Rovnix bootkit framework updated

welivesecurity.com/2012/07/13/rovnix-bootkit-framework-updated/

July 13, 2012

Changes in the threatscape as regards exploitation of 64-bit systems, exemplified by the latest modifications to the Rovnix bootkit.

13 Jul 2012 - 11:05AM

Changes in the threatscape as regards exploitation of 64-bit systems, exemplified by the latest modifications to the Rovnix bootkit.

We have been tracking the activity of the Rovnix bootkit family since April 2011. Rovnix was the first bootkit family to use VBR (Volume Boot Record) infection (NTFS bootstrap code) for loading unsigned kernel-mode drivers on x64 (64 bit) platforms. The reason for exploring further is the desire of the Rovnix developers to bypass antivirus detection. The payload of the first samples in the wild blocked internet connection for Russian users and forced them to send an SMS to a premium number in order to get their connection unblocked (Hasta La Vista, Bootkit: Exploiting the VBR).

These variants with the internet blocking payload stopped using the bootkit component during the summer of 2011. At that time the Rovnix framework was sold to Carberp developers responsible for the botnet Hodprot/Origami (Evolution of Win32Carberp: going deeper). The Carberp developers used droppers incorporating bootkit framework only up to the end of 2011. We don't have information about other sales of the Rovnix bootkit framework. But we only have information relating to a really small percentage of infections with Rovnix based bootkit code. A few days ago we got some interesting samples and quick analysis revealed similar VBR modifications to the Rovnix.B family. After unpacking the dropper we found a typical component for providing the next steps for installation of the Rovnix bootkit's BkSetup.dll module. The compilation timestamp of BkSetup.dll module looks fresh and is dated 24/06/2012: certainly it's possible to fake a timestamp, but up to now developers had not changed the date on BkSetup.dll).

Field Name	Data Value	Description
Machine	014Ch	i386®
Number of Sections	0005h	
Time Date Stamp	4FE761A8h	24/06/2012 18:51:20
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	0102h	
Magic	010Bh	PE32
Linker Version	0009h	9.0

There is also a version information structure to be found in the resource section of BkSetup.dll filled as follows:

```
Child Type:          StringFileInfo
Language/Code Page:  1033/1200
FileDescription:     BkSetup Library
FileVersion:         2, 4, 0, 0
InternalName:        BkSetup
LegalCopyright:      Copyright (C) 2011
OriginalFilename:    BkSetup.dll
ProductName:         BkSetup Library
ProductVersion:      2, 4, 0, 0
```

The file version has been changed to 2.4: previously, Carberp used version 2.1 of BkSetup.dll and stored the information in the debug string sent to the C&C as debugging information.

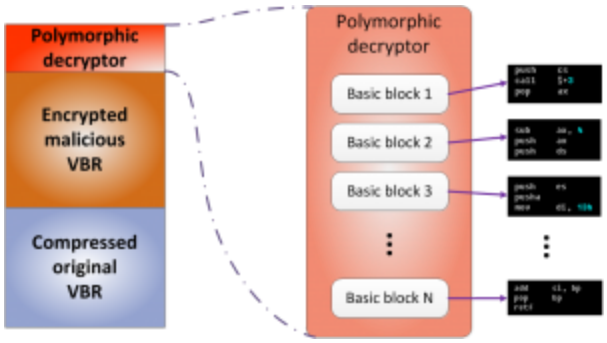
```
BKSETUP_%04x: BK setup dll version 2.1.
BKSETUP_%04x: Attached to a 32-bit process at 0xxx.
BKSETUP_%04x: Detached from a 32-bit process.
BKSETUP: Failed generating program key name.
BKSETUP: Already installed.
BKSETUP: OS not supported.
BKSETUP: Not enough privileges to complete installation.
BKSETUP: No joined payload found.
BKSETUP: Installation failed because of unknown reason.
BKSETUP: Successfully installed.
BKSETUP: Version: 1.0
BKSETUP: Started as win32 process 0xxx.
BKSETUP: Process 0xxx finished with status %u.
BKSETUP: Version: 1.0
BKSETUP: Started as win32 process 0xxx
```

The base functionality of BkSetup.dll is centred on the process of infection and setting up the hidden storage partition. A call graph of the main BkSetup.dll routines looks like this:

