Flame: Bunny, Frog, Munch and BeetleJuice…

By Alexander Gostev

Published: 2012-05-30 · Archived: 2026-04-05 22:27:59 UTC

As already mentioned in the previous blog post about Flame, the volume of its code and functionality are so great that it will take several months for a complete analysis. We're planning on continually disclosing in our publications the most important and interesting details of its functionality as we reveal them.

At the moment we are receiving many inquiries about how to check systems for a Flame infection. Of course the simplest answer, for us, is to advise to use Kaspersky Lab Antivirus or Internet Security. We successfully detect and delete all possible modifications of the main module and extra components of Flame.

However, for those who want to carry out a detailed check themselves, at the end of this article we will give the necessary recommendations and advice.

MSSECMGR.OCX

The main module of Flame is a DLL file called mssecmgr.ocx. We've discovered two modifications of this module. Most of the infected machines contained its "big" version, 6 Mb in size, and carrying and deploying additional modules. The smaller version's size is only 900 Kb and contains no additional modules. After installation, the small module connects to one of the C&C servers and tries to download and install the remaining components from there.

Mssecmgr may be called different names on actual infected machines, depending on the method of infection and the current internal state of the malware (installation, replication, upgrade), e.g., wavesup3.drv, ~zff042.ocx, msdclr64.ocx, etc.

Complete analysis of the mssecmgr module will follow in our upcoming blog posts.

The first activation of this file is initiated by one of the external features – either Windows WMI tools using a MOF file if the MS10-061 exploit is used, or using a BAT file:

```
s1 = new ActiveXObject("Wscript.Shell");
s1.Run("%SYSTEMROOT%system32rundll32.exe msdclr64.ocx,DDEnumCallback");
```

(source code of MOF file, svchostevt.mof)

When activated, mssecmgr registers itself as a custom authentication package in the Windows registry:

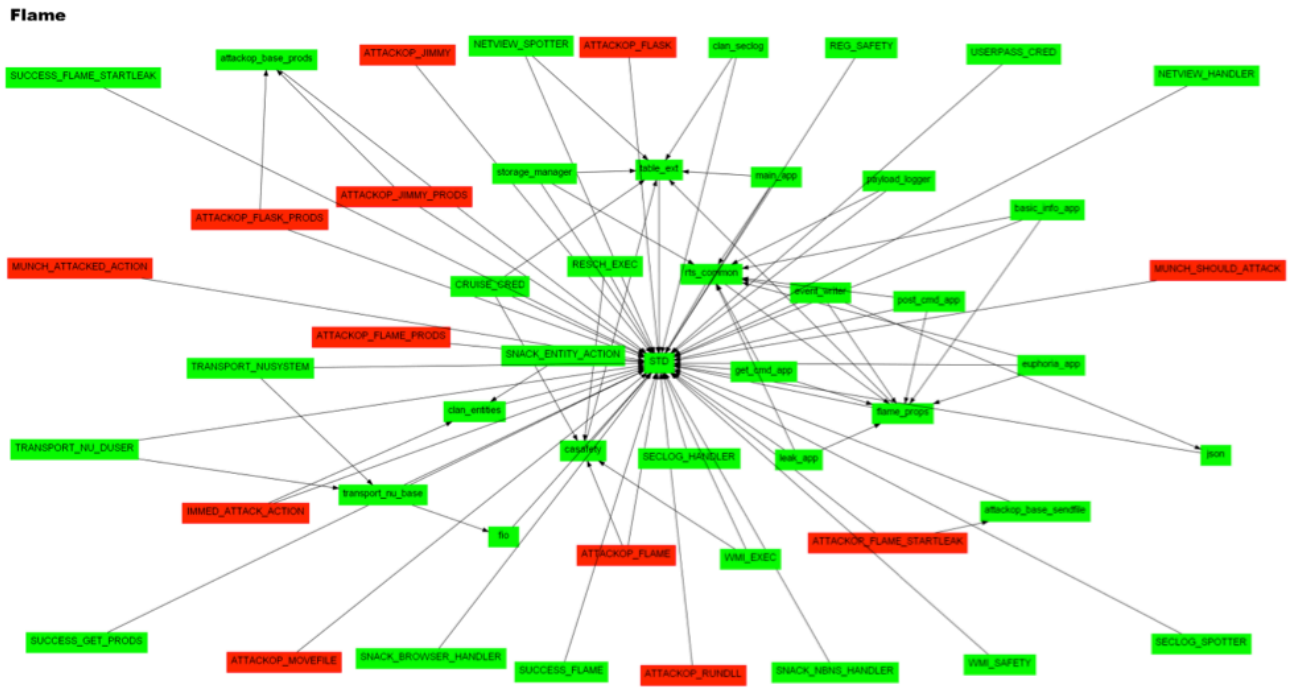
HKLM_SYSTEMCurrentControlSetControlLsa
Authentication Packages = mssecmgr.ocx [added to existing entries]

