

# HoneySpider Network

Fighting client side threats



**HONEYSPIDER**  
*network*

Piotr Kijewski (NASK/CERT Polska)

Carol Overes (GOVCERT.NL)

Rogier Spoor (SURFnet)

# Outline

- > Honeyclient overview
- > What & Why  
HoneySpider Network ?
- > Goals
- > Threat focus
- > Project overview & status
- > Technical concept
- > Wrap up



# Honeyclient overview

# What is a Honeyclient ? (I)



## Definition:

*Honeyclients are active security devices in search of malicious servers that attack clients. The honeyclient poses as a client and interacts with the server to examine whether an attack has occurred.*

Source:

[http://en.wikipedia.org/wiki/Client\\_honeypot/\\_/\\_honeyclient](http://en.wikipedia.org/wiki/Client_honeypot/_/_honeyclient)

# What is a Honeyclient ? (II)

- > Different honeyclients depending on level of interaction:
  1. Low interaction honeyclients
  2. High interaction honeyclients

# Low Interaction Honeyclient

- > Light weight or simulated clients (web crawler)
- > Identifies known attacks based on:
  - Static analyses
  - Signatures
- > May fail to emulate vulnerabilities in client applications
- > Tools:
  - HoneyC
  - SpyBye
  - PhoneyC

# High Interaction Honeyclient

- > Fully functional operating system with vulnerable applications (browsers, plugins)
- > Detection of known/unknown attacks via comparison of different states (before and after visit of a server)
- > Slow & prone to detection evasion
- > Tools:
  - Capture-HPC
  - MITRE Honeyclient
  - HoneyMonkey

# What & Why HoneySpider Network?

# Honeyclient project – What?

- > Joint venture between SURFnet, NASK and GOVCERT.NL.
- > Development of a complete system, based on low- and high-interaction honeyclient components.
- > To detect, identify and describe threats that infect computers through Web browser technology.

# Honeyclient project – Why?

- > Attack vector has shifted:
  - Number of browser exploits increased last years.
  - Massive compromises of vulnerable websites which redirect to malware.
  - (Obfuscated) Java- & VB-scripts used as vehicle to serve exploits. *(examples coming up in a minute)*
- > Better understanding client side threats.
- > Provide a service to constituents.

# Honeyclient project – Why? (II)

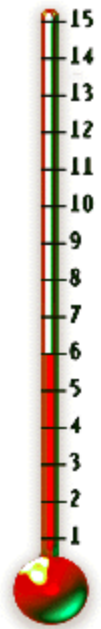
- > Existing honeyclient solutions don't meet our requirements, regarding:
  - Integration & management
  - Stability & maturity
  - Limited heuristics
  - Stealth technology
  - Self-learning

# Goals

- > Build a stable and mature system, capable of processing bulk volume of URL's.
- > Detect and identify URL's which serve malicious content.
- > Detect, identify and describe threats that infect computers through browser technology, such as:
  - Browser (0)-day exploits
  - Malware offered via drive-by-downloads

# Project overview

- > Completed functional & technical requirements
- > Organized project management
- > Software development started in September 2007
- > Project will be finished begin 2009



# Project status

Milestone	Work item	Status
<b>1</b>	- <i>Low-interaction Honeyclient</i>	<i>Done</i>
<b>2</b>	- <i>Central database</i> - <i>Import URL's</i> - <i>Webinterface</i>	<i>Done</i>
<b>3</b>	- <i>High-interaction Honeyclient</i>	<i>Started</i>
<b>4</b>	- <i>Website layout</i> - <i>Integration ARAKIS / SURF IDS</i> - <i>External analyses</i>	<i>To Do</i>



# Threat Focus

# Threat focus

- > Different threats need different approaches
- > Main focus on three kinds of threats (see next slides)
- > More to come in the future. Possible options:
  - Phishing attempts
  - Email attachments (e.g. Office files)

# Threat focus 1: Drive-by Download

- > Download of malware without awareness of the user.
- > Malware offered and executed through exploitation of (multiple) vulnerabilities in browser, plugin, etc.
- > Specific vulnerabilities targeted, based on:
  - Browser (IE/Firefox)
  - Browser plugins
  - JVM versions
  - Patch level operating system



# Threat focus 2: Code obfuscation

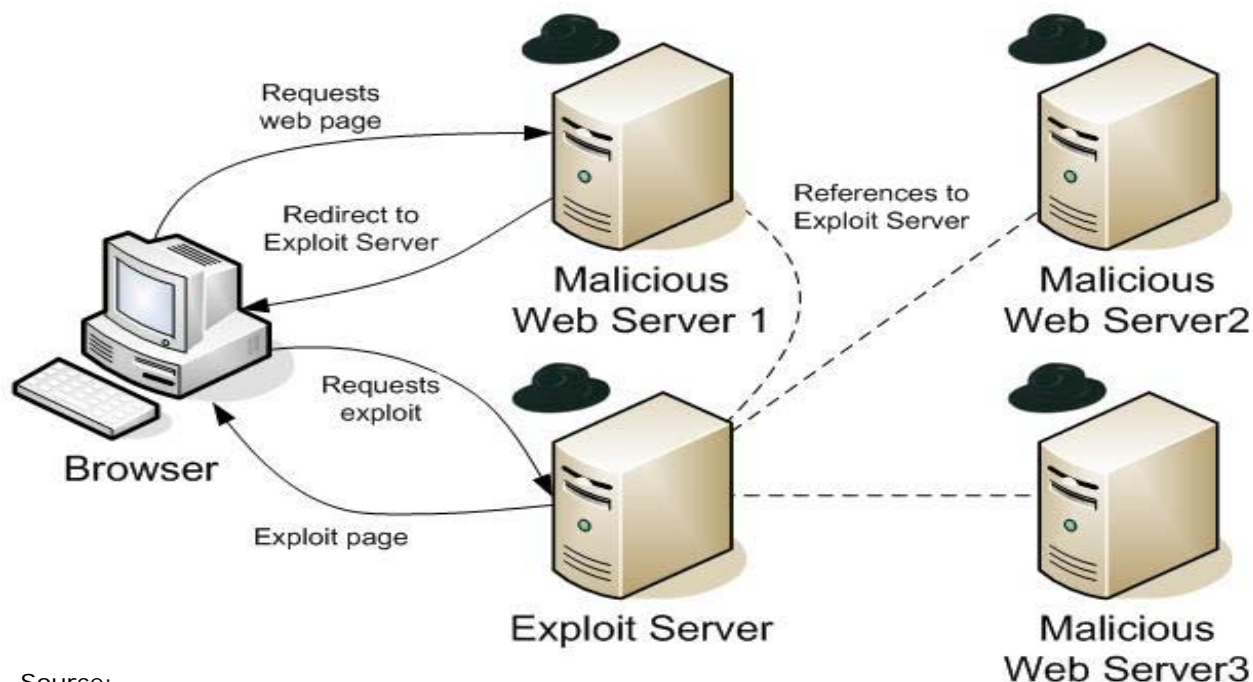
## > Code obfuscation

- Hide the exploit-vector
- Evasion of signature-based detection (AV products, Intrusion Detection Systems)
- Examples seen for Javascript, VBScript

```
function xor_str(plain_str, xor_key){
    var xored_str = "";
    for (var i = 0 ; i < plain_str.length; ++i)
        xored_str += String.fromCharCode(xor_key ^ plain_str.charCodeAt(i));
    return xored_str;
}
var plain_str =
"\xf6\xdb\xdc\xdb\xdc\xa0\xb7\xa4... \xff\xed\xdb\xdc\xdb\xdc";
var xored_str = xor_str(plain_str, 214);
eval(xored_str);
```

