THE MODERN ROGUE – MALWARE WITH A FACE
Hamish O’Dea
Microsoft Pty Ltd, Level 5, 4 Freshwater Place, Southbank VIC 3006, Australia
Email hamisho@microsoft.com

ABSTRACT
Over the past year we have seen a significant increase in reports of the type of malware commonly known as rogue security products, or simply ‘rogues’. These programs, which display false alerts of system infection and ask for payment to ‘clean’ the system, have been around for years; however they have recently become more cunning, more sophisticated and more prevalent.

This paper examines what has changed in the rogue landscape in recent times and compares their evolution to that of other types of malware. We look at the ways in which rogues are similar to other malware, from their distribution to the methods they use to evade detection and how they react to large-scale elimination by Windows Defender and the Malicious Software Removal Tool. We also examine what makes rogues unique and how they extend social engineering techniques beyond the point of getting the malware onto the system through to the user’s interaction with the malware itself and beyond. We look at how rogues deal with the distinct challenges of having a recognizable brand and the ways they take advantage of a user’s trust in their computing platform, from the operating system to the browser and even the search engine they use.

By analysing rogues in the same way as we look at other types of malware, we get a better idea of how they fit into the overall threat landscape. The rogue is usually the end product of a malware infection scenario – the final payload. As opposed to spam bots, backdoors or password stealers, rogues try to obtain money directly from the user. A rogue differs from most malware only in that it has a face.

ROGUE DISTRIBUTION
When looking at individual malware components it can be difficult to see their purpose in the overall plan of whoever wrote them. Given that the overwhelming majority of malware is written for profit, it stands to reason that every individual malicious program has some purpose towards this goal. When analysing the Bagle botnet in 2005, we observed that every individual component served at least one of these functions:

- Expansion
- Survival
- Exploitation

This model can be applied to any system that relies on computer compromise by malware. The malware must be distributed in some way (usually the more machines, the better), it must survive on the target systems (at least long enough to execute its payload) and it must exploit that target to ultimately benefit its creator. Any type of malware that is particularly prevalent is likely to be very effective in all of these areas. Before looking at how rogue security scanners carry out these functions, the first question is: are rogues prevalent?

PREVALENCE
Win32/Renos is a generic name we use at Microsoft for rogue downloaders. Some simply download and install a rogue silently, while others display bogus warnings of malware infection or system compromise before prompting to install a rogue security scanner. Measuring the prevalence of Win32/Renos can give us a good idea of the impact that rogues are having.

Win32/Renos was the most prevalent malware family in the second half of 2008 (SIR). It was found by Microsoft security tools – including the Malicious Software Removal Tool (MSRT) and Windows Defender – on more than 4.3 million computers between July and December 2008. The second most prevalent family was found on approximately 3.7 million computers. Not only was Win32/Renos top by a large margin, but the number two family was Win32/Zipjob, which is also known to download rogues.

In addition to Win32/Renos, rogues themselves were some of the highest reported threats over the same period. Two rogues appeared in the top ten families: Win32/FakeXPA at position seven with almost 1.7 million computers and Win32/FakeSecSen at number eight with about 1.5 million computers.

How have rogues become so prevalent? None of them self-propagate (at this time), so how do they get onto so many machines? As is commonly the case with modern malware, there are several different ways. The following are some of the most notable.

BLACKHAT SEO
Search engine optimization, or SEO, is a term used to describe techniques for increasing traffic to a website from search engines. Sometimes these practices are considered acceptable; those that aren’t are often called ‘blackhat SEO’ techniques. Any technique that is used to improve traffic to sites hosting content that is not relevant to the search term is blackhat SEO. Since people generally don’t go looking for malware, this is what rogue creators need to do.

There have been several well documented cases of blackhat SEO being used to distribute rogues. The techniques involved in getting a site to a high position in a search result are beyond the scope of this paper, but I will present an example to illustrate how they have been used effectively for rogues in particular.

On 2 April 2009, a popular web comic called XKCD published a comic that included the phrase ‘Higgs excitation’ [1].

The term ‘Higgs excitation’ became the 25th most searched term on 2 April, according to Google trends [2]. As

Figure 1: XKCD comic, 2 April 2009.
Windows is the top target for malware due to being the dominant desktop operating system, Google search is the top target for malware when it comes to search engines. In this particular case, there was very little information about the term ‘Higgs excitation’ on the web; the majority of the search results were not related to the term at all.

The idea is that a search engine’s spider will see the doorway page, but a person following a link from a search result is directed to a different page either through server-side redirection (e.g. a .htaccess file on the web server) or client-side redirection (e.g. meta refresh commands or JavaScript). Often you might see a doorway page if you enter the URL directly into your browser, rather than following the link from a search result. This is because the redirect scripts often check the HTTP referer.