On the next system boot, the module is loaded automatically by the operating system.

After updating the Windows registry, mssecmgr extracts any additional modules that are present in its encrypted and compressed resource section (resource “146”) and installs them. The resource is a dictionary that contains configuration options for mssecmgr and other modules, the modules themselves (DLL files), and parameters that need to be passed to these modules to load them properly, i.e., decryption keys.

We are analyzing the additional modules and will provide more information about their functionality in coming blog posts.

When installation is completed, mssecmgr loads available modules and starts several execution threads that implement a channel to the C&C servers and Lua interpreter host, and other features – depending on the configuration. The functionality of the module is separated into different “units” that have different namespaces in the configuration resource and have distinct names in log messages, which are extensively used throughout the code.



Here is a brief overview of the available units. The names were extracted from the binary and the 146 resource.

Beetlejuice	Bluetooth: enumerates devices around the infected machine. May turn itself into a “beacon”: announces the computer as a discoverable device and encode the status of the malware in device information using base64.
Microbe	Records audio from existing hardware sources. Lists all multimedia devices, stores complete device configuration, tries to select suitable recording device.
Infectmedia	Selects one of the methods for infecting media, i.e. USB disks. Available methods: Autorun_infector, Euphoria.

Autorun_infector	Creates “autorun.inf” that contains the malware and starts with a custom “open” command. The same method was used by Stuxnet before it employed the LNK exploit.
Euphoria	Create a “junction point” directory with “desktop.ini” and “target.lnk” from LINK1 and LINK2 entries of resource 146 (were not present in the resource file). The directory acts as a shortcut for launching Flame.
Limbo	Creates backdoor accounts with login “HelpAssistant” on the machines within the network domain if appropriate rights are available.
Frog	Infect machines using pre-defined user accounts. The only user account specified in the configuration resource is “HelpAssistant” that is created by the “Limbo” attack.
Munch	HTTP server that responds to “/view.php” and “/wpad.dat” requests.
Snack	Listens on network interfaces, receives and saves NBNS packets in a log file. Has an option to start only when “Munch” is started. Collected data is then used for replicating by network.
Boot_dll_loader	Configuration section that contains the list of all additional modules that should be loaded and started.
Weasel	Creates a directory listing of the infected computer.
Boost	Creates a list of “interesting” files using several filename masks.
Telemetry	Logging facilities
Gator	When an Internet connection becomes available, it connects to the C&C servers, downloads new modules, and uploads collected data.
Security	Identifies programs that may be hazardous to Flame, i.e., anti-virus programs and firewalls.
Bunny Dbquery Driller Headache Gadget	The purpose of these modules is not yet known.

Additional modules are installed in the %windir%system32 directory:

mssecmgr.ocx
advnetcfg.ocx
msglu32.ocx
nteps32.ocx

soapr32.ocx
ccalc32.sys
boot32drv.sys

Additional modules downloaded from the C&C servers are installed in the same directory.

The modules of the Flame malware produce a lot of data files that contain extensive execution logs, plus the collected information – screenshots, lists of processes, hardware listings, etc.