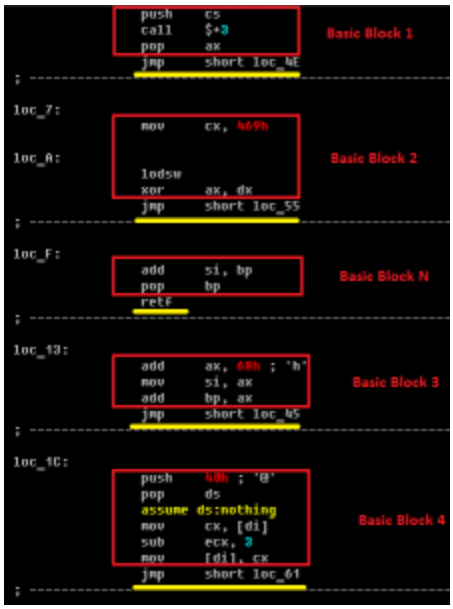
A new sample with a new version of the Rovnix bootkit framework is indirect evidence of renewed sales activity and in the near future we may possible be able to disclose details of the relationship between Rovnix and other malware families. And now we will go deeper into the technical details of the Rovnix.D modification.

Polymorphic bootstrap code

Since Rovnix.B the modified bootstrap code has used polymorphic code in order to bypass static antivirus signature detection. Originally, polymorphic decryption code was detected in Carberp samples incorporating bootkit code. The following figure shows the basic workings of the polymorphic decryption code.

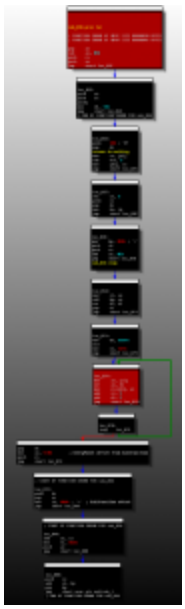


A simple trick of polymorphism based on permutations of the basic code blocks always results in the malware's getting control of the decrypted malicious VBR code. The basic code blocks look like this:



[polymorphic code from Rovnix.B]

The reason for using polymorphic code is to bypass static signature detections by antivirus engines: this code can only be detected generically using emulation. Emulation is a technique closely related to sandboxing where the code is executed in a safe virtual environment in order to analyse it dynamically. The differences between the Rovnix.D and Rovnix.B versions are presented in this flow graph:



[Rovnix.D variant (left) and Rovnix.B variant (right)]

The place where the encrypted malicious VBR is stored has been changed too, and after execution of the polymorphic code, control is transferred to the decrypted malicious VBR code. The different placement of the encrypted malicious VBR looks like this:

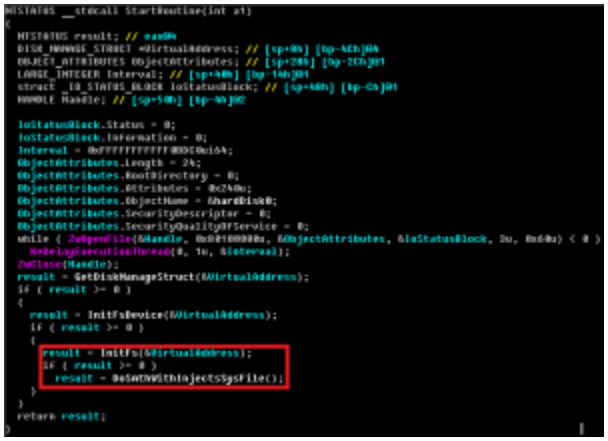
We also found changes to the function for decrypting and reading the malicious unsigned driver from raw sectors in the hard drive. These sectors are not used for general hidden storage, but only as a location for storing the malicious driver, which injects the payload into specified user-mode processes on the infected machine. Differences can be seen in the following figure:

[Rovnix.D variant (left) and Rovnix.B variant (right)]