For example, here is a script from a doorway page that redirects to a page distributing the rogue Win32/FakeXPA:

```javascript
var ref,i,is_se=0;
var se = new Array("google.","msn.","yahoo.",
"comcast.","aol.");
if(document.referrer)ref=document.referrer;
else ref="";
for(i=0;i<5;i++)
{
if(ref.indexOf("site:")!=-1) {}
else {if(ref.indexOf(se[i])>=0)
is_se=1;
if (navigator.userAgent.indexOf("Firefox")!=-1)
document.write('<div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background-color: #FFFFFF; padding: 0px"><iframe src="http://macrosoftwarego.com/go.php?id=2022&key=4c69e59ac&p=1" width="100%" height="100%"></div>");
else if (navigator.userAgent.indexOf("Chrome")!=-1)
document.write('<div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background-color: #FFFFFF; padding: 0px"><iframe src="http://macrosoftwarego.com/go.php?id=2022&key=4c69e59ac&p=1" width="100%" height="100%"></div>");
else window.location="http://macrosoftwarego.com/go.php?id=2022&key=4c69e59ac&p=1";}
}
else {if(ref.indexOf(se[i])>=0)
is_se=1;
if (navigator.userAgent.indexOf("Firefox")!=-1)
document.write("<div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background-color: #FFFFFF; padding: 0px"><iframe src="http://macrosoftwarego.com/go.php?id=2022&key=4c69e59ac&p=1" width="100%" height="100%"></div>");
else if (navigator.userAgent.indexOf("Chrome")!=-1)
document.write("<div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background-color: #FFFFFF; padding: 0px"><iframe src="http://macrosoftwarego.com/go.php?id=2022&key=4c69e59ac&p=1" width="100%" height="100%"></div>");
else window.location="http://macrosoftwarego.com/go.php?id=2022&key=4c69e59ac&p=1";}
)
```

The script checks the referer and attempts to load a page from macrosoftwarego.com only if the referer contains a string indicating it is from a search portal. It is also interesting to note that if the browser is Firefox or Google Chrome, the script opens the page in an IFRAME, otherwise it redirects the browser by changing window.location.

The page on macrosoftwarego.com redirects again, but this time the redirection is performed by a server-side script. The script does not appear to check the referer, but does check the IP address from which the request originates. When requested from an address traceable to Microsoft, this page simply returns HTTP error 404 (not found). When retrieving the page from elsewhere, it returns HTTP 302 to redirect the requestor to http://antivirus-scanner-v1.com/?id=2022&smersh=27cda2196&back=%3DQ0xDz4MMQOM%3D0, the page that hosts the Win32/FakeXPA rogue.

In the ‘Higgs excitation’ example, some of the doorway pages redirected to a page that claimed it encountered an error displaying a video and instructed the user to download a ‘new version of video decoder’. Other doorway pages redirected to fake online scanner pages. In all cases the pages would eventually try to download a malicious executable.

From just this one search term we found installers for several different families of rogues, including Win32/FakeRean, Win32/Winwebsec, Win32/WinSpywareProtect and Win32/FakePlus.
Other types of malware are disseminated using SEO techniques – notable families include Win32/Alureon, Win32/Zlob and Win32/Koobface – but rogues are one of the most common. We will examine why later.

SPAM

Spam is still a popular method of distributing malware of various types and rogues are no exception. Some spam directs users to fake ‘video codec required’ or online scan web pages like the ones above, while other spam links directly to malicious executables. One campaign, for example, purported to link to free videos of celebrities.

This particular spam contained a link directly to malware, but the malware itself varied over time. In some cases it was a trojan called Win32/Cbeplay; in others it was Win32/Remos. Win32/Cbeplay is a trojan that receives instructions to download and execute other malware. At the time this email was sent, this usually included a spam bot such as Win32/Srizbi, Win32/Rustock or Win32/Cutwail. Often Win32/Cbeplay would also be instructed to download Win32/Remos. One way or another this particular spam campaign, along with the highly publicized ‘CNN.com Daily Top 10’ spam around the same time [4], was distributing Win32/Remos and through that a rogue called Win32/antivirusxp.

FLASH ADVERTISEMENTS

Malicious Flash advertisements, often called ‘malvertisements’, have been associated with rogues at least since early 2007 when they were used to distribute programs that called themselves ErrorSafe and Drivecleaner (variants of Win32/Winfixer) [5]. The process is quite simple – the Flash advertisements contain script to redirect to other URLs, often only after a particular trigger date.

To begin with the advertisements were typically for non-existent companies, but they soon moved to fake advertisements for real products and services, presumably in an effort to appear less suspicious. The practice has been used extensively by Win32/FakeXPA and has continued at least until May 2009 with rogues such as Win32/FakeRemoc (going by the name ‘Total Virus Protection’) and
Win32/Winwebsec employing advertisements like the one shown in Figure 8. As is the case with SEO techniques, these usually redirect to fake online scanner pages.

In December 2008 the US Federal Trade Commission (FTC) requested and received a restraining order against two companies involved with distributing rogues including Win32/Winfixer and Win32/FakeXPA. The FTC complaint alleged that the companies ‘falsely claimed that they were placing Internet advertisements on behalf of legitimate companies and organizations’ [6].

WIN32/WALEDAC AND WIN32/CONFICKER

It is not just specialist downloaders like Win32/Renos that have been known to install rogues. In fact, it seems that virtually every major malware family installs a rogue at one time or another, either directly or indirectly. This isn’t surprising as we know that rogue earnings are shared via an affiliate system [7], meaning that any malware creator can get a share of the profits. One high profile example is Win32/Waledac [8]. While we had observed Waledac downloading a rogue called Win32/FakeSpyPro before, it was reports that Waledac was being downloaded by Conficker [9] that attracted public interest. There had been much speculation as to how the Conficker botnet would be used to generate revenue – we were not surprised to see a rogue as the eventual payload.