The files are saved in the %windir%temp directory using following names:

~DEB93D.tmp
~8C5FF6C.tmp
~DF05AC8.tmp
~DFD85D3.tmp
~DFL.tmp*
~dra.tmp*
~fghz.tmp
~HLV.tmp*
~KWI988.tmp
~KWI989.tmp
~rei524.tmp
~rei525.tmp
~rf288.tmp
~rft374.tmp
~TFL848.tmp
~TFL849.tmp
~mso2a0.tmp
~mso2a1.tmp
~mso2a2.tmp
sstab.dat*

There are also optional files that can be found in the %windir%system32 directory:

Advpck.dat
ntaps.dat
Rpcnc.dat

Also, in %windir%:

Ef_trace.log

Flame stores its persistent data in files using different formats. All data is encrypted using several algorithms and keys. Some of the files are databases created using a built-in SQLite3 library. These databases contain data

produced by both data stealing and replication routines. We will provide more information about these files in later publications.

The directory names used by the additional components of Flame may differ slightly depending on the type of installation and the configuration options from resource 146:

C:Program FilesCommon FilesMicrosoft SharedMSSecurityMgr
C:Program FilesCommon FilesMicrosoft SharedMSAudio
C:Program FilesCommon FilesMicrosoft SharedMSAuthCtrl
C:Program FilesCommon FilesMicrosoft SharedMSSndMix

These directories may contain the following files:

dstrlog.dat
lmcache.dat
mscrypt.dat (or wpgfilter.dat)
ntcache.dat
rccache.dat (or audfilter.dat)
ssitable (or audache)
secindex.dat
wavesup3.drv (a copy of the main module, mssecmgr.ocx, in the MSAudio directory)

Flame can also produce or download files with the following names:

svchost1ex.mof
Svchostevt.mof
frog.bat
netcfgi.ocx
authpack.ocx
~a29.tmp
rdcvlt32.exe
to961.tmp
authcfg.dat
Wpab32.bat
ctrllist.dat
winrt32.ocx
winrt32.dll
scsec32.exe
grb9m2.bat
winconf32.ocx
watchxb.sys

sdclt32.exe
scaud32.exe
pcldrvx.ocx
mssvc32.ocx
mssui.drv
modevga.com
indsvc32.ocx
comspol32.ocx
comspol32.dll
browse32.ocx

As a consequence, we can provide a method for a quick "manual" check of your systems for presence of a Flame infection:

1. Perform a search for the file **~DEB93D.tmp**. Its presence on a system means that it either is or has been infected by Flame.

2. Check the registry key HKLM_SYSTEMCurrentControlSetControlLsa
Authentication Packages.
If you find mssecmgr.ocx or authpack.ocx in there – you are infected with Flame.

3. Check for the presence of the following catalogs. If present – you're infected.

C:Program FilesCommon FilesMicrosoft SharedMSSecurityMgr
C:Program FilesCommon FilesMicrosoft SharedMSAudio
C:Program FilesCommon FilesMicrosoft SharedMSAuthCtrl
C:Program FilesCommon FilesMicrosoft SharedMSSndMix

4. Conduct a search for the rest of the filenames given above. All of them are quite unique and their being discovered would mean that there is a strong possibility of an infection with Flame.

P.S. We have checked the information as suggested in the comments at our blogpost regarding a possible relation to FLAME (Flexible Lightweight Active Measurement Environment) software from Brazil.

Interestingly, the name picked by us fully matches that software, which also uses LUA for implementing business logics. The FLAME software is used to measure network characteristics by deploying measurement agents and collecting data in a central database. Despite some similarities, we think that this software is unrelated as it serves different objectives. Besides the LUA engine, the core of communication in FLAME is XMPP protocol, which is not used in the Flame malware.

The authors might have been inspired by the FLAME project and re-implemented similar architecture – only for the different goal, or this is all just a coincidence. We don't have any other reason to think that it is somehow related to the Flame malware.
