EvilBunny: Malware Instrumented By Lua

December 16, 2014

Cyphort Labs has come across a sophisticated malware sample, aiming to trick sandboxes and showing rather uncommon tricks to evade detection. The malware is dubbed 'EvilBunny' and is designed to be an execution platform for Lua scripts injected by the attacker.

EvilBunny is a technically fascinating piece of malware, among a set of targeted samples seen in the wild around 2011. The name EvilBunny is derived from debug information embedded in the malware's dropper. The malware itself is written in C++, multi-threaded, aims to detect installed anti-virus- and firewall solutions and accepts a vast number of different control commands. Furthermore, the specified piece incorporates a Lua 5.1 interpreter, which allows the malware to execute Lua scripts and change its behavior at runtime.

BINARY DETAILS

Malware Dropper

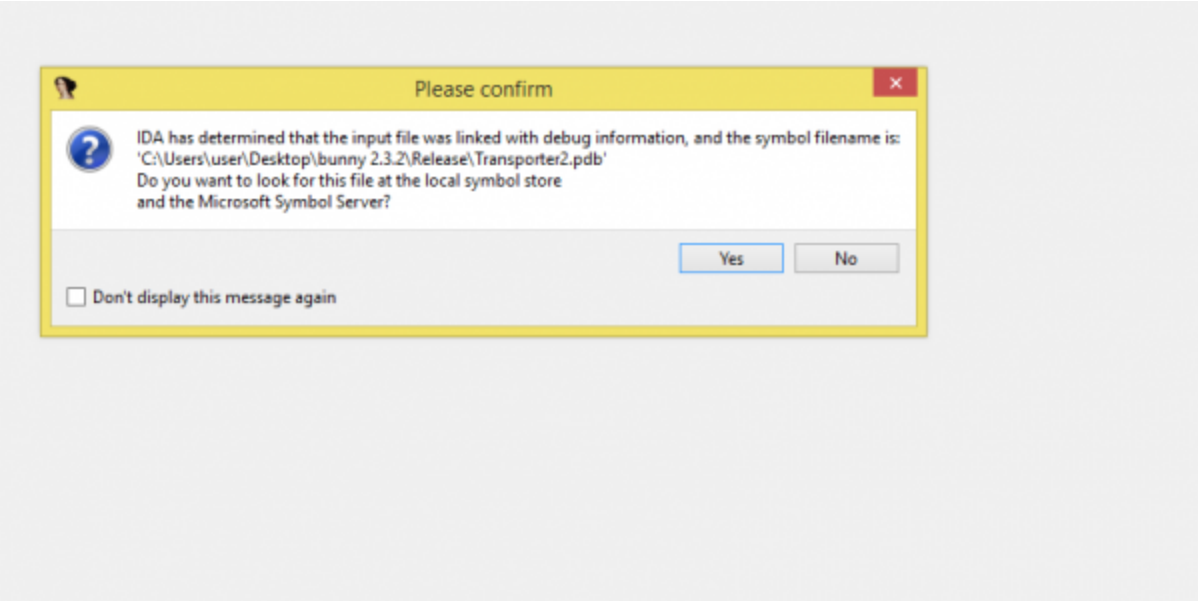
MD5	c40e3ee23cf95d992b7cd0b7c01b8599
SHA-1	1e8b4c374db03dcca026c5feba0a5c117f740233
File Size	943.5 KB (966144 bytes)
Compile Time	2011-10-25 19:28:00

EvilBunny Payload

MD5	3bbb59afdf9bda4ffdc644d9d51c53e7
SHA-1	1798985f4cc2398a482f2232e72e5817562530de
File Size	773.5 KB (792064 bytes)
Compile Time	2011-10-25 19:28:39

THE DROPPER

The EvilBunny malware was originally delivered through a malicious PDF document, exploiting CVE-2011-4369 as reported on http://blog.9bplus.com/analyzing-cve-2011-4369-part-one/. After successful exploitation the malware dropper would be loaded onto the system and infect the machine with EvilBunny.