# Threat focus 3: Compromised websites

Exploits imported from other servers via iframes, redirects, Javascript client side redirects

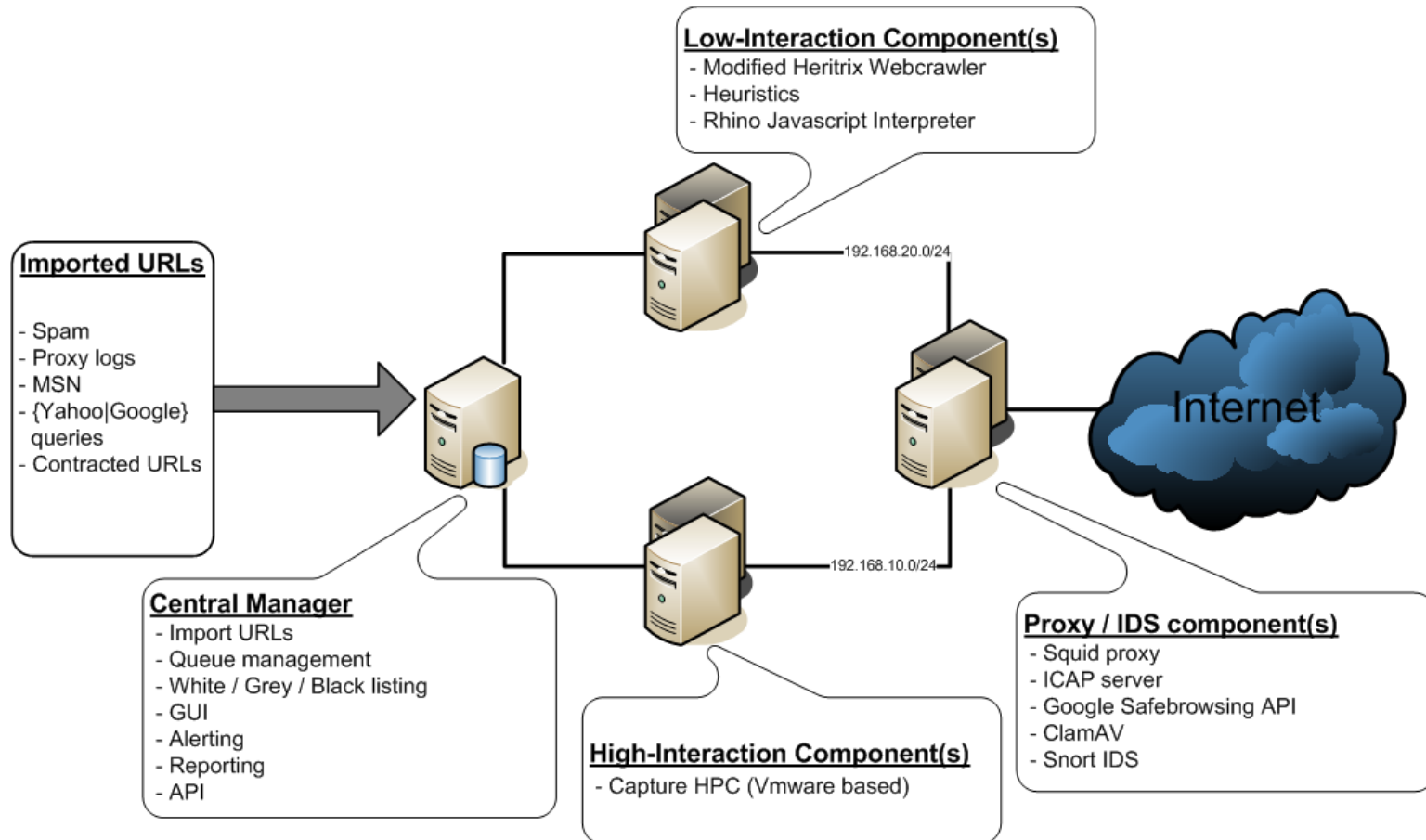


Source:

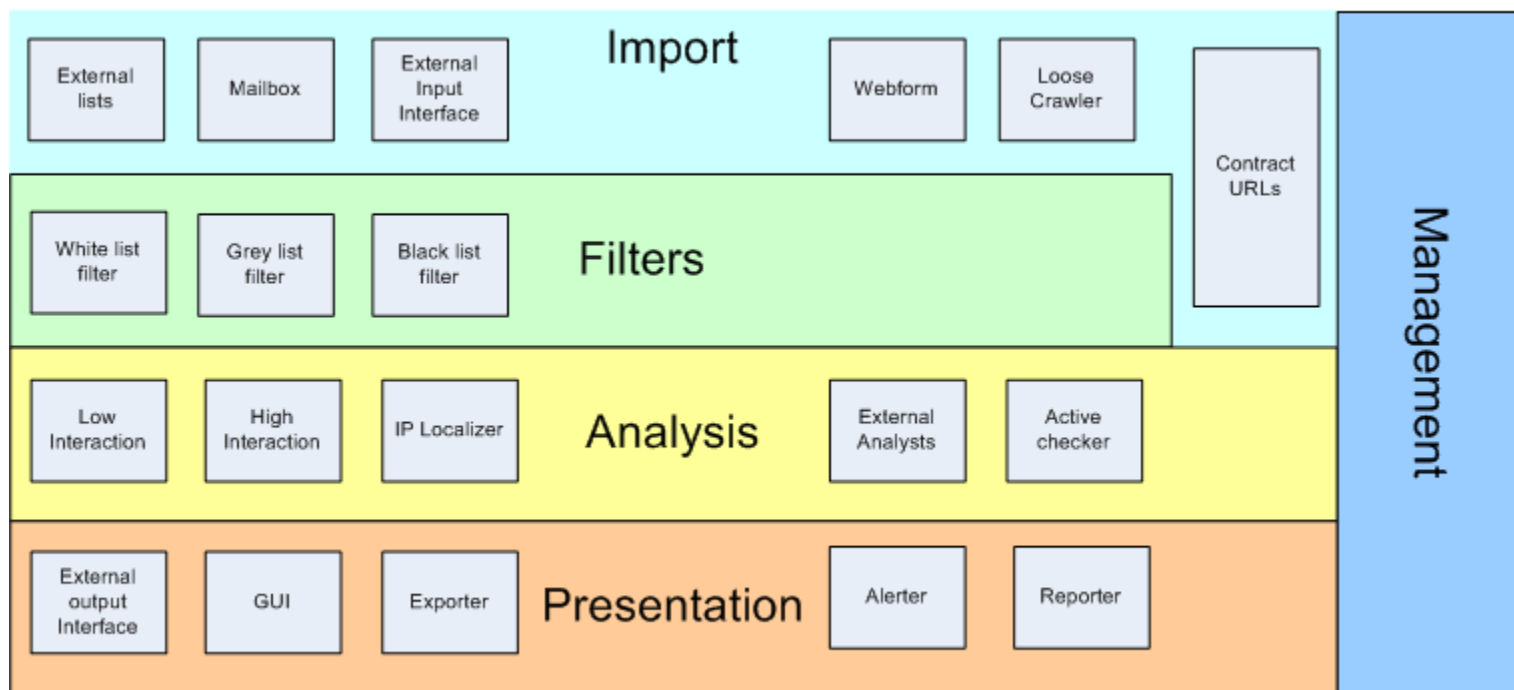
[http://www.honeynet.org/papers/mws/KYE-Malicious\\_Web\\_Servers.htm](http://www.honeynet.org/papers/mws/KYE-Malicious_Web_Servers.htm)