WIN32/KOOFFACE

Win32/Koobface is a worm that spreads through social networking sites, including Facebook [10]. Win32/Koobface was removed from almost 200,000 machines when it was first added to the MSRT in March 2009 [11]. It has a variety of payloads but a common one has been to launch ‘chrome-less’ Internet Explorer windows to display fake online scanner pages that deliver rogues.

The page that Win32/Koobface loads is often the same page that SEO doorway pages redirect to; the URLs always include some kind of affiliate ID to connect the distributor to the page impression. In the same way, Win32/Koobface isn’t tied to any particular rogue. We have seen it distribute Win32/FakeXPA, Win32/Winwebsec and Win32/InternetAntivirus using these pop-ups.

Win32/Waledac and Win32/Koobface aren’t the only high profile malware families that we have seen distributing rogues. Win32/Alureon, Win32/Cutwail and Win32/Bredolab are all prevalent families that we have observed downloading rogues in addition to their other payloads.

ROGUE SURVIVAL

Unlike a bot, whose goal is to survive on a compromised host for as long as possible, a rogue’s purpose may be fulfilled in a very short time; however, avoiding detection is just as important to a rogue as to any other type of malware, especially at the time of installation.

CODE OBFUSCATION

Code obfuscation of one form or another is virtually ubiquitous in the world of malware. It describes the process of compressing, encrypting or otherwise modifying the malware’s code and data and wrapping it inside a layer of code that deciphers it at runtime. The goal is to hide the malware’s true form and purpose from someone trying to reverse engineer it and, above all, from anti-malware scanners. Code obfuscation is often combined with server-side polymorphism, where the malware binaries hosted on a malicious or compromised host are regularly updated, making the threat appear polymorphic. In some cases the malware binary may be different every time it is downloaded; in others it may only change once a day.

Code obfuscation is employed by rogues just as it is for other malware types. In fact, we often see the same obfuscators being used for rogues as we see applied to other sorts of malware. A simple example is Win32/FakeSecSen, which used commercially available software called ASProtect, commonly used by other malware such as IRC bots.

Increasingly we see custom, ‘underground’ obfuscators being used. These are generally used only for malware and give the malware creators more control, particularly when it comes to techniques for evading anti-malware scanners. The rogues Win32/InternetAntivirus and Win32/FakeSpyguard, for example, employ an obfuscator that has also been used extensively on prevalent variants of Win32/Alureon, a multi-purpose trojan.
Server-side polymorphism leads to huge numbers of distinct malicious files. Win32/FakeXPA was found in over 9,000 unique executable files in January 2009 alone.

DETECTION VS. REACTION

Using custom obfuscation wrappers enables the malware creator to more quickly adapt to detection by anti-malware scanners. Server-side polymorphism is effective against simple checksum or signature matching, but generic signatures and code emulation can ‘see through’ the obfuscation and detect the underlying malware regardless of changes to the polymorphic outer shell. We have seen custom obfuscators manually changed, very quickly, in response to generic detection. In tracking and analysing rogues, we have found that they are probably adapted more quickly and more often than any other type of malware. Moreover, we have seen several cases where these adaptations were, without doubt, targeted specifically at Microsoft’s anti-malware engine.

The reason why the makers of rogues in particular spend time and effort trying to evade detection by Microsoft’s anti-malware engine can be seen in the numbers of rogues we detect. Rogues come under the scope of Windows Defender and the MSRT. Windows Defender is included with Windows Vista and available as a free download for earlier Windows versions and MSRT is delivered to all Windows users who want it, through Microsoft Update and Windows Update. Between them, they have a very large install base and the ability to make an impact on the size of a rogue’s revenue.

None of this proves that these rogues are changed deliberately to avoid Microsoft’s anti-malware signatures; however, there are many cases where the evidence of this is extremely compelling and several where it is undeniable. Two anti-emulation tricks used by the obfuscator utilized by Win32/InternetAntivirus (as well as Win32/FakeSpyguard, Win32/Alureon and others) are particularly notable.

On 23 March 2009 we discovered a variant of Win32/InternetAntivirus that called the Win32 API function GetVolumeInformation() in order to query the serial number of the C: drive. The following is a fragment of disassembly from that sample:

```assembly
0041965D push '\:\C'
00419662 push 0 ; DWORD nFileSystemNameSize
00419664 push 0 ; LPTSTR lpFileSystemNameBuffer
00419666 push 0 ; LPDWORD lpFileSystemFlags
00419668 push 0 ; LPDWORD lpMaximumComponentLength
0041966A push dword ptr [ebp-0Ch] ; LPDWORD lpVolumeSerialNumber
0041966D push 0 ; DWORD nVolumeNameSize
0041966F push 0 ; LPTSTR lpVolumeNameBuffer
00419671 push esp ; LPCSTR lpRootPathName
00419672 add byte ptr [esp], 1Ch
00419676 push esp ; LPCTSTR lpRootPathName
00419677 push esp ; LPCTSTR lpRootPathName
00419678 add byte ptr [esp], 1Ch
... (Sample SHA1: 0x02feb5f90a768b391f1669080c6342f4fa27394c)
```

The chances that the serial number would match the value this code is checking for (0x7C812ED4) when running in the wild on a real system are virtually none, therefore execution would continue normally and the malware body would be decrypted and executed. When running inside our anti-malware engine’s emulator, however, the value did match 0x7C812ED4. This caused execution to take a different path, leading to an exception; the malware body was not decrypted and we did not detect the file. Interestingly, we were not setting the value to 0x7C812ED4 deliberately. This was the number that was pointed to by lpVolumeSerialNumber before the function was called and our implementation of GetVolumeInformation() did not change it.