The functionality of the dropper can be summarized in the following steps:

- Sandbox check and anti-virus product enumeration
- Dropping payload 'netmgr.exe'
- Creating a registry key for persistence
- Creating a registry key for deletion of the dropper

Searching for a sandbox environment, the malware tests the module file name to see if it is less than 5 characters long or if it contains any of the four strings 'klavme', 'myapp', 'TESTAPP' or 'afyjevmv.exe'. Also, it verifies if less than 15 processes are running in the environment, using the API call EnumProcesses. In case any of the conditions is met execution will abort.

Accessing the systems WMI (Windows Management Interface) the malware queries the installed AntiVirus software by issuing 'SELECT * FROM AntiVirusProduct'. Names of AntiVirus products are represented as hard coded SHA-256 hashes, namely the following:

```
d4634c9d57c06983e1d2d6dc92e74e6103c132a97f8dc3e7158fa89420647ec3

4db3801a45802041baa44334303e0498c2640cd5dfd6892545487bf7c8c9219f

bfe74ca464620a62f11b8c47a3778bb132d84fecd90ce7c75817970f2eeeca51  Antivirus

443b6fb65fa57d57ee3113e48e9b4ed1db2921d5352e27fa85064cd60553c3ff  BitDefender

e1625a7f2f6947ea8e9328e66562a8b255bc4d5721d427f943002bb2b9fc5645

588730213eb6ace35caadcb651217bfbde3f615d94a9cca41a31ee9fa09b186c

f1761a5e3856dceb3e14d4555af92d3d1ac47604841f69fc72328b53ab45ca56  Kaspersky
```

If any of the indicated products is installed and active on the machine execution will abort.

The dropper will place the EvilBunny malware under %APPDATA%\Perf Manager\ or %WINDIR%\msapps\; depending whether the dropper is running with administrative privileges or not. Persistence for the dropped payload is achieved by a registry key under [HKLM|HKCU]\…\CurrentVersion\Run which points to the dropped binary named netmgr.exe.