# Technical concept

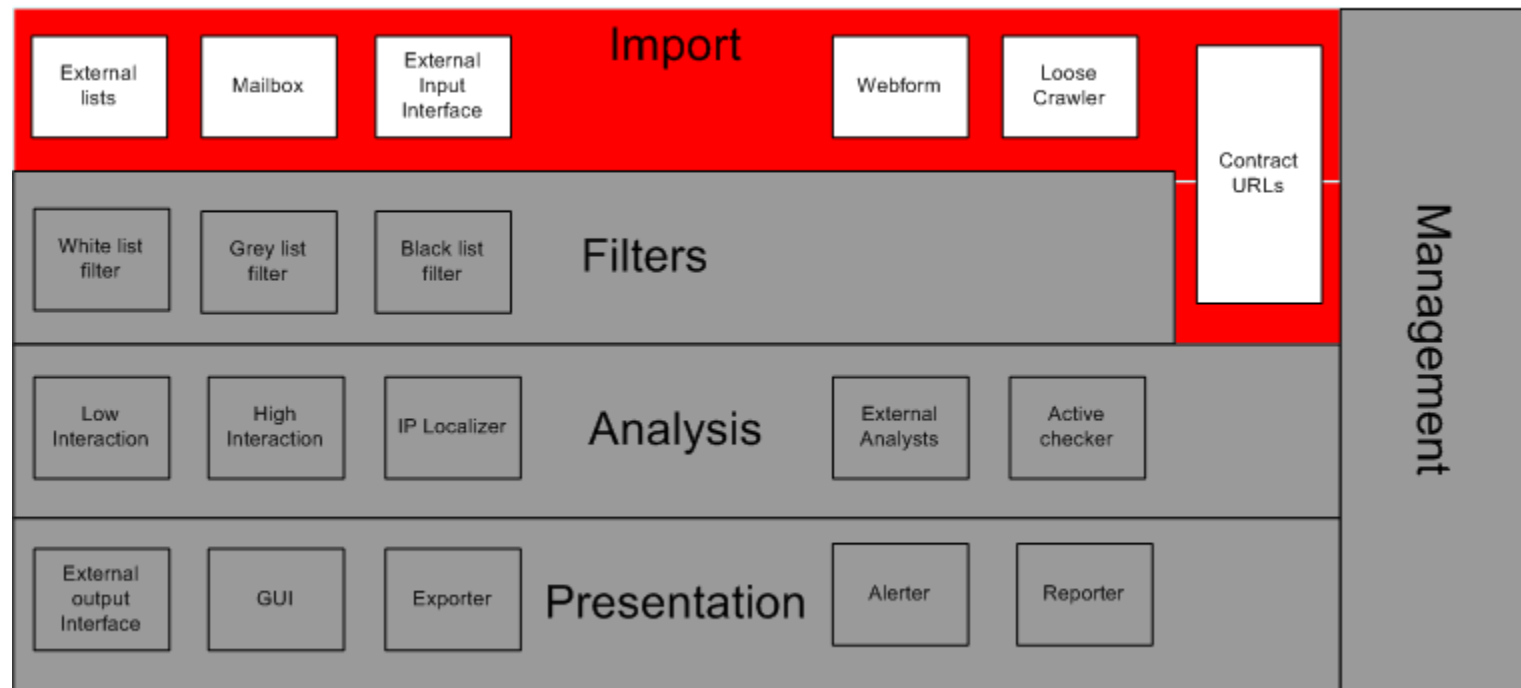
# Architecture



# Technical concept



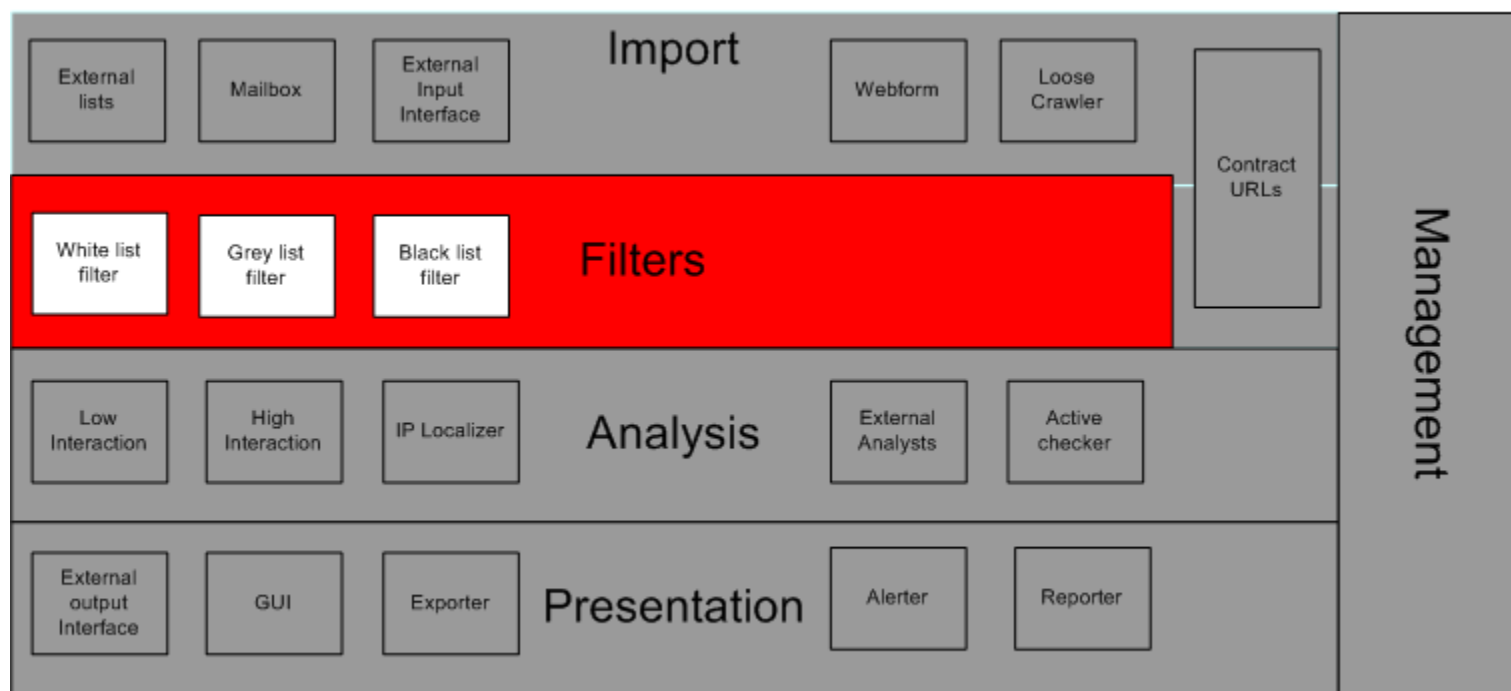
# Import layer



# Import layer

- > URL's (aka objects) imported via:
  - Mailbox (POP)
  - File inclusion
  - HTTP(S) (pull method)
  - Webform
  - {Google|Yahoo}-queries
- > URL's prioritized based on importance / origin
- > Contracted URLs:
  - Important URLs which need to be checked
  - frequently (sites of constituents / customers)

