Popureb
A small rootkit with a big reputation

Mike Wood, Michele Freschi and Ahmed Zaki
SophosLabs, Sophos Inc.

July 8, 2011

Abstract

Popureb is a rootkit for the Windows platform that affects the Master Boot Record (MBR) of the infected disk. Once installed, the rootkit has three main components: the malicious MBR to load the rootkit at boot time, the driver to protect the malicious MBR from alteration, and the malicious agent loaded by the driver. While it is technically inferior to other MBR-based rootkits with respect to obfuscation, reliability, and self-protection techniques, Popureb has received significant media attention recently. Several online articles have suggested Popureb requires a complete reinstallation of Windows to clean an infected computer. On the contrary, remediation is relatively straightforward and the following technical analysis of Popureb is intended to dispel any lingering myths regarding its operational abilities.
1 Introduction

Popureb is a rootkit for the Windows platform. The rootkit hijacks the computer at boot time using a rogue Master Boot Record (MBR) to decrypt and execute two additional components: a driver to protect the rogue MBR, and a malicious agent to carry out nefarious behaviour.

Popureb is installed by a standalone dropper, implemented as a single, self-contained Windows program that runs from userland. The dropper is responsible for making all of the initial system changes to bootstrap the infection, including re-writing the MBR and installing the rootkit driver. Notably, the droppers examined to date do not attempt any privilege escalation exploits to carry out these operations. As such, running a Popureb dropper from a limited user account (i.e. a non-Administrator user account) does not result in a successful infection.

The dropper’s inability to infect computers from a limited account is one of the many inferiorities of Popureb when compared to other modern rootkits. In fact, several components of the infection, including the driver object and the agent process, remain easily accessible via standard forensics tools. For example, the driver object can be found in the list of Driver objects by Microsoft’s WinObj.exe[1], and the agent can be both seen and killed via ProcessExplorer.

As rootkits go, Popureb is not very good at hiding or protecting itself. But it does employ some stealthy and self-protecting techniques, so it qualifies as a rootkit nevertheless.

2 Popureb Operation

This section details the operation of the Popureb rootkit, including installation, infection persistence, self-protection techniques (also known as armouring) and malicious activity.

2.1 Popureb Dropper

The dropper is very simple and straightforward. It is not packed and does not include any anti-emulation or anti-debugging code. A simple control flow obfuscation is used, which is a minor hiccup for statically analyzing the code.

```c
je continue
jne continue
db 0E9h ; or 0E8h
continue:
```

The dropper’s aim is substituting the MBR and saving the other components (the bootkit loader, the driver and the main agent) on disk.
In detail, the Popureb dropper:

- writes the driver, the main agent, a copy of the original MBR and the bootkit code (Fig 1) to disk at 0x34600 bytes\(^1\) from the end of the disk. The whole blob is scrambled with a ROL 3 operation.

![Figure 1: Disk blob](attachment:image.png)

<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000</td>
<td>Bootkit loader</td>
</tr>
<tr>
<td>0x002D5</td>
<td>Driver</td>
</tr>
<tr>
<td>0x02400</td>
<td>Original MBR</td>
</tr>
<tr>
<td>0x02600</td>
<td>Main Agent</td>
</tr>
</tbody>
</table>

- creates the new MBR from the original one, substituting the code area with the malicious one and patching inside the offset to the blob. The rest (from 0x1B0 on) is left intact, thus preserving the partition table and the boot sector signature.

- drops the driver in the current directory with the filename `hello_tt.sys` and attempts the following series of actions to load the driver into memory and subsequently cover up its tracks:
  - creates and starts a service named “hello_tt” for the driver. This runs the Popureb driver to setup the hook to protect the Pop-

---

\(^1\)0x34600 = 0x2600 bytes for the bootkit loader, the driver and the MBR + 0x32000 bytes for the agent (with padding)
ureb MBR. Note that the service fails to start for some Popureb droppers that corrupt the hello_tt.sys file. In this case, the Popureb driver is not loaded and the Popureb MBR is left unprotected until the system is rebooted (see Section 2.3).

– stops and deletes the “hello_tt” service. This removes the registry entries to cover up the tracks of the driver’s execution above. Note that the service fails to be stopped for some Popureb droppers when either the handle to control the service lacks the SERVICE_CONTROL_STOP permission or when the driver lacks a DriverUnload routine. In both these cases, the registry entries are not removed and persist across reboot.