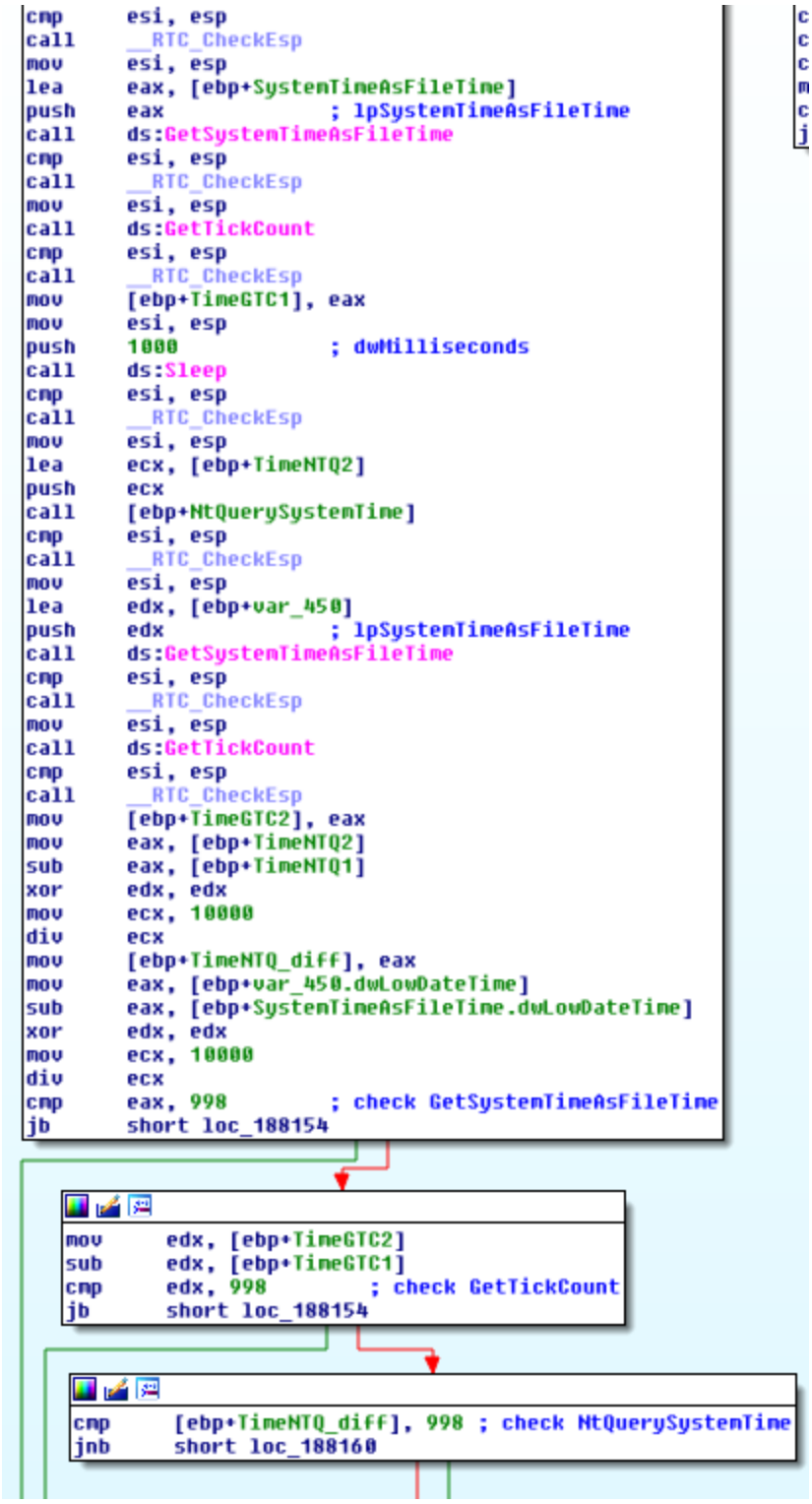
EVIL BUNNY

EvilBunny is a multi-threaded bot with an integrated scripting engine. It incorporates a Lua engine and downloads and executes Lua scripts to reach a certain level of polymorphism. The Lua scripts can call back into the C++ code to alter the malware behavior at runtime.

The malware seeks to keep a low profile on the infected machine, while executing the botmaster's commands and Lua scripts. In total Suspect #4 exhibits three different methods for receiving C&C input and executing commands; directly via HTTP, through a downloaded database file or as a scheduled task. Also, the malware will generate numerous files to help its execution and frequently reply back to the C&C with status messages.

The initial purpose of the malware seems to be sharing execution load among infected host machines. However, due to the lack of the original Lua scripts and the extensive functionality of the embedded Lua engine the original intentions of the attackers remain unknown.

Similar to its dropper, the binary seeks to evade sandboxes. In addition to the previously described trick EvilBunny performs hook detection to trick environments which hook time retrieval APIs. These are NtQuerySystemTime, GetSystemTimeAsFileTime and GetTickCount. Every API is called twice to calculate a delta, while performing a sleep(1000) operation between iteration one and iteration two. The final condition is, if any of the three deltas is below 998 milliseconds execution will abort. This can only be the case if any of the three API's return values is modified by a system monitoring solution, like a sandbox.