# Filter layer



25-9-2008

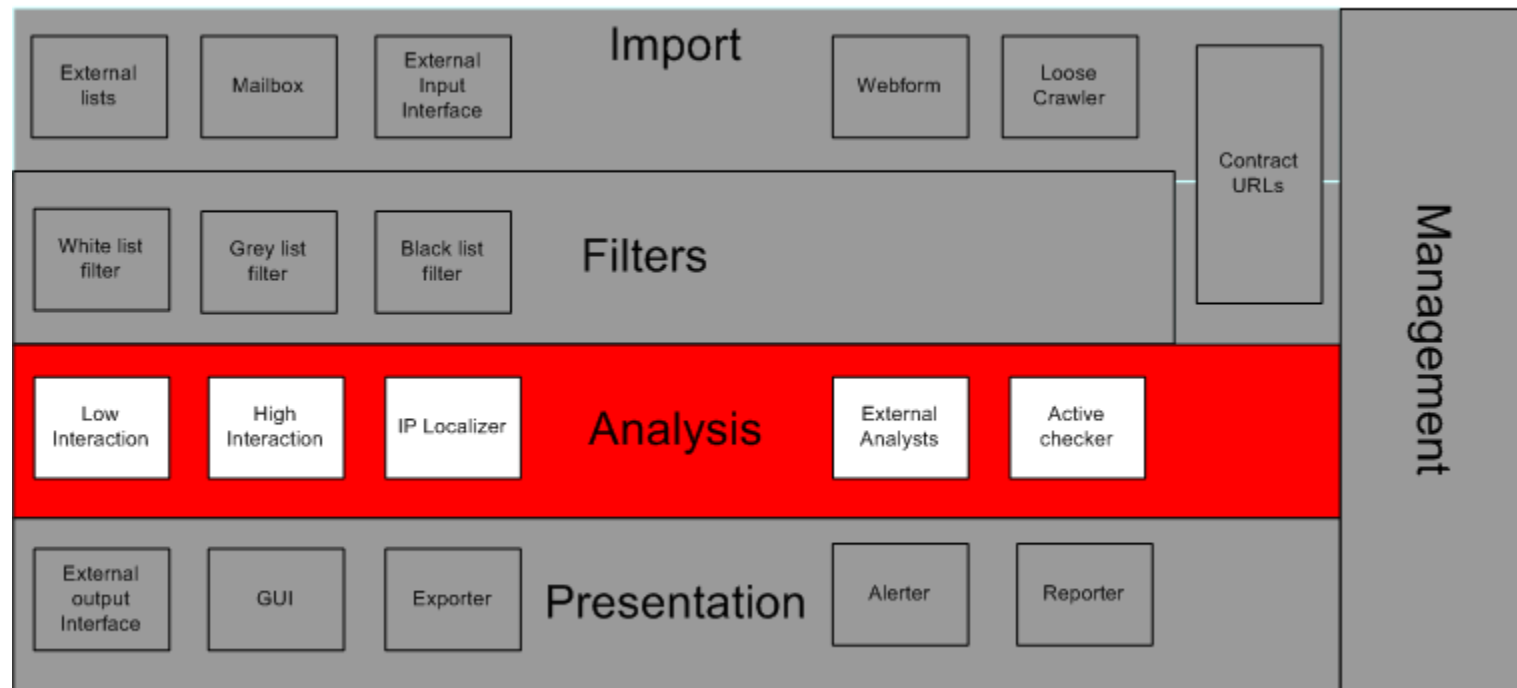
The HoneySpider Network -  
Fighting client side threats



# Filter layer

- > Filter already analyzed & unreachable URL's
  - Applies on all URLs, except contracted URLs
- > Filter lists:
  - White: URL's classified **benign**
  - Grey: URL's classified *suspicious*
  - Black: URL's classified **malicious**
- > Hit count & TTL (or permanent) on every listed URL