– deletes hello_tt.sys. Note that driver file is deleted regardless of the “hello_tt” service status.

• starts the main agent process.

The overall behaviour of the dropper suggests that little effort has been put into covering its tracks and into the robustness of its operations:

• The driver is often not removed from the services database.

• The dropper itself is not deleted.

• The dropper will fail to infect when run from a limited user account (i.e. a non-administrator account).

• The dropper does not check if it is overwriting already-allocated sectors when writing the payload to disk. Any original data from these sectors is unrecoverable – irreparably damaging the file (or other system component) to which they were allocated.

2.2 Popureb Master Boot Record

The Popureb MBR is responsible for starting the Popureb rootkit on an infected computers. The startup process has three main stages:

• reading and decrypting the Popureb blob from the end of the disk, to reveal the bootkit loader, driver and original MBR

• setting up the chain of system hooks used to load the driver when Windows boots

• transferring control to the original MBR to continue the boot process

To fetch its encrypted data, the Popureb MBR has the zero-based absolute sector number of the blob, as well as the number of disk sectors to read, hard-coded within the MBR itself. The MBR reads and decrypts the
bootkit loader, the driver and original MBR. The agent is decrypted and loaded by the driver (see Section 2.3), while the driver is loaded by the hooks installed by the bootkit loader code. Note that the PE image for the agent has its 'MZ' word zeroed as stored on disk. The offline disk layout for a Popureb-infected computer is illustrated in Figure 2.

The bootkit loader is responsible for setting up the chain of hooks necessary to load the Popureb driver when Windows boots. Firstly, the \texttt{int 13} handler is hooked to filter for functions 02h and 42h, which are disk read and extended disk read respectively. This hook monitors disk reads until \texttt{ntldr} is being loaded. The hook then patches the raw \texttt{ntldr} with a call back into the bootkit loader, thus subverting the bootstrap process. From there, the base address of \texttt{ntoskrnl.exe} is found using the \_\texttt{BlLoaderBlock} data structure. Once the base address is retrieved, the bootkit uses a custom hashing algorithm to search for the \texttt{PsGetCurrentProcess} export and install yet another hook. This \texttt{PsGetCurrentProcess} hook is responsible for creating the system thread and fixing the patch in the API so that it does not get called again. This system thread will be responsible for replacing the original Microsoft driver \texttt{beep.sys} with the malicious driver. The malicious \texttt{beep.sys} gets loaded automatically by Windows as part of the system startup. Notably, this bootkit hooking process is identical to that of “BootRoot”, documented in early January of 2011 [2].

The control transfer to the original MBR occurs once the disk read hooks are in place. As described above, the disk read hooks ensure Popureb is able to hijack the normal Windows boot kicked off by the original MBR.

2.3 Popureb Driver

The Popureb driver has two main responsibilities: protecting the MBR from modification, and starting the main agent. Note that the driver protects only the MBR. The sectors containing the encrypted blob at the end of the disk can be written to as usual.

The Popureb driver performs three key operations:

- drops the main agent on disk and creates an entry in autorun registry key for it
• deletes **beep.sys** from the drivers directory, triggering in some cases a Windows File Protection alert

• hooks the **StartIO** function of the lowest miniport driver, usually at-api, to prevent writes to MBR space.

The hook function filters only **IRP_MJ_INTERNAL_DEVICE_CONTROL** packets and specifically only SCSI read/write operations. After checking that the read/write offset falls inside the protected area, the driver changes the operation type from **SCSIOP_WRITE** to **SCSIOP_READ**, and passes control to the original **StartIO** function to complete the operation.

### 2.4 Popureb Agent

Due to the claims that Popureb required a Windows reinstallation\(^1\), our technical analysis of Popureb has largely focused on the operational aspects of the rootkit and remediation. The following description of the Popureb agent is therefore brief.

The primary purpose for the Popureb agent appears to be generating traffic to online advertisements. On startup, a mutex is used to ensure only a single instance of the agent runs. The mutex is also intended to prevent the dropper from re-running on an already infected computer. However, as the mutex names can be different across different Popureb dropper-agent pairs, a system may still be corrupted by multiple Popureb dropper executions (see Section 3.3).

