Fake Antivirus: Journey from Trojan to a Persistent Threat

By Jagadeesh Chandraiah, Researcher, SophosLabs UK
Contents

Abstract .................................................. 2
1. Introduction ........................................... 2
2. FakeAV Trends ......................................... 3
3. Infection Vectors ....................................... 4
   3.1 Black Hat SEO ...................................... 4
   3.2 Malvertising ......................................... 5
   3.3 Spam Campaigns ..................................... 5
   3.4 Exploit Kit ........................................... 6
4. Packer Evolution ....................................... 8
   4.1 Anti-Emulation API ................................. 8
   4.2 Process Environment Block (PEB) and Thread Information Block (TIB) Access 9
   4.3 Kuser Shared Data (KSD) .......................... 9
   4.4 How is this done? ................................. 10
5. What drives FakeAV? .................................. 11
6. Related Work .......................................... 11
7. Conclusion ............................................ 11
References ............................................... 12
Abstract
Fake antivirus (FakeAV) is one of the largest families of malware that we have seen in recent times. FakeAV has grown over the years to be a persistent and prevalent threat. In this paper, we study the evolution of FakeAV over the last three-and-a-half years. We analyse the major FakeAV events, infection vectors and some important anti-emulation/anti-reverse engineering (RE) tricks used by FakeAV packers. We also analyze how exploit kits are used to infect users with FakeAV and study how a polymorphic packer found in underground internet forums is used to encrypt and compress the malware binary.

1. Introduction
FakeAV (a type of scareware) pretends to scan a computer for viruses and presents the user with false malware alerts in order to swindle them to clean nonexistent threats. The number of FakeAV variants has increased considerably over the last three-and-a-half years as a result of its success in infecting so many users and because of the profits it earns for the author [1]. A typical FakeAV infection warning is as shown in Figure 1.

If we study the FakeAV infrastructure, the software itself and its method of infection, we can see that this threat is not the work of script kiddies. It is an organized work of professional cyber criminals. Given its large volume of samples and the technicality involved in the malware, a greater study of important components of FakeAV will better equip us to confront future threats of this scale and to improve our protection mechanisms accordingly.
2. FakeAV Trends
FakeAV remains an active threat. Every day many people find their computers infected with this malware [21]. We started to see a large rise in FakeAV infections in 2009 [18] where it was used in some of that year’s largest malware attacks. For example, in one instance The New York Times website was used to distribute their FakeAV through the website’s advertising network [11]. Some government embassy sites were infected as a result [22].

During that year we saw FakeAV being spammed out in great numbers [19], and social networking websites Facebook and Twitter were used to spread FakeAV [23]. By using such popular internet services, the FakeAV perpetrators were able to achieve high success in distribution rates and to earn large profits as a result [1]. It was these huge profits that drove them to target other operating systems and eventually led to FakeAV for Mac [24].

In August 2011, there was a sharp decline in some of the FakeAV reports due to the action of law enforcement agencies. This made the processing of the victims’ credit cards more difficult [26]. Though there was a brief decline in FakeAV infections, in recent months we have seen a resurgence of both older variants and new strains. The graph shown in Figure 2 shows that we are still seeing thousands of samples of FakeAV every month.

Figure 2. Sophos top five FakeAV detections.
3. Infection Vectors
In this section we look at the various infection vectors used by FakeAV to reach end users. FakeAV has evolved considerably and employs many different techniques to infect users. Some of the most common methods used include social engineering, spam campaigns and advertising networks [1]. We briefly discuss each of these methods and analyze how they work.

3.1 Black Hat SEO
Black Hat search engine optimization (SEO) is a technique used to trick search engines into displaying malicious URLs in search results. The malicious webpages are filled with popular keywords in order to achieve a higher ranking in the search results. When the end user searches the web, one of these infected webpages is returned. If selected, there may be multiple redirections, eventually leading to a FakeAV payload [5].

In most cases, SEO attacks are hosted on compromised websites to benefit from their reputation and make them hard to blacklist [5]. Usually the most popular keywords from services such as Google Trends are used to generate webpages via PHP scripts placed on the compromised website. These PHP scripts will then monitor for search engine crawlers and feed them with especially crafted webpages that are then listed in the search results. Then, when the user searches for their keyword or images and clicks on the malicious link, they will be redirected to the FakeAV payload [6].
3.2 Malvertising
Advertising networks are commonly used to distribute FakeAV [4, 7]. Website owners who wish to display adverts on their webpages usually employ third-party services to do so. This is often achieved by embedding trusted JavaScript or Iframes into their webpages. If one of these trusted advertising vendors is compromised, they may end up inadvertently infecting all of the websites using their service [8, 9].