To counter this, we implemented a quick fix to our GetVolumeInformation() implementation which set the data pointed to by lpVolumeSerialNumber to a fixed value. The number has no real significance, so I chose the serial number of the C: drive on my own lab computer, which happened to be 0xf49ca14e. The signature with this change was released at approximately 1:30 PM, 23 March (GMT). At 10:02 PM, 23 March (GMT) we received another sample of Win32/InternetAntivirus, which contained slightly different code:

```assembly
00419663 push '\:\C'
00419668 push 0 ; DWORD nFileSystemNameSize
0041966A push 0 ; LPTSTR lpFileSystemNameBuffer
0041966C push 0 ; LPDWORD lpFileSystemFlags
0041966E push 0 ; LPDWORD lpFileSystemFlags
0041966F push 0 ; DWORD nFileSystemNameSize
00419671 push esp ; LPCSTR lpRootPathName
00419672 add byte ptr [esp], 1Ch
00419676 push esp ; LPCTSTR lpRootPathName
00419677 push esp ; LPCTSTR lpRootPathName
00419678 add byte ptr [esp], 1Ch
0041967C call eax ; KERNEL32.DLL:VolumeInformationA
0041967E add esp, 4
00419681 mov eax, [ebp-14h]
00419684 cmp eax, 0F49CA14Eh
00419689 ja short continue
0041968B cmp eax, 0F49CA14Eh
00419690 jb short continue
00419692 int 3 ; Unhandled exception
00419693 continue:
00419693 pop eax
... (Sample SHA1: 0xfab5efa048d9c106af3eb6ebad86df286ba163f)
```

The code was now checking specifically for the value returned by our engine – the one I had chosen from my lab computer only hours earlier. We immediately changed our implementation again, employing a variable serial number (essentially random) so that the value could not be used to identify our emulator. The next variant of the malware used a different anti-emulation technique.

The same obfuscator has also used the current directory, specifically the length of the current directory string, to
recognize if it was running in the Microsoft anti-malware engine's emulator. This meant that malware using the obfuscator would not work if it was run from certain directories. We countered this by changing the current directory, within our engine's virtual environment, to 'C:\Documents and Settings\Administrator\Desktop\'. We made this change only when we recognized a file that appeared to be encrypted with this particular obfuscator. On 7 April 2009 we saw a sample of Win32/InternetAntivirus that contained this anti-emulation code:

```
00419676 push 0 ; lpBuffer
00419678 push 0 ; nBufferLength
0041967A call eax ; GetCurrentDirectoryW
0041967C cmp eax, 0
0041967F jb short cmp_to_48
00419681 jmp short jmp_cmp_to_4
00419683 cmp_to_4:
00419683 cmp eax, 4
00419686 ja short cmp_to_48
00419688 jmp short exit
0041968A jmp_cmp_to_4:
0041968A jmp short cmp_to_4
0041968C cmp_to_48:
0041968C cmp eax, 48
0041968F jb short continue
00419691 jmp short jmp_cmp_to_49
00419693 cmp_to_49:
00419693 cmp eax, 49
00419696 ja short continue
0041969B jmp_exit:
0041969B push 69260152h
0041969E push [ebp+var_10]
004196A0 call sub_4192B1
004196A5 push eax
004196A6 call eax ; GetTickCount
004196A9 call eax ; GetTickCount
004196AC add esp, 40h
004196AE retn
004196AF jmp_cmp_to_49:
004196AF jmp short jmp_cmp_to_49
004196B1 jmp_exit:
004196B1 add esp, 4
```

(Sample SHA1: 0x50E8E5A65023C95C563D957C2ED6465F06 A6792)

