NOVETTA

ZoxPNG Preliminary Analysis

Overview

ZoxPNG is a very simple RAT that uses the PNG image file format as the carrier for data going to and from the C2 server. ZoxPNG supports 13 commands natively. In addition, ZoxPNG has the ability to load and execute arbitrary code from the C2 server providing an almost unlimited feature set. For instance, ZoxPNG provides no functionality for key logging, screen grabbing or file execution. If an attacker required such functionality, the attacker could construct a simple shellcode binary which the ZoxPNG binary could execute thereby expanding the feature set of the Trojan.

ZoxPNG does not contain any configuration information. The attacker using ZoxPNG must specify the C2 address as a command line argument.

Functions

ZoxPNG uses a notion of function registration to assign command handlers to specific, sequential IDs. The order in which the handlers are registered dictates the ID of the command. The ID values start at 0x80061001. The following ID to function mappings have been observed:

ID	Function Description
0x80061001	Initiate a remote shell
0x80061002	Interact with the remote shell (send command, read response)
0x80061003	Download a file from the C2 to the victim's machine
0x80061004	Upload a file to the C2 from the victim's machine
0x80061005	Obtain information about the attached drives
0x80061006	Create a directory
0x80061007	Find/List files
0x80061008	Delete a file
0x80061009	Move/Rename a file
0x8006100A	List all activate processes
0x8006100B	Kill a process (by PID)
0x8006100C	Sleep
0x8006100D	Add a new handler function
0x8006100E	Shutdowns ZoxPNG

Communication

Communication between the C2 and the ZoxPNG binary occurs through the HTTP request for a PNG file. For requests to the C2, the Trojan issues a GET request to a specific URL whereas requests that contain information from the Trojan for the C2 occur as POST requests with an attached PNG file. The format for the URL is as follows:

http://{C2 Address}/imgres?q=A380&hl=en-US&sa=X&biw=1440&bih=809&tbm=isus&tbnid=aLW4-J8Q1lmYBM:&imgrefurl=http://{C2Address}&docid=1bi0Ti1ZVr4bEM&imgurl=http://{C2 Address}/{4 digit year}-{2 digit month}/{4 digit year}{2 digit month}{2 digit day}{2 digit hour}{2 digit minute}{2 digit second}.png&w=800&h=600&ei=CnJcUcSBL4rFkQX444HYCw&zoom=1&ved=1t:3588,r:1,s:0,i:92&iact=rc&dur=368&page=1&tbnh=184&tbnw=259&start=0&ndsp=20&tx=114&ty=58

ZoxPNG encodes additional information within the Cookie: field of the HTTP request. This information consists of:

```
#pragma pack(push, 1)
struct VictimSystemData
{
        char fIs64BitProcess;
        char field_1;          // binary result of an obscure test
        char bOSMajorVersion;
        char bOSMinorVersion;
        int dwActiveCodePage;
        int dwRandomValue;
        int dwMegsOfMemory;
        int dwPID;
        char szComputerName[32];
};
#pragma pack(pop)
```

The Cookie: field contains the VictimSystemData within the SESSIONID= parameter as a Base64 encoded string.

The HTTP request will include a user-agent field. ZoxPNG attempts to call the API function ObtainUserAgentString in order to use the local user-agent string. In the event that the API call is unsuccessful, ZoxPNG will default to the following user-agent string:

Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NETCLR 2.0.50727)

The format of the PNG file that carries data to and from the C2 is relatively straight forward. From data coming from the C2, the PNG file must start with the following bytes: [0x89, 0x50, 0x4E, 0x47, 0x0D, 0x0A, 0x1A, 0x0A]. The DWORD starting at offset 0x21 contains the size of the data within the PNG file while the data begins at offset 0x29. The DWORD at 0x21 is in big-endian format. The data at offset 0x29 is compressed using the deflate (version 1.1.4) system. Novetta was unable to observe a live sample of the PNG file originating from the C2 but it is reasonable to believe that the overall format of the PNG file is the same as the format as the PNG file that the ZoxPNG binary sends to the C2 as the important offsets of 0x21 (33) and 0x29 (41) are identical.