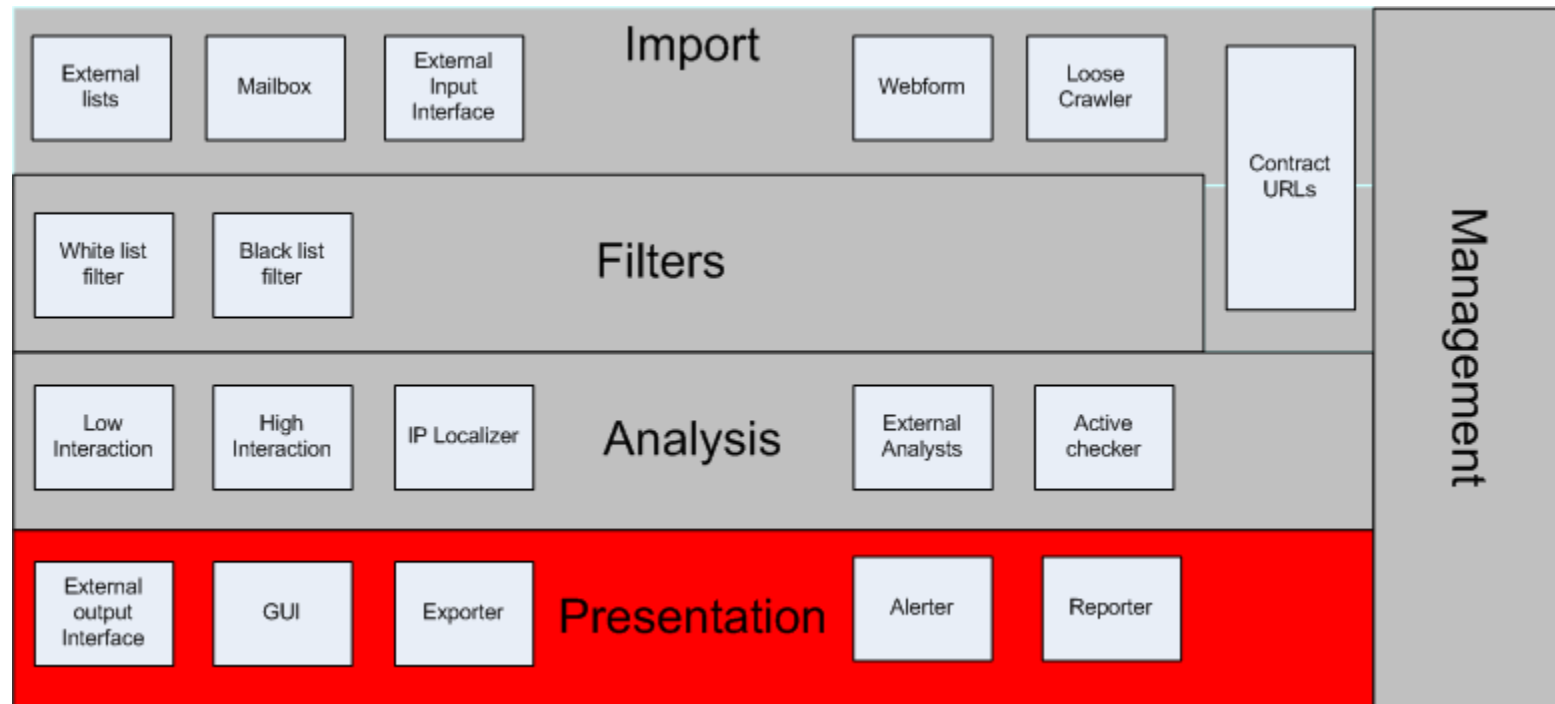
# Analysis layer



# Analysis layer

- > {Low, high}-interaction components  
*(see upcoming slides)*
- > External analysis of malware or URL
- > Plugins for:
  - VirusTotal
  - Anubis
  - Norman Sandbox
  - CW Sandbox
  - Stopbadware
- > Results stored in database
- > Storage {ISP, ASN, Country} information

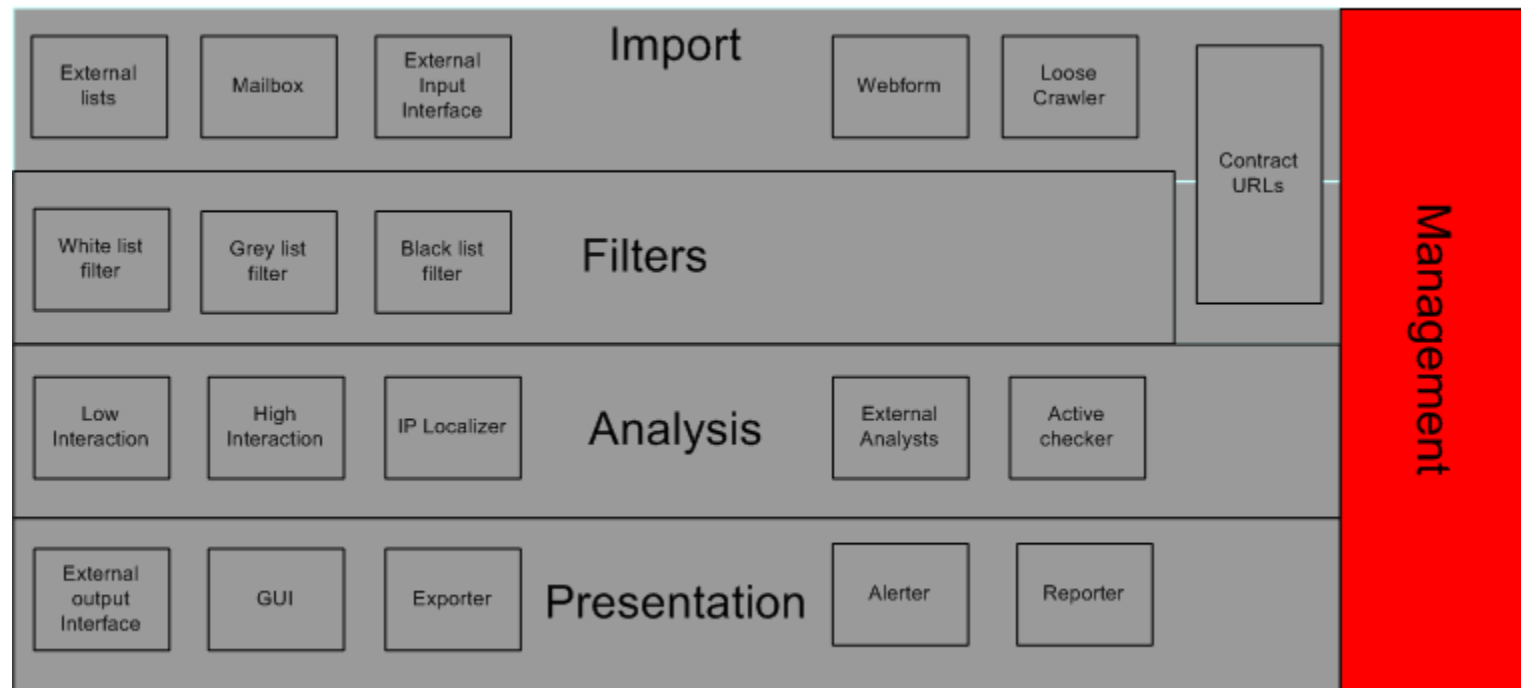
# Presentation layer



# Presentation layer

- > Web-based GUI
- > Alerter plugin
  - Sends alerts via email, SMS
- > Reporter plugin
  - Creates reports (PDF) with graphical statistics and/or detailed information
- > External output plugin
  - External systems can fetch results of processed objects

# Management layer



# Management layer

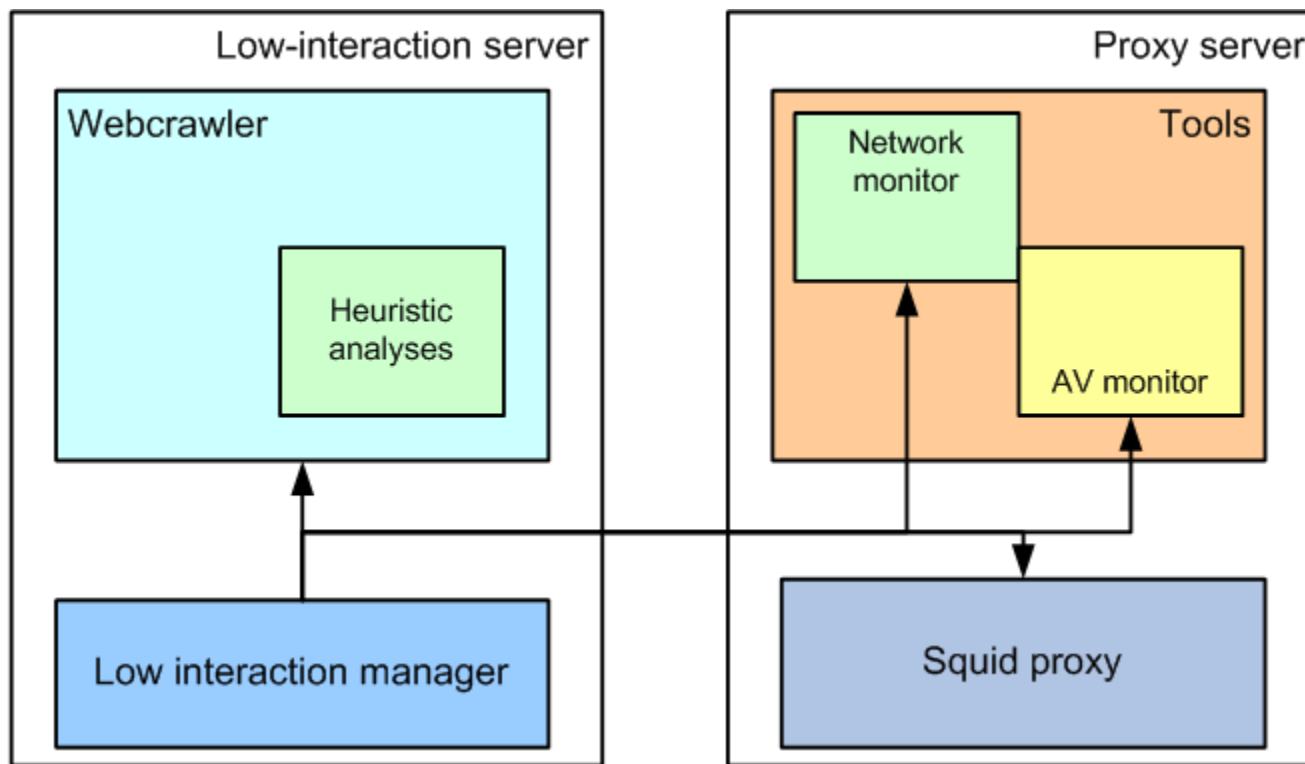
- > Objects tagging
  - Confidence level
  - Priority level
  - Process classification
  - Alert classification
- > Queue manager
  - Manages the main object-queue
- > Signature manager
  - Generation of signatures
  - Judge quality of signatures
  - Distribute signatures to {Network|AV} monitor