Essentially, if the length of the buffer required to hold the current directory string was 48, 49, or <= 4 bytes, the program would exit without decrypting. Our current directory string was 49 bytes long (including the null-terminating character). Notably, this not only stopped our engine from decrypting the file, but it also prohibited malware that used the obfuscator from running in many real situations. Our next move was to change our current directory string to 'C:\Program Files\Internet Antivirus Pro\', which is 41 bytes long including the null-terminating character. This is the directory that the Win32/InternetAntivirus fake scanner ran from. Not long after that change we received a sample with this code:

```
0047D676 push 0 ; lpBuffer
0047D678 push 0 ; nBufferLength
0047D67A call eax ; GetCurrentDirectoryW
0047D67C cmp eax, 5
0047D67F jb short jmp_cmp_to_40
0047D681 jmp short jmp_cmp_to_5
0047D683 cmp_to_5:
0047D683 cmp eax, 5
0047D686 ja short jmp_cmp_to_40
0047D688 jmp short exit
0047D68A jmp_cmp_to_5:
0047D68A jmp short jmp_cmp_to_5
0047D68C cmp_to_40:
0047D68C cmp eax, 40
0047D68F jb short continue
0047D691 jmp short jmp_cmp_to_41
0047D693 cmp_to_41:
0047D693 cmp eax, 41
0047D696 ja short continue
0047D69B exit:
0047D69B push 69260152h
0047D69E push [ebp-var_10]
0047D6A0 call sub_47D2B1
0047D6A5 push eax
0047D6A6 call eax
0047D6A8 pop eax
0047D6A9 call eax
0047D6AA add esp, 40h
0047D6AE retn
0047D6AF jmp_cmp_to_41:
0047D6AF jmp short jmp_cmp_to_41
0047D6B1 jmp_exit:
0047D6B1 push [esp+58h+var_58], 697A6AFEh
... (Sample SHA1: 0x6A9684AF4E9A495A8F78537DBB380332EC 25659)

This was a surprising result. As the decryption/unpacking layer would exit if the buffer containing the current directory was 41 bytes long, such as 'C:\Program Files\Internet Antivirus Pro\', the rogue was essentially crippled. When it ran after installation, or when the system was rebooted, or even when launched through the start menu, it would simply exit. We left our current directory as it was and employed other ways to counter the anti-emulation technique. Again, the next variant of the malware moved on to another technique.

Win32/FakeXPA has been similarly quick in reacting to generic detection. In January 2009 there were 12 changes made to Win32/FakeXPA in reaction to changes we made to our signatures. We were able to track each new variant by monitoring their download locations. Often these locations changed daily, but their redirect web pages were always updated to point to the new domains. The following are a few of the domains used to host Win32/FakeXPA around that time:
\textit{THE MODERN ROGUE – MALWARE WITH A FACE}  \textsc{O’DEA}

- anti-virus-secure-scanner.com
- live-antivirus-pc-scan.com
- premium-anti-virus-scan.com
- bestantimalwarescanner.com
- best-antimalware-scanner.com
- internetupdateserver.com
- securedupdatedownload.com

By monitoring each new variant, we could see that they were changing in reaction to our own signature changes. Win32/FakeXPA’s anti-emulation code was only changed after our own signatures were modified. It is worth noting that Win32/FakeXPA has often used multiple anti-emulation methods at a time. So while at Microsoft we only saw changes in reaction to our own signature updates, it is possible that Win32/FakeXPA was also being updated to evade other anti-virus tools. If these changes did not affect our own detection, we would be unlikely to notice them.

Introducing new anti-emulation techniques in code obfuscation wrappers isn’t the only way rogues have tried to evade detection. Win32/FakeSecSen actually removed obfuscation in an attempt to avoid recognition. The rogue was initially encrypted with ASProtect, but then changed to using no obfuscator or packer at all. This had the effect of making the files appear less suspicious to our engine, to the extent that we no longer detected them generically. Once we adjusted our signatures for this change, Win32/FakeSecSen began a series of changes to their code and functionality to try to evade generic detection. These generally involved splitting strings within the rogue’s binary into multiple parts and then joining them together with sprintf() statements. This not only divided the strings but also changed the code that referenced those strings; enough of these changes would eventually produce a file that was not generically detected.

Changes at this level of the code would be relatively time consuming for the malware creator, requiring a lot of trial and error. Eventually Win32/FakeSecSen’s creator gave up. The increasing amount of work required to escape detection was undoubtedly part of the reason, but there was something else.

### THE MSRT EFFECT

In November 2008, Win32/FakeSecSen was added to the list of families detected and cleaned by MSRT. MSRT was released on 11 November. At the time, updated variants of the rogue were being released, on average, every two days. By 24 November 2008, MSRT had removed the rogue from more than one million distinct computers. On 24 November we saw the last new variant of Win32/FakeSecSen.

Win32/FakeXPA was added to MSRT’s family list in December 2008. MSRT removed it from approximately 400,000 machines in the first week. While significantly smaller than the number of computers cleaned of Win32/FakeSecSen, this was still a relatively large number for a new family.

The kind of changes that Win32/FakeXPA makes in order to evade detection are more sophisticated and yet would require less time and effort to accomplish than those of Win32/FakeSecSen. By introducing new anti-emulation checks into its de-obfuscation code rather than changing the functional code underneath, Win32/FakeXPA has proven much more adaptable. Win32/FakeXPA’s creators could have a list of anti-emulation techniques targeted at any given malware scanner; once the scanner handles one, they could simply move on to the next one. If a scanner begins to detect their de-obfuscation code, adapting it would be more difficult, but this code can be made extremely polymorphic. Without access to the polymorphic engine that generates this code, detection for it tends to be hit and miss.

By contrast, Win32/FakeSecSen’s creators modified the structure – or even the behaviour – of the program itself. Especially when a scanner’s detection is based on behaviour observed by emulating the sample, this could be a lengthy process of trial and error.

Win32/FakeSecSen didn’t disappear completely when development on it ceased, but its prevalence dropped dramatically. Win32/FakeXPA, which has been updated continuously, remains common.

![Figure 11: Removals of Win32/FakeSecSen and Win32/FakeXPA by MSRT.](image)

### CHANGING BRANDS

So far, all of the survival techniques we have examined could be applied to any kind of malware. Indeed, many of them are used in obfuscators that have applied to many different types of malware. Rogues also face survival challenges that don’t apply to other kinds of malware, however.