The agent’s functionality is grouped into a set of seven or eight threads. Each thread has a similar structure, performing some initialization before settling into a loop performing a specific task driven by the current agent’s configuration data. Some of the tasks performed by the agent include:

- updating the agent’s configuration data from a remote site

- browsing URIs via Internet Explorer or COM objects, as specified in the configuration data

- downloading and executing further PE executables, as specified in the configuration data

The following filenames are known to be used by Popureb agents:

- `c:\alg.exe`

- `%WINDIR%\Temp\ali.exe`

Some artifacts indicating the presence of a Popureb agent include:

\(^1\)For details, see Section 3.1
• mutex names IE_2011_Mutex or PreventSecondInstance

• two files in `%ALLUSERS%\Documents\My Videos`:
  – `<random>.tmp` which is a copy of itself marked as System and Hidden
  – `PulgFile.log` INI file holding part of the configuration

• two files in `%WINDIR%\Temp`, one .exe and one .ini with the same prefix (e.g. ali.exe and ali.ini)

• the `%WINDIR%\VC.ini`, part of the ad-clicking configuration data

Some of the domains referenced by the Popureb agents examined to date include dh818.info, 765321.info, dy213.com, and uuying.com. Notably, three of these domains are registered to the same person in China.

3 Remediation

All that is required to clean a Popureb-infected system is to restore the original host MBR. Recall that both the driver and the agent components are only stored on-disk in their ROL-encrypted form, and must be dynamically loaded during boot to operate. Thus, removing the Popureb MBR from the boot process turns the Popureb encrypted components into a benign binary blob.

3.1 Is A Windows Reinstallation Necessary?

There is considerable confusion and hype regarding Popureb remediation. Early reports dramatically claimed a complete Windows reinstallation was required [4]. Although Microsoft has since clarified remarks from its initial blog on Popureb [3], follow-on articles continue to over-exaggerate its stealth- ing capabilities, some even continuing to suggest the computer needs to be “rebuilt” [6].

There is also the misperception that an MBR-based rootkit guarantees that it and all its derivative components are “invisible to both the operating system and security software” [5]. However, an MBR-based rootkit simply uses the MBR as the entry point to start its stealthing components. Any subsequent stealthing behaviour is dependent on the behaviour of those components – they are not hidden simply by virtue of the fact they were loaded via the MBR. As the technical analysis above demonstrates, there are several gaps in the runtime stealthing behaviour of Popureb.
3.2 Cleanup Options

To restore the MBR following a Popureb infection, options include:

**Sophos Bootable Anti-Virus (SBAV)**

Detects the Popureb MBR, restores the original host MBR, and wipes out the Popureb encrypted blob from the end of the disk. Visit the following URI for instructions on creating an SBAV image: [http://www.sophos.com/support/knowledgebase/article/52011.html](http://www.sophos.com/support/knowledgebase/article/52011.html)

**fixmbr via Windows Recovery Console**

As referenced in Microsoft’s original blog on Popureb [3], this process uses bootrec.exe via a Windows Recovery Console to restore the original host MBR.

3.3 System Corruption by Popureb Droppers

The Popureb droppers examined to date do not perform a reliable check to determine if there is an existing Popureb infection. If multiple Popureb droppers are executed or a single dropper is executed multiple times, the second dropper may corrupt the system to the point that the computer will no longer boot. The disk will fail to boot if the second (or subsequent) Popureb dropper replaces the original host MBR stored within the Popureb encrypted blob with another Popureb MBR. The disk will fail to boot as the bootkit loader relies on the original MBR to continue loading Windows (see Section 2.2). In this scenario, the original MBR must be restored via the Windows Recovery console method.

4 Discussion

Popureb appears to be an evolving malware family aimed at carrying out click-fraud. The domains referenced suggest the malware is targeted towards Chinese websites (and users). This falls directly in line with what appears to be an earlier iteration of Popureb dubbed BootRoot – first seen in late December of 2010 – in which 99% of infected users reside in China [2]. Notably, this earlier version of the rootkit lacks the driver component and uses the bootkit loader to start the agent directly. The earliest samples containing the driver component were recorded in the SophosLabs malware collection in mid-April 2011 – a few hundred MBRs, drivers, agents and droppers combined.

Multiple factors suggest the rootkit aspects of Popureb may still be in their early stages of development, including its shortcomings in stealthing, and its operational unreliability. Whether Popureb evolves into a more robust rootkit or whether it lingers on as a nuisance remains to be seen.
But whatever the future may hold, the hype around Popureb up to this point has been largely unjustified.
References