# Low/High - Interaction & Heuristics

# Low interaction component (I)

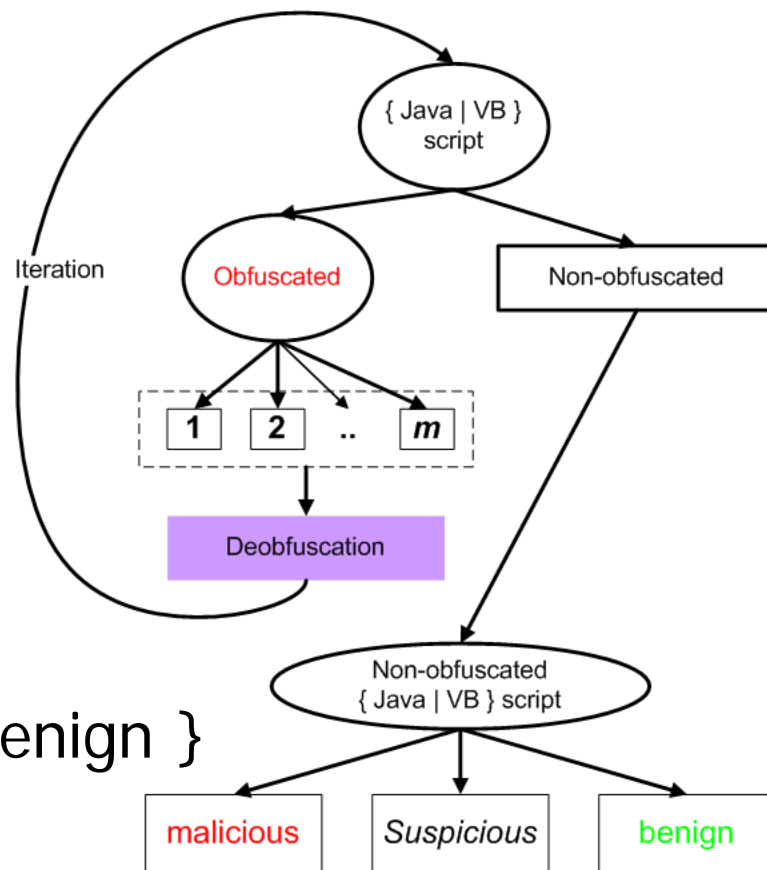
- > Webcrawler (Heritrix)
- > Rhino JavaScript interpreter
- > Heuristics
- > Google Safebrowsing API
- > Low-Interaction Manager
- > Controls & retrieves data from:
  - Webcrawler
  - Squid proxy
  - ClamAV
  - Snort IDS

# Low interaction component (II)



# Heuristics – Detection malicious scripts

- > Classification:  
Obfuscated or not?
- > If obfuscated: what  
type of obfuscation?
- > Deobfuscation
- > Classification:  
{ malicious | suspicious | benign }