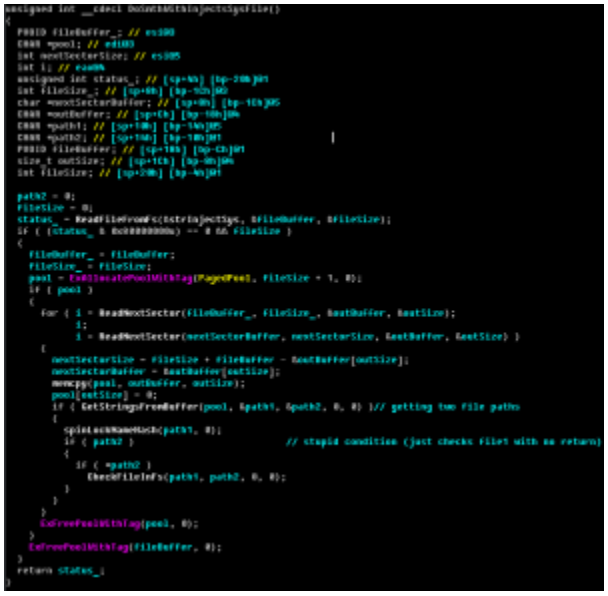
All these changes are directed towards bypassing antivirus detections and do not represent fundamental changes in the general Rovnix bootkit framework structure.

Changes in hidden file storage

The structure of the hidden file system looks similar to the previous Rovnix modification and has already been described in previous blog posts (Rovnix Reloaded: new step of evolution). However insignificant changes were found in the file system initialization code. A strange function call was detected with the ability to read the file INJECTS.SYS from hidden storage.



This curious function extracts one or two paths from the file INJECTS.SYS to files on the standard file system. The function code is presented in the figure below:



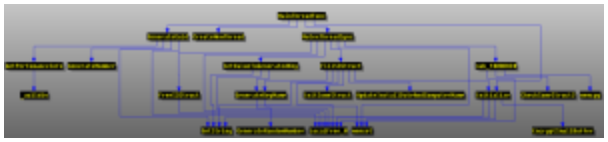
Control flow never gets to execution of this code because the condition always receives NULL and control is never transferred to this code. In my opinion this modification in Rovnix.D is used for tests, and we aren't seeing many detections in the wild. Rovnix.D seems to be a transitional version in preparation for something else, but at this moment we don't have a clear understanding of what that might be.

Payload

The payload module includes functionality for downloading and executing additional modules from the C&C server. This module does not provide hooks and other malicious modifications to system memory, and is not generally detected by most common antivirus engines.

The C&C domain is rtttt-windows.com

The payload module works in multithreading mode and can communicate with the malicious driver. For synchronization reasons the payload generates the mutex: Global<Generated_string>. The call graph for the main thread looks like this:



The payload module can also send an encrypted buffer to the malicious driver to be written to hidden storage and injection into processes. Before this Rovnix has not used any functionality for multiple injects, and provided only one payload module. Rovnix.D can use multiple payloads and can be used to provide a botnet for rent, and at the end of the rental period the payload will be changed.

Conclusion

At this moment changes can be seen in the landscape of complex threats for the x64 platform. The Sirefef (ZeroAccess) family has been migrated to user-mode in its latest modifications (ZeroAccess: code injection chronicles). Olmarik/Olmasco (TDL4 and MaxSS modification) does not account for a large percentage of infections in the wild and has stopped evolving (The Evolution of TDL: Conquering x64). Why are rootkits/bootkits for the 64 bit platform dying? In my opinion the ways in which x64 systems can be infected are severely limited, and the search for something new requires ample time and considerable experience on the part of the developer. Most bootkit infections have used MBR-modification, but this method is pretty old and by this time most common antivirus engines provide checks for a modified MBR. The Rovnix family used other ways to infect with modification of the VBR, but a constant stream of new modifications necessitates the provision of a great deal of debugging information to the C&C. The complexity of development and debugging on multiple platforms is one reason for the high price of the Rovnix bootkit framework. For example the fully-featured builder costs $60.000 including basic support for half a year. This price is only for the bootkit package and excludes the cost of exploits for escalating privilege in order to get access allowing modifications deep into the system.

In future, complex stealth technologies will mostly be used in targeted attacks, because the cost of buying and using them is not commensurate with the anticipated profit for typical cybercriminals. We now have less than ten families of x64 bootkits and their activity in the wild is also decreasing.

SHA1 hashes for the Rovnix.D droppers mentioned are:

- C1C0CC056D31222D3735E6801ACBA763AC024C5B
- 764B4F0202097F2B41A2821D30A7424490BF3A42

Special thanks to my colleagues Pawel Smierciak and Maxim Grigoryev.

Aleksandr Matrosov, Security Intelligence Team Lead

13 Jul 2012 - 11:05AM

Sign up to receive an email update whenever a new article is published in our Ukraine Crisis – Digital Security Resource Center

Newsletter

Discussion