MALWARE ANALYSIS 3

DISSECTING WINLOCKER – RANSOMWARE GOES CENTRALIZED

Aditya K. Sood, Richard J. Enbody
Michigan State University, USA

Rohit Bansal
Independent security researcher, USA

Winlocker, aka Gimemo, has revolutionized the design of ransomware. Before digging deep into the design of Winlocker, let’s talk briefly about ransomware. As the name suggests, this class of malware forces the user to pay a sum of money in order to regain control of the infected system. Ransomware locks down certain functionalities of the operating system (or even the whole operating system, depending on the design) as well as software running on the infected machine. When a user tries to interact with the system, the malware is activated and demands a ransom.

We have already seen many different types of ransomware, but Winlocker is the first we have seen that is centralized in nature – all the infected machines are controlled using two different Command and Control (C&C) panels. One of the C&C panels is used for verification of the transaction generated to pay the ransom. If the transaction is verified and the required amount is transferred to the attacker’s e-currency account, an email is sent to the attacker to unlock the infected system. The other C&C panel is used for managing the administrative operations such as sending unlock commands to the infected system. Use of a centralized platform to manage the ransomware has enabled the attacker to build a crimeware service that can be sold in the underground market. Winlocker’s creator has already started an Insidious Winlocker Affiliate Program (IWAP) in which Winlocker is provided as a crimeware service. Buyers of the service share access to the C&C panel that monitors successful infections for ransom payments – they do not have access to the administrative control panel. Figure 1 shows the C&C panel that is shared with the buyers under the affiliate program.

WINLOCKER WORKING FLOW

Unlike traditional ransomware, Winlocker does not install as a disguised program that is listed in the Add/Remove programs tool. Winlocker is a sophisticated ring 3 layer rootkit that executes nefarious operations. Winlocker performs API hooking to circumvent the communication flow of the target processes and then injects malicious hooks to control the execution. This makes Winlocker much more powerful, which allows it to lock the operating system completely. Winlocker bypasses the User Account Control (UAC) and Data Execution Prevention (DEP) protection schemes very easily. It works successfully on almost all versions of Windows including XP, Vista and Windows 7 on both x32 and x64 systems.

The working flow is described as follows:

• Regular malware infection frameworks such as botnets, browser exploit packs, etc. are used to spread the Winlocker ransomware across the Internet.

• Winlocker is wrapped in a dropper that deletes itself after successful installation of Winlocker in the system, as shown in Figure 2. The dropper looks for the %COMSPEC% environment variable to get the full path and uses ‘/c del’ batch commands to delete itself by redirecting the output to ‘>> NUL’. Winlocker executes instantly and locks the operating system completely.

• Winlocker is installed in the ‘C:\ProgramFiles\System\’ folder as a file named system.exe with an associated
file, Key.txt, as shown in Figure 3. The filename might vary with different versions of Winlocker. The Key.txt file contains certain configuration and system-related information that is required to restore the system later on.

- Winlocker displays a ransom page which is built using a custom template that is based on the Windows Active Template Library (ATL) at the backend to communicate with the C&C server. (The dialog creation and design will be discussed later.) The user is forced to provide an access code to unlock the system. To get the access code, the user has to go to a third-party service provider that charges a few dollars and generates the access code. This code must be entered into the Winlocker ransom template to unlock the system. Direct credit card transactions are not allowed on the infected system. Figure 4 shows MoneyPak [1] being used as an e-currency for the transaction. As soon as the money is received by the attacker, the unlock command is issued from the C&C panel.

Figure 3: Folder and file generation.

A number of different Winlocker templates are used in different countries, as listed in [2].