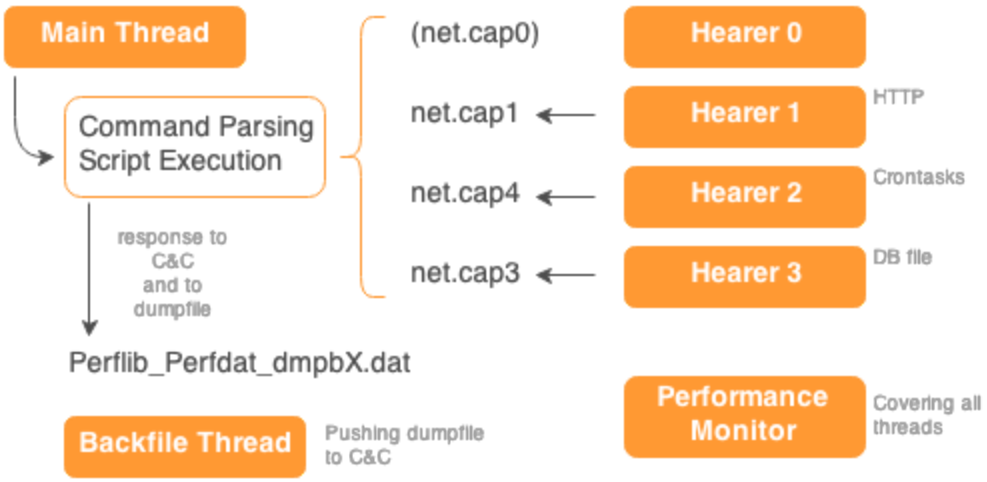
At start-up netmgr.exe decrypts a configuration file stored in its resource section, revealing three URLs, among timeout settings and encryption keys:

- http://le-progres.net/images/php/test.php?rec=11206-01
- http://ghatreh.com/skins/php/test.php?rec=11206-01
- http://www.usthb-dz.org/includes/php/test.php?rec=11206-01

All three of these URLs served as C&C contacts when the attack was still ongoing, sending commands or Lua scripts to the infected host. Two of them, le-progres.net and usthb-dz.org, are now sinkholed by Kaspersky Labs.

THE THREADING MODEL

EvilBunny comes with a solid multi-threading model, which seeks to assure fail-safe and high-performance execution. The malware runs a main thread, which manages four worker threads and performs C&C command parsing and Lua script execution. The worker threads are dedicated to receive commands and scripts through different ways. Next to that, the main thread also runs sub threads to maintain log files the malware creates during execution and to keep track of the overall system load the malware creates. The worker threads are internally dubbed 'hearer', which is believed to stand for 'listener'. It can be concluded thereafter that the malware authors were no English native speakers.



The main action of the malware is carried out in the main thread, which parses commands and executes Lua scripts, provided by the worker threads via command files.

The hearer thread's purpose is to receive instructions from the remote servers and provide them to the main thread. Such instructions are commands and/or one or more Lua scripts. Each hearer has a dedicated method to receive instructions which is either separately via HTTP from the server, aggregated through a downloaded data file or as tasks to be configured as scheduled tasks. The hearer threads dump the received instructions to their associated net.cap-files, from where the main thread's command parsing routine fetches and executes them.

LUA MAGIC

EvilBunny incorporates an interpreter for Lua 5.1, LuaSocket 2.0.2 and C/Invoke Lua bindings. Lua is a lightweight programming language designed as a scripting language which can be embedded into applications, providing a C API for doing so. C bindings are provided through C/Invoke and enable Lua scripts to perform callbacks to C/C++ code. This

constellation can be found in many video game engines to provide polymorphic behavior in games. Engine and game play features are injected through Lua scripts, which instrument the game engine code.

The Lua interpreter is very small, compiled roughly 180kB, thus can easily be integrated in an application. The C/Invoke bindings enable Lua to be completely independent from the C/C++ application, so injected scripts can be pure Lua code.

The Lua interpreter is a powerful code base which enables EvilBunny to change functionality on the fly, as different scripts are downloaded and executed. The scripts define the functionality as they perform callbacks to the C/C++ code in the malware binary. In general this is a rather uncommon technique, but it has been observed before, especially in connection with some adware variants. Lua scripts are text based, which under certain conditions might be easier to tunnel through intrusion detection and firewalls in place than binary content. Also the scripts are much smaller than an entire binary, which might be used for updating the malware. These scripts are about 5-10Kb if big, while a binary has around 50-200Kb or more.