The format of the PNG file originating at the ZoxPNG side of the conversation has a defined format which could be potentially leveraged by IDS. The following table defines the known values of the PNG value (regardless of the data appended):

Offset	Known Values	Notes
0 (8 bytes)	0x89 0x50 0x4E 0x47 0x0D 0x0A 0x1A 0x0A	PNG header
8 (4 bytes)	0x00 0x00 0x00 0x0D	Length of image header chunk
12 (4 bytes)	'IHDR'	Image header tag
16 (13 bytes)	0x00 0x00 0x00 0xC8 0x00 0x00 0x00 0x64 0x08 0x00 0x00 0x00 0x00	Specifies 200x100px 8-bit image
29 (4 bytes)	0xE6 0xED 0x20 0xD7	CRC32 value of IHDR chunk
33 (4 bytes)	variable	Size of IDAT (embedded data) chunk
37 (4 bytes)	'IDAT'	Data header tag
41 (n bytes)	variable	Embedded data of n bytes
41+n (4 bytes)	variable	CRC32 value of IDAT chunk
45+n (4 bytes)	0x00 0x00 0x00 0x00	Length of IEND chunk
49+n (4 bytes)	'IEND'	Image end tag
53+n (4 bytes)	0xAE 0x42 0x60 0x82	CRC32 value of IEND chunk

Note that the embedded data within the IDAT tag may be compressed using the deflate function.

Detection

Detecting ZoxPNG over the network could be possible by looking for the following string which appears to be static among the observed samples:

png&w=800&h=600&ei=CnJcUcSBL4rFkQX444HYCw&zoom=1&ved=1t:3588,r:1,s:0,i:92&iact=rc&dur=368&page=1&tbnh=184&tbnw=259&start=0&ndsp=20&tx=114&ty=58

Detecting ZoxPNG on disk is possible using the same string as indicated in the following YARA signature:

```
rule zox
{
	strings:
		$url =
"png&w=800&h=600&ei=CnJcUcSBL4rFkQX444HYCw&zoom=1&ved=1t:3588,r:1,s:0,i:92&iact=rc&dur=368&page=1&tbnh=184&tbnw=259&start=0&ndsp=20&tx=114&ty=58"
	condition:
		$url
}
```

Known Samples

The following table identifies the known ZoxPNG samples along with key metadata for each.

SHA1	Compile Date	File Size
60415999bc82dc9c8f4425f90e41a98d514f76a2	10 May 2013 at 07:16:54	44,432 bytes
40f9cde4ccd1b1b17a647c6fc72c5c5cd40d2b08	10 May 2013 at 07:16:54	47,200 bytes
7dd556415487cc192b647c9a7fde70896eeee7a2	10 May 2013 at 07:16:54	47,207 bytes

Evolution

Sample SHA1:b51e419bf999332e695501c62c5b4aee5b070219 appears to have a tangential relationship to the ZoxPNG samples listed above. The sample, known as ZoxRPC, has a compile date of 11 July 2008 at 04:28:21, placing it nearly 5 years ahead of the known ZoxPNG samples. Given the large time differential between ZoxRPC and ZoxPNG, making a direct relationship between the two generations is difficult. There are several attributes that would appear to indicate a connection between the two Zox variants:

1. The use of the term “iiscmd” with a relationship to the remote shell functionality
2. The identifiers used for each command roughly align.

ZoxRPC ID	ZoxPNG ID	Function Description
0x80061001	0x80061001	Initiate a remote shell
0x80061005	0x80061002	Interact with the remote shell (send command, read response)
0x80061003	0x80061003	Download a file from the C2 to the victim’s machine
0x80061002	0x80061004	Upload a file to the C2 from the victim’s machine

ZoxRPC uses the MS08-067 vulnerability, specifically portions of code found on this public website: http://www.pudn.com/downloads183/sourcecode/hack/exploit/detail861817.html.

By researching the unique strings related to the iiscmd, iisput, and iisget strings, it appears that the original source code, the code upon which all Zox variants are based, dates back to 2002. As part of the IIS vulnerability disclosure of 2002 for the vulnerability MS02-018, the source code for the proof of concept code contains not only several strings found within the Zox binaries, but several of the functions as well. The source code upon which the Zox family is based is found at http://www.exploit-db.com/download/21371/. Given the several years between the original source code, 2002, and both ZoxPNG (2013) and ZoxRPC (2008), the code upon which Zox is based has mutated and evolved, but there are clearly sections of code that have remained largely unaltered.