In September 2009 the New York Times website was serving FakeAV as a result of one of their advertising services becoming compromised [11]. Figure 3.2 is an example of how malicious JavaScript can be injected alongside legitimate advertising script in order to redirect users to a Chinese domain. In practice, this leads to further redirects eventually leading to a rogue website ‘online-antivir-scan09<dot>com’.

3.3 Spam Campaigns
Spam messages that include malicious attachments, links to binaries and drive-by download sites are another common mechanism for distributing FakeAV. Spam emails are often sent with content associated with typical day-to-day activities such as parcel deliveries, or taxation documents, designed to entice users to click on links or run attachments. When users succumb to these kinds of social engineering tricks they are quickly infected either directly via the attachment, or indirectly via a malicious website. This is known as a drive-by download. Usually in drive-by download attacks the malware is installed on the victim’s machine without any interaction or awareness and occurs simply by visiting the website [9].
3.4 Exploit Kit

Exploit kits are an important component of the FakeAV distribution chain. They are often used to infect users as part of the drive-by download attacks mentioned above. Here we will briefly describe how a common exploit kit called Blackhole can be used to distribute FakeAV (and other malware) to end users. There are several exploit kits that are used in this way, of which Blackhole is one of the most popular and sophisticated. The Blackhole exploit kit is protected with a commercial PHP packer in an attempt to prevent analysis. It exploits many of the vulnerabilities commonly targeted by malware, including those for Java and PDF.

As shown in Figure 3.4, when the victim navigates to an infected webpage the exploit kit will gain information about the user’s computer (e.g., operating system, browser type Etc) and will attempt to exploit various appropriate vulnerabilities. If exploitation is successful, the victim will be infected with FakeAV or potentially any other malicious payload.

Criminal gangs using exploit kits are sophisticated in employing custom/commercial packers to try and avoid antivirus detection and analysis. Figure 3.5 is a snippet of obfuscated JavaScript code from a Blackhole exploit script.

Figure 3.6 and Figure 3.7 show the extracted parts of the de-obfuscated JavaScript which includes code to check PDF, Java and Flash versions, as well as dynamic iframe creation code.
The Blackhole exploit kit has different configuration options in its user interface. Figure 3.8 and 3.9 are screenshots showing the number of infections by country and the IP blacklisting feature of Blackhole, respectively.

Figure 3.8: Blackhole Exploit kit showing infections by country and exploits [15].

Figure 3.9: IP Blacklisting used by Blackhole [15].
4. Packer Evolution
FakeAV authors have graduated over the years from not using any packers to using some of the most complex polymorphic packers that we have seen. Figure 4.1 is a code extract from a FakeAV variant from 2008 that does not have a packing layer. Note that without a packing layer we can clearly see some of the typical strings used in early FakeAV variants.

Over the years, different variations of FakeAV have used several layers of packing in order to try and evade antivirus detection. We observe that during the evolution of FakeAV the majority of changes have occurred in the anti-emulation and anti-reverse engineering (RE) tricks. In this section, we will look at some of these tricks used by FakeAV packers over the last eighteen months.

4.1 Anti-Emulation API
An emulator is the name given to software used to replicate the behaviour of another system. X86 emulators are used to mimic the behaviour of the x86 family of processors. Emulators are often used by antivirus engines to simulate the behaviour of a program during execution. Malware authors are aware of this, and will insert specific anti-emulation instructions into their code in order to try and break emulation by antivirus software. A good example of this is shown in Figure 4.2.

As you can see, the return value of the XRegThunkEntry API is expected to be six. If any other value is returned, the code will jump to an infinite loop of meaningless instructions. Emulators usually support the most common APIs, but when it executes this odd API (XRegThunkEntry) it is unable to return the expected value, and thus will cease to emulate the sample further. Unless the emulator is extended to support XRegThunkEntry, emulation will cease at this point.

If we follow the code in Figure 4.2 further we see that WSAddressToStringW is also used as an anti-emulation API. In this case, even if the emulator supported this API and returned the proper value, the malware author expects it to set a specific return value to GetLastError. Any other value leads to the infinite loop.
4.2 Process Environment Block (PEB) and Thread Information Block (TIB) Access
The PEB is an area in memory which contains information about each running process. The PEB and TIB are often used by FakeAV and other malware families for anti-emulation/anti-debugging tricks. Figure 4.3 is an example of the PEB being used by the Mal/FakeAV-LS variant. In the sample code, EAX gets added to 0x30 in order to evade any detections by antivirus software that look for the direct access to the PEB. After fetching the value of the PEB, it queries the 0x1F8 location which points to the ActivationContextData in the PEB structure. It then compares the value stored at that location to decide its further flow.