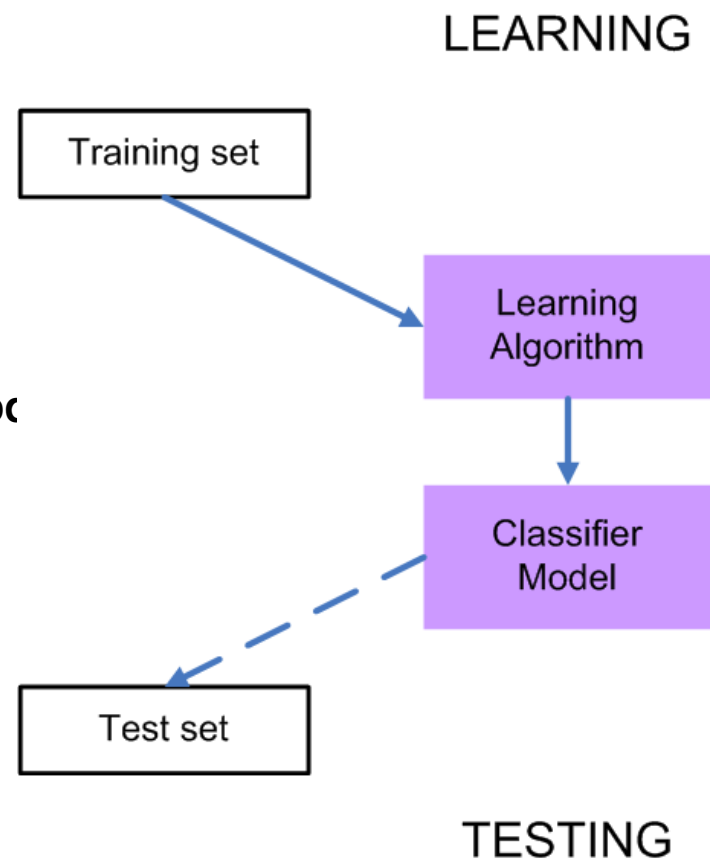


# Heuristics - Approach & goal

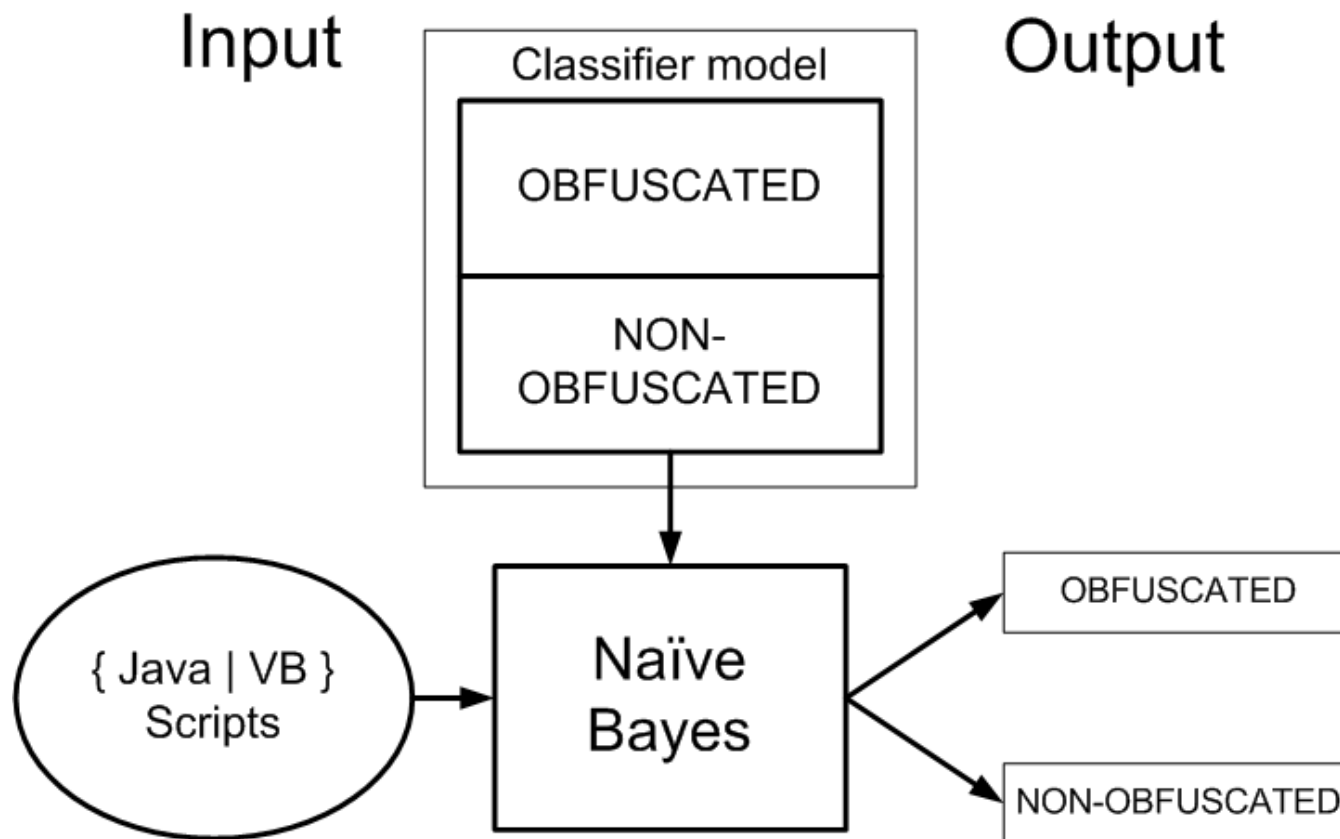
- > Approach
  - **Building** classifier models **based on** machine learning **and** data mining-based **techniques** for text classification.
- > Goal:
  - **Classification of** previously unseen {Java | VB} Scripts (i.e. assigning them to proper pre-defined categories)
- > Tool of choice:
  - **Weka - Data mining software**
  - **Google n-grams**

# Heuristics - Classifier model

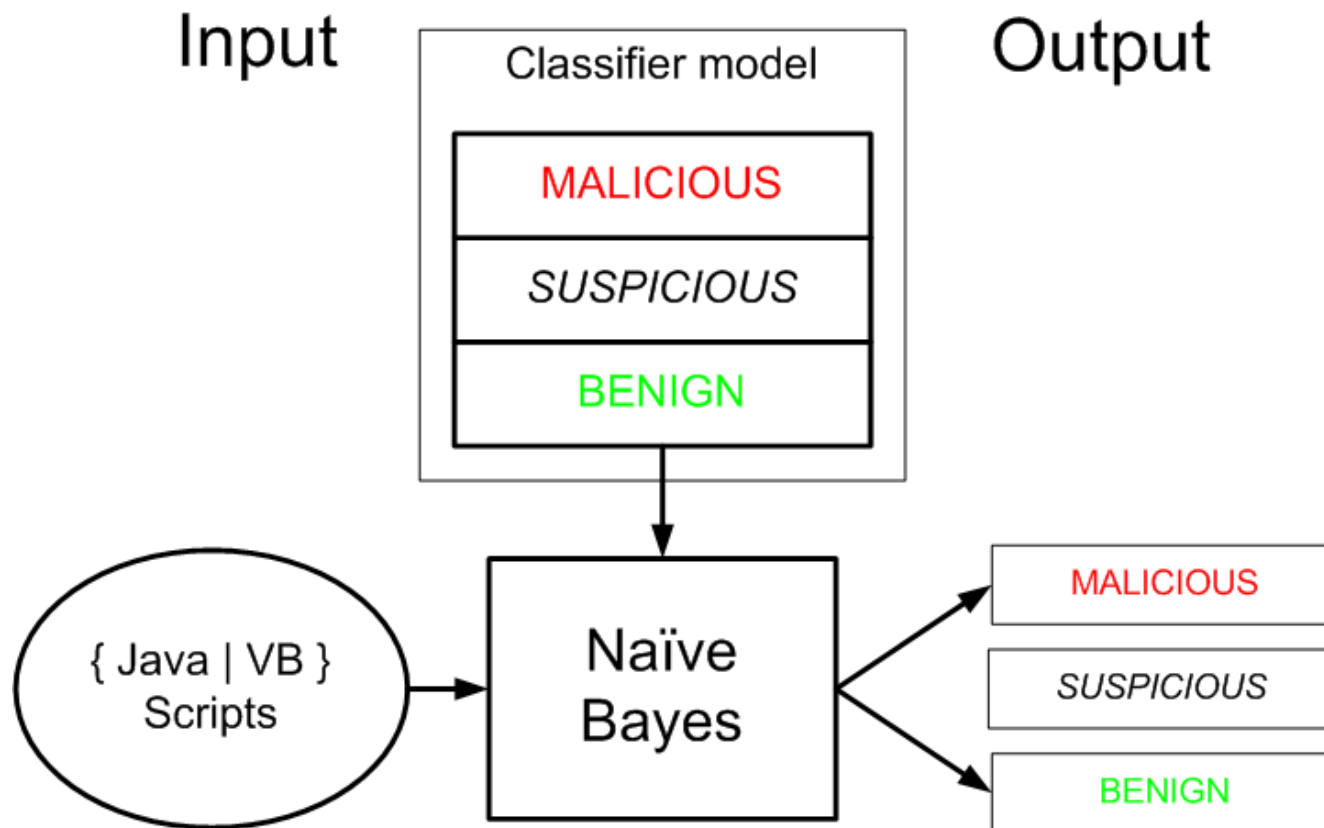
- > Training set & test set
  - Samples with a class label (e.g. 'obfuscated JS', 'non-obfuscated JS')
- > Learning with training set
  - Build a classifier model with good generalization of properties for each class
- > Testing with test set
  - Validate a classifier model (i.e. its accuracy in prediction classes of unseen items)



