Additional technical details re: W32.Flamer--Symantec

Additional technical details re:W32.Flamer--Symantec

Good afternoon

In an effort to keep you posted on developments concerning W.32 Flamer,researchers at Symantec have continued their analysis and have posted the ablog, Painting a Picture of W32.Flamer, which contains furthertechnical details of the various components that make up Flamer. Fullunderstanding of Flamer requires analyzing approximately 60 embedded Luascripts, reverse engineer each or the sub-components, then piece it backtogether. As an analogy, reverse engineering Flamer as opposed tostandard malware is like re-creating an architectural drawing, not just for asingle house, but for an entire city.

While Flamer is a complex threat, the use of Lua, a lightweight scriptinglanguage provides a simple way for the attacker to develop new functionality inthe threat. Full text of the blog is below.

If you have any questions please feel free to let me know and I can put you intouch with a researcher from Symantec Security Response.

Cheers,
James

Painting a Picture of W32.Flamer
BySymantec Security Response

The number of different components in W32.Flamer is difficult to grasp. The threat is a welldesigned platform including, among other things, a Web server, a databaseserver, and secure shell communications. It includes a scripting interpreterwhich allows the attackers to easily deploy updated functionality throughvarious scripts. These scripts are split up into 'apps' and the attackers evenappear to have something equivalent to an 'app store' from where they canretrieve new apps containing malicious functionality.

To get an idea of how all these components fit together, the best place tostart is a file called mssecmgr.ocx. This is W32.Flamer's main file and it isthe first element of the threat executed by an infected computer. The filemssecmgr.ocx contains a large number of sub-components. A breakdown of thevarious components and how they are stored in this file are shown in Figure 1below:

Figure1. Overview of W32.Flamer components

Most sub-components, including the various scripts, are stored in a largeencrypted resource embedded in mssecmgr.ocx. This resource has a tableat the start which lists the various entries and acts as an index to retrievethe data. It is a similar approach to a standard file system. As well ascontaining the scripts, several standalone executable DLL files are alsoincluded in the resource table. Some of these are advnetcfg.ocx,nteps32.ocx, boot32drv.sys, msglu32.ocx, soapr32.ocs, jimmy.dll, 00006411.dll,and more.

Outside of this resources section there exists code to implement a HTTP server,SOCKS proxy, SSH, SQL database and, of course, the Lua interpreter. Lua is alightweight scripting language which was specifically designed for embeddinginto other applications. It is a simple way to extend the functionality of anapplication. It allows a developer to write a script which can then utilize thefunctionality of the program it is embedded into. By using Lua, the attackerhas reduced the effort required when developing new functionality or 'apps' forW32.Flamer.
When msseccmgr.ocxis executed, it uses some convoluted tricks to embed itself into the runningsystem as shown in Figure 2 below:

Figure 2. Initial execution of mssecmgr.ocx

It first extracts and executes advnetcfg.ocx. The mssecmgr.ocx file then passesnteps32.ocx to advnetcfg.ocx which in turn injects nteps32.ocx into the cleanfile shell32.dll. With these files running, mssecmgr.ocx then launches the mainLua 'app' script.
This script starts off by looking for any updated 'app' from the app database.(In the code, the repository of 'apps' is referred to as FLAME.) It alsocontains the ability to upload to the app repository. After this it createsseveral database files to store stolen data and starts running various other'apps'.

The collection of Lua scripts can be split up as shown in Figure 3 below:

Figure 3. Lua scripts used by W32.Flamer

At the bottom, there are library scripts which provide functionality to otherscripts, such as database access or network access. On top of these are thehigher level scripts which perform the malicious actions.

These top level scripts are broken into sets. A brief example of some of theseare:
ATTACKOP - Attack another machine and move onto it using a variety of techniques, including exploitsCASafety - Check for AV softwareCRUISE - Steal credentialsEuphoria - Spread over LNK vulnerability and junction point directorySome of thescripts rely on additional executables to operate. For example The ATTACKOPscripts can call the soapr32.ocx file. This file then steals a range of networkdata from the local computer and then writes it to an RC4 encrypted file.

The ATTACKOP_JIMMY script may use one of the following executables to performits functionality:
mssvc32.ocxmsglu32.ocxjimmy.dllThe above isa brief overview of W32.Flamer and how it is laid out. Full understanding ofW32.Flamer requires analyzing each of the approximately 60 embedded Luascripts, reversing each of the sub-components, and then building this all backtogether. As an analogy, reversing W32.Flamer as opposed to reversing standardmalware is like re-creating an architectural drawing, not just for a singlehouse, but for an entire city.
For moreinformation please contact:JamesCurtis
Max Australia
+61 2 9469 5740
james.curtis@maxaustralia.com.au
DebbieSassine
Symantec Corporation
+61 2 9086 2140
debbie_sassine@symantec.com