The TIB is a data structure in the Windows operating system which holds information about the running thread. There is a TIB for every thread running in the system. The values in this structure can be used by the malware to detect emulation/debugging of the sample. In Figure 4.4 the malware is accessing the TIB using FS:18 and checking the value at 0x10. This points to FiberData in the TIB structure to decide the next instruction to execute.

4.3 Kuser Shared Data (KSD)
Kuser Shared Data is a shared area in memory which contains a vital data structure for Windows. KSD is mapped into all the processes in memory usually at 0x7FFE0000 and direct access to it is limited [14]. If KSD is not populated with appropriate values, the malware authors use that to detect the presence of an emulator.

As investigated by Mathur et al [13], the FakeAV packer code is also known to access 0x7FFE0304 (SystemCallReturn). It uses the value C3 at this location as a return from obfuscated calls. The value at 0x7ffe0030 (NtSystemRoot) can then be used as a decryption key [14].

Figure 4.3: Process Environment Block Access.

```assembly
mov eax, 0dh
add eax, 16h
add eax, 0dh
mov eax, fs:[eax] ; FS:[30] PEB
mov [ebp+var_10], 0FFFFFFF56h
mov [ebp+var_C], eax
mov [ebp+var_8], 0FFFFF31h
mov eax, [ebp+var_C]
mov ecx, [eax+1F8h] ; PEB + 1F8 (ActivationContextData)
mov [ebp+var_10], ecx
cmp [ebp+var_10], 0 ; test value
```

Figure 4.4: Thread Information Block Access.

```assembly
mov edx, 18h
xor ecx, 0B42AA20h
mov eax, fs:[edx] ; Access TIB
ror edx, 06h
mov edx, eax
mov esi, [edx+10h] ; TIB +10 (FiberData)
shr esi, 06h
add esi, 0FFFFFFFh
test esi, esi ; Check Value
```

Figure 4.5: KUSER_SHARED_DATA Access.

```assembly
test dword ptr ds:7FFE0004h, 0FFFFFFFh
jz short junk_locn
```
4.4 How is this done?
Packers are used as compression and/or encryption layers to wrap malware binaries and attach the decompressing/decryption code to the original file. FakeAV authors extensively use custom packers or packers advertised on underground forums [20]. Figure 4.6 shows an example of a list of different packers displayed on such forums [20].

For the purpose of this discussion, we verify an old polymorphic packer called Crum Polymorphic Packer, which can be used to pack PE files. As we see in Figure 4.7, the packer supports anti-dumping, changing file icons, variable key length and variable anti-emulation features.

We used Crum to pack notepad.exe in order to observe the changes made by the packer. Figure 4.8 and Figure 4.9 show the PE file structure before and after packing, respectively. After the packing, two additional sections with packed data and minimal stub are added to the file.
The new sections have the packed data with the entry point in the last section. As this is a polymorphic packer, the code in the entry point changes every time the file is packed in order to evade any basic signature-based detection. When we tested the anti-emulation features of the packer, junk loops and MMX [27] instructions were added to trick the emulator, as shown in Figure 4.10.

FakeAV and many other major malware families have started to use anti-emulation/anti-RE tricks increasingly to evade emulator-based PE malware detection. It’s very important for security vendors to continually improve their emulation technology and stay on top of these threats.

5. What drives FakeAV?
Many of the criminals behind FakeAV operate in a network of affiliates called Partnerka [26]. These affiliates are formed by thousands of advertisers backed by numerous gangs. Partnerka use techniques like Black Hat SEO, social networks, spam and other malware families in order to drive more traffic to their affiliate sites. These affiliates get paid for every successful install (also known as a pay-per-install scheme) and based on FakeAV-generated revenue. Stone-Gross et al. [1] studied three such underground FakeAV business networks and revealed that they had netted more than $130 million dollars between them.

6. Related Work
FakeAV has long been one of the most interesting malware families on the antivirus radar. Different stages of the FakeAV lifecycle have been analyzed and revealed. In a similar study, Trend Micro [12] analyzed the different infection vectors and malware behaviour. It confirms the infection mechanisms described in this case study, but it doesn’t include the analysis of packers themselves. Also the various anti-emulation and anti-RE tricks described by Mathur et al. [13] reassert some of the tricks presented in this paper.

7. Conclusion
In this study we have analyzed the infection vectors, tricks used in packer layers and the evolution of FakeAV over the last three-and-a-half years. From this study, we have a better understanding of FakeAV operations and are better prepared to defend against similarly scaled threats in the future. We have analyzed different components of the FakeAV threat as a single entity and understood its strategy and techniques well. We conclude that FakeAV is one of the major threats of recent times and is still actively infecting users. Future work could involve monitoring different operations and components of this threat and use that to build systems that can better protect users from FakeAV infection.
References


Fake Antivirus: Journey from Trojan to a Persistent Threat


20. www.damagelab.org