A brand is something most companies cultivate and protect, so that potential customers might recognize it, but malware creators don’t want their software to be recognized for what it is. Searching the web for ‘Antivirus 2009’, for example, returns a lot of results that clearly label it as unwanted, with information on how to remove it. ‘Antivirus 2009’, by itself, is clearly not a good brand.

One way that rogues have tried to counter this is by constantly changing their brands, sometimes even using multiple brands at once. Win32/FakeSecSen used 12 different names, each with a different user interface appearance or...
‘skin’, most of which were being updated and maintained concurrently.

The code and functionality of the rogue was virtually identical from one variant – or ‘brand’ – to the next. The file names that were used when the rogue was installed varied and the interface elements (buttons, banners and so on) were stored in resources within the rogue binary. Only the resources needed to be changed to give it a different look and feel; Win32/FakeSecSen was effectively ‘skinnable’. This led to many security vendors giving different variants of the rogue completely different names, such as ‘Antivirus2008’ and ‘SpywarePreventer’. Any generic detection for the rogue, however, was unlikely to differentiate one variant from another. These multiple brands also caused confusion between completely different rogues, such as Win32/FakeSecSen (calling itself ‘Windows Antivirus 2008’, ‘Vista Antivirus 2008’ and similar names) and Win32/FakeXPA (calling itself ‘XP Antivirus 2008’).

Win32/FakeXPA was a very different beast from the start. Rather than maintaining multiple brands for the same product simultaneously, its creators would introduce a new variant from time to time, quickly ‘retiring’ the old brand. There are two distinct rogues that Microsoft calls Win32/FakeXPA (as they have shared some common code). The one we will examine is the most actively updated and has extensively used code obfuscation and anti-emulation techniques. Its core functionality has changed very little over time. It is distributed as a small installer, which displays a message box offering a link to its terms and conditions and waits for the user to click ‘continue’ or ‘next’ before downloading and installing the phony scanner.

The current version of Win32/FakeXPA, as of June 2009, calls itself ‘Personal Antivirus’ rather than ‘XP Antivirus’ and boasts a much more sophisticated look, but operates in exactly the same way.

In between the early ‘XP Antivirus’ and the latest ‘Personal Antivirus’ versions, the rogue has been named ‘Antivirus 2009’, ‘Antivirus 360’ and, for a short time, ‘Total Security’.

THE ROGUE DIFFERENCE

A rogue’s brand is one of the elements that separates rogues...
from most other types of malware. A rogue needs a brand or a name associated with it because, to make money, people must willingly pay for it.

**DIRECT PAYMENT**

Regardless of the actions or behaviour of the malware involved, the rogue creator gets their revenue directly from the end-user; that is, the person interacting with the malware. This clearly has advantages over other malware money-making schemes, such as spam, where more steps are involved. It is even more direct than a password stealer simply grabbing a credit card number. The user whose credit card is charged for a rogue is not going to report it stolen if they authorized the transaction. This is also a drawback, of course.

Getting the rogue installed isn’t enough. People must be guided to the payment page and convinced to go through with the transaction.

Some payment pages include social engineering of their own, offering bundle deals and lifetime subscriptions for low prices, but this is just the final stage of a process that commonly starts before any malware is even installed.

**END-TO-END SOCIAL ENGINEERING**

Most of the techniques rogues use to install themselves rely heavily on user interaction, such as redirecting the user’s browser to a fake online scanner before even offering a malware binary to download. One of the reasons they go to these lengths is that the malware itself is part of a larger social engineering attack. This attack begins before the rogue is installed, maybe with a pop-up ‘security warning’ from your browser, before an online scanner is loaded.

The process continues with the online scan which tells you that you need to install security software. The security software then tells you that there is malware on your system and that you must register the software to remove it. As we’ve seen, some rogues go further, with components that imitate elements of the operating system and browser to reinforce your need for the rogue and, in particular, how to go about paying for it. But even for the most basic rogue, the social engineering attack doesn’t finish at this point. For stage three (profit), the rogue needs you to enter your credit card number on an order form. Only then has the malware served its purpose.

In many cases, even the order form is not quite the end. You will most likely receive working registration details which will ‘activate’ the rogue, at which point it will turn off its fake warnings and tell you that it has cleaned the system. This is not just to make you believe you got something for your money, but also to deter you from installing real security software that might recognize other malware that was installed with the rogue and might be still active.

**SOFTWARE AS MARKETING**

Many years ago, virus payloads typically had obvious effects. Often they would destroy data or display messages or graphics that would immediately alert the user to the fact that something was wrong. As profit became the near universal motivator for writing malware, payloads became invisible. Anything that notified the user that something was wrong with their system might lead to the malware being found and removed.

In a way rogues, as malware payloads, hark back to the payloads of old; they want you to know that they are running. But where the messages that the old viruses displayed could be likened to graffiti, the messages displayed by a rogue are advertisements. In fact, the entire program is
advertisement, usually with no other significant functionality at all. Its purpose is simply to convince people to pay money for it. Not surprisingly, rogue creators have preferred advertising tactics that are generally considered questionable at best – lying, exaggerating and promoting fear.