Figure 4: Winlocker in action.
WINLOCKER DIALOG GENERATION

Winlocker generates a custom dialog to be shown to the user when the system is locked. Winlocker uses the standard built-in Windows APIs to design the dialog. Let’s see what kind of functions are used:

- The dialog is generated using the ShowDialog function which is called when WM_INITDIALOG is dispatched by the system handler. The ShowDialog function reveals the full screen to the user.
- The GetWindowLongA function is used to retrieve the style of the dialog. The SetWindowLongA function is used to remove all the extra header objects, such as buttons, from the dialog.
- Using the RegisterHotKey function, shortcuts such as ALT-TAB are disabled. The SetWindowsPos function is deployed to force the dialog box to be displayed on top (setting the position) of all other running windows. The SetForegroundWindow function sets the ransom dialog in the foreground.
- Using the GetDlgItem and MoveWindow functions, Winlocker restricts the resizing of the window when WM_SIZE is dispatched by the system handler.
- Winlocker is finally activated and displayed on top of all other windows when the WM_TIMER message is dispatched. To do this, Winlocker enumerates all the running windows using EnumWindows to obtain the handles which are required to put all other windows in the background. It also uses SetWindowsHookEx to handle the different kind of keys to be used in the ransom dialog.

Winlocker also uses a primary function from the Active Template Library (ATL) [3] which registers a window

```c
hModule = LoadLibraryA("atl32.dll");
hAddr = GetProcAddress(hModule, "AtlAxWinInit");
hAddr();
HWND hWnd = GetModuleWindowA(0);
CreateDialogParamA(hWnd, (LPCSTR)0x3E8, 0, DialogFunc, 0);
while (GetMessageA(&msg, 0, 0, 0))
{
    TranslateMessage(&msg);
    DispatchMessageA(&msg);
}
hFree = FreeLibrary(hModule);
ExitProcess(0);
```

*Listing 1: AtlAxWinInit loading.*

Figure 5: AtlAxWin class with control object.

class that is used to host ActiveX controls. Basically, it is used to define different controls. Winlocker registers its inherent window class using the AtlAxWinInit function as shown in Listing 1. Figure 5 (Resource Hacker) shows that the AtlAxWin class is used by Winlocker to register a control object that carries a reference to the remote C&C panel.

WINLOCKER CHARACTERISTICS

In this section, we take a look at some of the modifications performed by the Winlocker ransomware in the system.

SafeBoot – safe mode modification

As Winlocker is designed specifically for ransom purposes, its functionality is very targeted in nature. Winlocker actually deletes all the entries present in the registry hives that relate to safe mode booting. The SafeBoot [4] option in the registry usually has two sub entries. The ‘minimal’ SafeBoot option allows the minimum set of device drivers to be loaded in safe mode. The ‘network’ SafeBoot option allows the system to have the minimum set of device drivers and networking capabilities during safe mode. Winlocker actually deletes the entry in the following key: ‘HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal’ and ‘HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network’. Some of the deleted entries are presented in Listing 2.

System restore – modification

The system restore functionality plays a significant role in the success and demise of the ransomware. The malware authors have to manage the system restore capability for successful infection and control of the ransomware in the system. As the name suggests, system restore allows
Listing 2: List of SafeBoot entries deleted by Winlocker.
### Validating installation using HTTP

Once the system is infected, Winlocker connects back to the C&C server. It sends a GET request to receive the notification that the C&C panel has actually established a connection with it. Listing 3 shows the request sent by Winlocker with user-specific information. The user information plays a critical role because certain functionalities of Winlocker are dependent on this information (for example, if Winlocker is installed on a machine with administrator access, it will infect all the other users on the system as well). The C&C server sends the HTTP response as valid. As a result, Winlocker executes the ransom template after locking the system. The usual pattern of the request is:

```
http://<IP Address>/c35312f03a7e0557a44db2326bd29040/k.php?i=4u2RejXq9bKEBr0PJ6u2TgkYzVbMhGh30Re6wp8hKExVn0I4u2RejXq9bMB&us=Administrator&cl=de&f=0&a=aff_3556
```

Here, we have looked at the primary characteristics of Winlocker. In [5], a researcher has reversed the Winlocker builder – this may prove useful for writing Winlocker patches.

### CONCLUSION

In this paper, we have discussed the design and behaviour of the Winlocker ransomware. At this point in time, Winlocker infects machines with the collection of a ransom payment as its only goal. It has copied a standard design used by botnets and become centralized. In reality, Winlocker is a popular crimeware service in the underground market.

### REFERENCES