# Heuristics - Naïve Bayes (I)



# Heuristics - Naïve Bayes (II)



# Other implemented heuristics

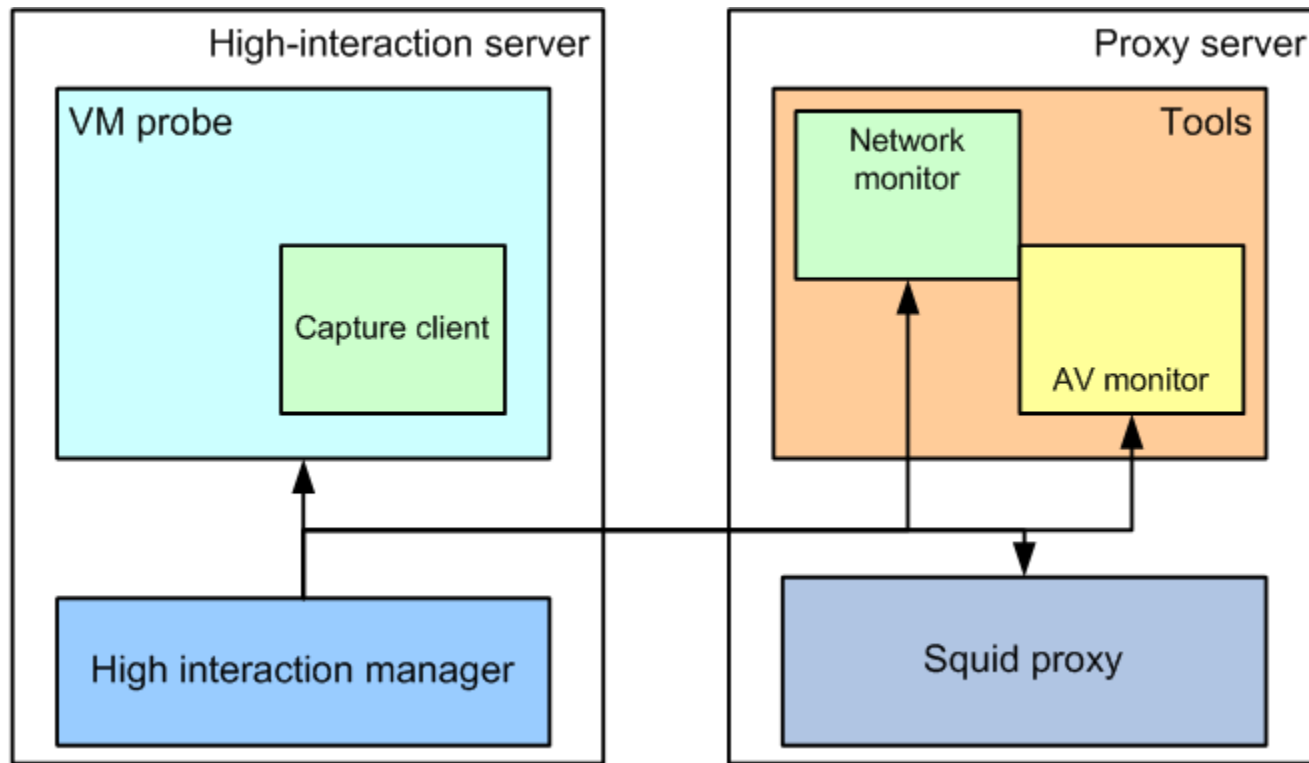
- > JSAdvancedEngineDetection
  - Triggers on behaviour interpreted differently in different browsers.
- > JSIterationCounter
  - Triggers when output of a Rhino iteration results in an obfuscated JavaScript.
- > JSExecutionTimeout
  - Triggers when Rhino hangs during execution of a JavaScript.
- > JSOutOfMemoryError
  - Triggers when Rhino starts to allocate excessive amount of memory when processing JavaScript.

# High interaction component (I)

- > Based on Capture-HPC
- > Multiple patch levels Microsoft Windows
- > IE / Firefox (possibly plugins, like QuickTime & Flash)
- > Checks for:
  - Started or terminated processes
  - Filesystem modifications
  - Registry modifications
- > Proxy (Squid) with ClamAV
- > Google Safebrowsing API
- > Snort IDS
- > Pcap dumps



# High interaction component (II)



# Wrap up

## **HoneySpider Network project:**

- ✓ To identify suspicious and malicious URLs
- ✓ A combination of low- & high-interaction honeyclients
- ✓ Many URLs from multiple sources processed based on importance

# Links

- > HoneySpider Network
  - <http://www honeyspider.org/>
- > Capture HPC
  - <https://projects.honeynet.org/capture-hpc/>
- > Weka
  - <http://www.cs.waikato.ac.nz/ml/weka/>
- > Google n-grams
  - <http://code.google.com/p/ngrams/>
- > Heritrix
  - <http://crawler.archive.org/>



# Acknowledgements

## > NASK

- Juliusz Brzostek
- Krzysztof Fabjanski
- Tomasz Grudziecki
- Marcin Koszut
- Adam Kozakiewicz
- Tomasz Kruk
- Elzbieta Nowicka
- Cezary Rzewuski
- Slawomir Suliga

## > SURFnet

- Wim Biemolt
- Kees Trippelvitz

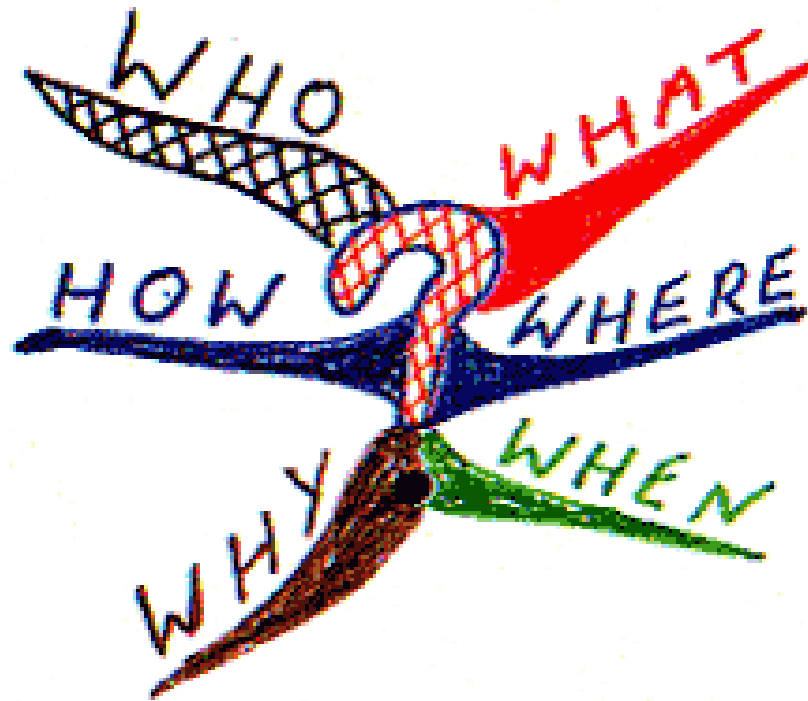
## > GOVCERT.NL

- Jeroen van Os
- Menno Muller

## > Qnet Labs

- Bas Sisseren

# Questions ?



25-9-2008

The HoneySpider Network -  
Fighting client side threats