Some of the falsehoods perpetrated by rogue marketing are taken from real and legitimate messaging, but twisted to be applied in the wrong context. A basic example is in the icons that rogue applications use. Win32/FakeSecSen often used an icon identical to that of the Windows Security Center, for example. The general look and feel of most rogues is similar to many legitimate security products; in some cases it is a direct copy.

These tactics start at the rogue web pages. Win32/FakeSpyPro, for example, uses a web page with a similar shield logo and colour scheme to Microsoft’s own security portal.

![Figure 20: Win32/FakeSpyPro web page.](image)

![Figure 21: Genuine Microsoft Malware Protection Center web page.](image)

The testimonials on Win32/FakeSpyPro’s website are not original either. Several are taken from http://www.norton-antivirus-online.com/norton-antivirus-testimonials.html (with the product name changed, of course).

Other rogues take a more original approach but tend more towards exaggeration and fear-mongering. Figure 24 shows an excerpt from the website for a rogue called Win32/FakePlus.

![Figure 22: Testimonials from Win32/FakeSpyPro web page.](image)

![Figure 23: Testimonials from http://www.norton-antivirus-online.com/norton-antivirus-testimonials.html.](image)

This type of disreputable advertising is comparable to the behaviour that makes the software itself potentially unwanted. In some cases both the claims on the website and the behaviour of the software itself fall into a grey area. The
assertions may be exaggerated but not patently false, and the software limited in functionality but harmless. There are many examples of registry ‘fixers’ that do indeed clean the Windows registry of unnecessary entries; however, the positive effects on the computer’s performance and security are likely to much less than the software’s authors claim. When it comes to most modern rogues, however, the claims are blatant lies and the software itself detrimental to the computer it runs on. Copying the logos, icons and testimonials of legitimate security software is just the beginning. The trend over the last couple of years has been to imitate elements of the typical user experience in order to exploit people’s trust in the things they use every day: their operating system, their browser and even their search engine. Nearly all of the most prevalent rogues in recent times have imitated Windows’ security messaging in one way or another. A common technique is to display an imitation of the Windows Security Center.

These are usually accompanied by pop-up messages from a system tray icon. Where real Windows Security Center messages recommend installing anti-virus software, rogue security centres customarily recommend registering the rogue. Where the real Security Center directs you to a page on the Microsoft website that provides a list of anti-virus products to choose from, a rogue security centre will take you directly to a web page where you can pay for the rogue. This kind of behaviour is clearly malicious. Apart from misrepresenting Microsoft’s recommendations, these usually replace or at least obscure the real Security Center and present an inaccurate picture of the state of the computer’s security. Not only is the displayed state of the machine’s virus protection false, but so are those of the firewall and automatic updates.

Rogues have extended this behaviour to other areas of the operating system, from changing the desktop background to

Figure 24: Excerpt from Win32/FakePlus web page.

Figure 25: Fake Windows Security Center windows displayed by Win32/FakeXPA (top left), Win32/FakeSpyguard (top right), Win32/WinSpywareProtect (bottom left), and Win32/FakeRean (bottom right).
displaying phony ‘blue screen of death’ crash messages. Several take the same technique to the browser, displaying messages that imitate Internet Explorer’s own security messages. Win32/Yektel, which is installed by Win32/FakeXPA, displays drop-down messages and full page warnings within Internet Explorer claiming to be from the browser itself. As with the fake security centre messages, they then direct the user to a payment page to register the rogue. Some variants of Win32/Yektel also add warnings, claiming to be from Google, to any pages from URLs with the word ‘google’ in them.

**SOFTWARE AS EXTORTION**

Most rogues go beyond aggressive marketing to sell software that has limited or no functionality. Rogues not only display warnings to promote a solution to a non-existent problem; in some cases they render the computer difficult to use, displaying constant pop-up windows or restricting normal functionality.

In May 2009, Win32/Winwebsec went as far as blocking execution of most programs altogether. Rather than blocking specific programs, it contained a list of programs that it allowed to run. This would lead to most programs failing, making removal of the rogue difficult; however, the technique was also used to further encourage payment of the rogue. Each time a program was blocked, Win32/Winwebsec would display the message shown in Figure 27, falsely reporting an infection and urging the user to activate (i.e. pay for) the rogue.

![Figure 26: A drop down message displayed in Internet Explorer by Win32/Yektel.](image)

When Win32/Yektel displays a full page warning message it effectively blocks access to the page that the user was trying to visit. This is an example of the second type of tactic that rogues creators use to persuade people to pay them money – extortion.

**Figure 27: A false Win32/Winwebsec file infection report.**

This behaviour is similar to that of so-called ‘ransomware’ such as Win32/SMSer, which effectively locks the machine and demands payment to regain access. Other rogues use similar techniques, albeit in more subtle ways. Frequent, repeated pop-up warnings of security compromise are the most common practice. Win32/WinSpywareProtect, for example, shows alert windows that appear in the centre of the screen, others that ‘slide’ in from the right-hand side of the screen, as well as pop-ups from the system tray. The windows it displays float on top of all other windows and provide no mechanism to be minimized or closed. Any attempt to clear the messages usually leads either to a registration window or a second warning. The sequence of answers required to escape each warning is to answer ‘no’ and then ‘yes’. For example, when the rogue’s scan has finished, it displays the message:

*There are serious threats detected on your computer. Your privacy and personal data may not be safe. Do you want to Clean and Protect your PC?*

To remove this warning one must either click the ‘Yes, remove threats’ button and enter the registration process, or click ‘No, continue unprotected’ and be presented with a second warning message:

*Are you sure? Your PC will not be protected against spyware.*

