers are already provided
  - Filters are a special case of Handlers used to modify content

Implementation

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Plugin Design



- grants or denies requests based on rules and presence of tickets
- sanitizes headers
- strips tickets off requests

- parses incoming HTML files
- adds ticket to identified URLs
- uses regExp for parsing

Overview

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Introduction
- The Information Security Lab
- Purpose
- Basic Principle

Design & Specification
- First Approach
- Cryptographic Primitives
- Architecture
- URL rewriting
- Deficiencies

Implementation
- Existing Software
- Details of the underlying Proxy Software
- Plugin Design

Results
- Security Considerations
- Performance
- Future Work

Q&A

Results

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Security Considerations (1)

- mainly designed to prevent unintended malicious activity
  - does not prevent a user from requesting a “dangerous” URL already present on an external web page
  - does not protect users from malicious content on requested pages
- Brute Force Attacks:
  - cracking secret key allows access to arbitrary URLs.
  - attacker can generate many input/output pairs for hash function
  - but “good” hash functions are “secure” even if many input/output pairs are known
  - periodical changes of secret would reduce amount of known pairs but also impact functionality

Security Considerations (2)

- stealing the secret key:
  - in principle, all handlers and filters have access to the config file and hence, the secret key
  - they could even disable the ParanoidProxy handlers by editiing the config file

➔ only handlers and filters from trusted sources should be installed

- reflecting URLs
  - example: user enters <a href="some malicous url"> as user name in a login form, and the web site replies, user <a href="some malicious url"> is not known, try again. The URL is recognized by the proxy and tagged with a ticket.

➔ allowed parameter values should be reasoned, e.g., '<>' not allowed

Results

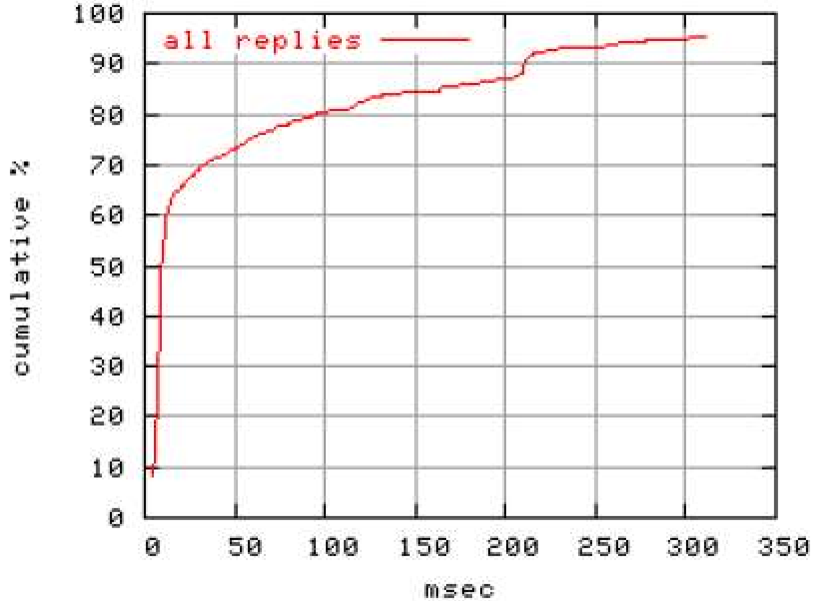
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Performance (1)

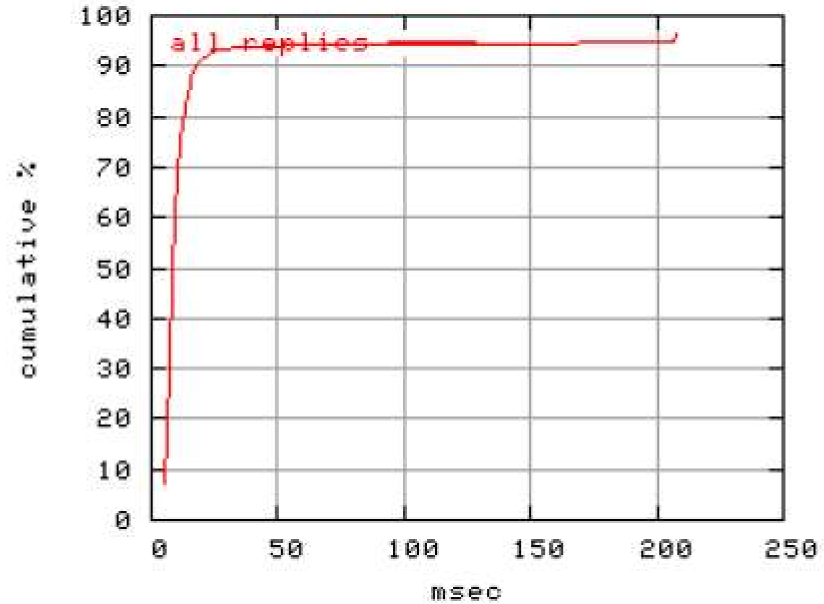
- Web Polygraph was used for performance measurements
  - allows to model lots of assumptions
  - simulates HTML content (important when testing parsers etc.)
- different test cases:
  - realistic model with low traffic volumes (0.6 requests/s), small files, and mixed content
  - large HTML files to measure response time increase introduced by parsing
  - best effort method
- slow-down was distinctly measurable in all test cases
  - response times increased by factor 3 in first case (now ~60 msec)
  - factor can go up to 20 or above for large files
  - throughput reduced by factor 3.5 in best effort mode (now ~4 MBit/s)

Results

Performance (2)



response time distribution
filter on (first test case)



response time distribution,
filter off (first test case)

- conclusion: performance is sufficient for use in lab, but should be improved for use in larger environments
  - implementation: code optimizations
  - design: inclusion of a cache for rewritten pages

Results

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Future Work

- integration of SSL support
- integration of a cache for improved performance
- more generic rule list
  - currently on a per-URL-basis
  - no wildcards allowed
  - no IP-address based rules
- easier Administration
  - rule administration currently done by editing XML files
  - rule list not loaded if rule file has syntax errors
  - helper application could write rule file based on inputs to a GUI

Overview

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Information Security Group
Prof. Dr. David Basin

Introduction

- The Information Security Lab
- Purpose
- Basic Principle

Design & Specification

- First Approach
- Cryptographic Primitives
- Architecture
- URL rewriting
- Deficiencies

Implementation

- Existing Software
- Details of the underlying Proxy Software
- Plugin Design

Results

- Security Considerations
- Performance
- Future Work

Q&A

Questions?

– report, software, javadoc and additional material available @
  http://n.ethz.ch/~fuchsd/proxy/

– thanks for your attention!