This time one must click ‘Yes, continue unprotected’ to finally escape the warnings or ‘No, purchase full version’ to start the registration process. A similar process must be followed for all of the rogue’s warning messages, which generally appear at least once a minute. As the messages float on top of other windows, the user has little choice but to interact with them. Eventually they are likely to tire of the messages and register the rogue just to be rid of them.

**Figure 28: Win32/WinSpywareProtect warning dialogs.**

As mentioned earlier, Win32/Yektel regularly displays a fake error message in Internet Explorer instead of the web page that the user tried to load. Win32/InternetAntivirus installs a component called Win32/FakeIA that displays similar errors, but accomplishes its purpose in a different way. Win32/Yektel is installed as a Browser Helper Object (BHO), enabling it to intercept content that is rendered by Internet Explorer. BHOs can be managed through Internet Explorer’s ‘Manage Add-ons’ feature and as such, Win32/Yektel is quite simple to disable if you know what you are looking for. Win32/Yektel gives its BHO the innocuous name ‘Research’ or ‘Helper’ in a small attempt to conceal its true purpose.

Win32/FakeIA does not install a BHO, but instead uses code injection, calling WriteProcessMemory() to write its code into the browser’s process and CreateRemoteThread() to execute that code. It can inject into iexplore.exe (Internet Explorer) and firefox.exe (Firefox) using this method, hooking the APIs send(), recv(), connect() and getaddrinfo(). This makes it harder to recognize the source of the browser’s behaviour and also enables a wide range of functionality. Win32/FakeIA displays two types of error messages; one claims to originate from the browser and the other imitates the Google search malware warning page. But Win32/FakeIA goes a lot further than that. Just like a real Google warning, this page has a link to www.stopbadware.org. The link is to the real stopbadware.org, but if that link is clicked, Win32/FakeIA...
intercepts the request and ‘redirects’ it, sending it to a domain that returns pages that imitate www.stopbadware.org. These imitations invariably report that the website the user tried to visit has been reported as a bad site. They contain a link with the text ‘Click to buy Antivirus’, but this doesn’t point at anything suspicious either; it points to a Google search for the term ‘antivirus’.

Win32/FakeIA interferes with searches for ‘antivirus’ as well, but it doesn’t do anything as obvious as poisoning the results to point to a page belonging to the rogue. Instead it allows the normal search results to be returned and redirects any links that the user follows from those results. This elaborate process is just another way to accomplish the two goals of any rogue creator: convince the user they need protection and direct them to pay the rogue maker for that protection. People who aren’t convinced by a recommendation from an unknown source may be more likely to be swayed by seemingly genuine messages from a source they trust. Even those that are wary of recommendations that seem to come from their browser or search engine are likely to be less suspicious of sites they arrive at through actively searching for a term like ‘antivirus’.

Win32/FakeIA also prevents the browser from retrieving pages from many domains that offer security products and advice. This is a basic malware technique for self-preservation, but Win32/FakeIA does this too with more subtlety than most. It permits the host name lookup and even the connection to the remote server, but it doesn’t pass through the data that the browser sends. So the browser waits for a response to its request but gets no answer, as the request never actually gets sent. Eventually the browser times out, as it would if the server was unresponsive. By filtering the requests from its send() hook, Win32/FakeIA is also able to block a particular path such as ‘google.com/webmasters’, rather than the entire google.com domain. It also allows the technique to work if the affected browser uses a proxy. As some of the substrings that will cause Win32/FakeIA to block a request are quite general, such as ‘spyware’ and ‘viruses’, it also contains a ‘whitelist’ of domains that are always allowed. This includes the rogue’s own payment sites and most search engines. Of course, all of the fake security warnings that rogues display are in fact a sign that the computer does have a security problem. Like the mob boss who demands payment for ‘protection’, rogues supply not just the offer of security, but the threat itself. The standard malware techniques rogues use to conceal themselves and avoid removal are likely to reinforce the idea that there is something wrong with the computer. Many rogues are either installed with other malware, or install other malware themselves. Win32/InternetAntivirus installs a password stealing trojan called Win32/Chadem, and Win32/Winwebsec sometimes downloads a worm called Win32/Slammer, for example. Rogues have often been installed by malware such as Win32/Cbeplay, which at the same time installs a spam bot like Rustock or Cutwail. Some proportion of users affected by this other malware is likely to notice something wrong and look for a solution. If a rogue was installed at the same time, its warnings of malware and security compromise are likely to be quite convincing.

CONCLUSION
The most widespread rogues are not merely overzealous security programs, but malware that is constructed and distributed in a fashion similar to backdoors, password stealers and spam bots. Displaying false alerts of system compromise and eliciting payment for protection is the direct payload of many of the most common malware families today. Not only are rogues themselves common, but they are also the indirect payload for many other malware families, including sophisticated examples such as Win32/Waledac and Win32/Koobface. The prevalence of the rogues, combined with the amount of work that goes into developing and updating them, demonstrates that they are a successful and popular money-making tool for malware authors. Rogues face unique challenges in terms of distribution, survival and exploiting their targets, but they have already proven resilient and are likely to remain a significant part of the malware landscape for some time to come.

FURTHER READING
For more information on the malware discussed in this paper, please see the Microsoft Malware Protection Center: http://www.microsoft.com/security/portal/.

Descriptions of the rogues mentioned throughout the paper can be found at these locations:

- Win32/FakeSecSen

- Win32/FakeXPA

- Win32/Winwebsec

- Win32/InternetAntivirus

- Win32/FakeSpyguard

- Win32/WinSpywareProtect

- Win32/FakeSpyPro
• Win32/FakeRean

• Win32/FakePlus

• Win32/Yektel

• Win32/FakeIA

• Win32/FakeRemoc

• Win32/Winfixer

• Win32/Antivirusxp

REFERENCES