IRONGATE ICS MALWARE: NOTHING TO SEE HERE...MASKING MALICIOUS ACTIVITY ON SCADA SYSTEMS

June 02, 2016 | by Josh Homan, Sean McBride, Rob Caldwell | Threat Research, Advanced Malware

In the latter half of 2015, the FireEye Labs Advanced Reverse Engineering (FLARE) team identified several versions of an ICS-focused malware crafted to manipulate a specific industrial process running within a simulated Siemens control system environment. We named this family of malware IRONGATE.

FLARE found the samples on VirusTotal while researching droppers compiled with PyInstaller — an approach used by numerous malicious actors. The IRONGATE samples stood out based on their references to SCADA and associated functionality. Two samples of the malware payload were uploaded by different sources in 2014, but none of the antivirus vendors featured on VirusTotal flagged them as malicious.

Siemens Product Computer Emergency Readiness Team (ProductCERT) confirmed that IRONGATE is not viable against operational Siemens control systems and determined that IRONGATE does not exploit any vulnerabilities in Siemens products. We are unable to associate IRONGATE with any campaigns or threat actors. We acknowledge that IRONGATE could be a test case, proof of concept, or research activity for ICS attack techniques.

Our analysis finds that IRONGATE invokes ICS attack concepts first seen in Stuxnet, but in a simulation environment. Because the body of industrial control systems (ICS) and supervisory control and data acquisition (SCADA) malware is limited, we are sharing details with the broader community.

Malicious Concepts

Deceptive Man-in-the-Middle

IRONGATE's key feature is a man-in-the-middle (MitM) attack against process input-output (IO) and process operator software within industrial process simulation. The malware replaces a Dynamic Link Library (DLL) with a malicious DLL, which then acts as a broker between a PLC and the legitimate monitoring software. This malicious DLL records five seconds of 'normal' traffic from a PLC to the user interface and replays it, while sending different data back to the PLC. This could allow an attacker to alter a controlled process unbeknownst to process operators.

Sandbox Evasion

IRONGATE's second notable feature involves sandbox evasion. Some droppers for the IRONGATE malware would not run if VMware or Cuckoo Sandbox environments were employed. The malware uses these techniques to avoid detection and resist analysis, and developing these anti-sandbox techniques indicates that the author wanted the code to resist casual analysis attempts. It also implies that IRONGATE's purpose was malicious, as opposed to a tool written for other legitimate purposes.

Dropper Observables

We first identified IRONGATE when investigating droppers compiled with PyInstaller — an approach used by numerous malicious actors. In addition, strings found in the dropper include the word "payload", which is commonly associated with malware.

- Both pieces of malware look for a single, highly specific process.
- Both replace DLLs to achieve process manipulation.
- IRONGATE detects malware detonation/observation environments, whereas Stuxnet looked for the presence of antivirus software.
- IRONGATE actively records and plays back process data to hide manipulations, whereas Stuxnet did not attempt to hide its process manipulation, but suspended normal operation of the S7-315 so even if rotor speed had been displayed on the HMI, the data would have been static.

A Proof of Concept

IRONGATE's characteristics lead us to conclude that it is a test, proof of concept, or research activity.

- The code is specifically crafted to look for a user-created DLL communicating with the Siemens PLCSIM environment. PLCSIM is used to test PLC program functionality prior to in-field deployment. The DLLs that IRONGATE seeks and replaces are not part of the Siemens standard product set, but communicate with the S7ProSim COM object. Malware authors test concepts using commercial simulation software.
- Code in the malicious software closely matched usage on a control engineering blog dealing with PLCSIM (https://alexsentcha.wordpress.com/using-s7-prosim-with-siemens-s7-plcsim/ and https://pcplcdemos.googlecode.com/hg/S7PROSIM/BioGas/S7%20v5.5/).
- While we have identified and analyzed several droppers for the IRONGATE malware, we have yet to identify the code's infection vector.
- In addition, our analysis did not identify what triggers the MitM payload to install; the `scada.exe` binary that deploys the IRONGATE DLL payload appears to require manual execution.
- We have not identified any other instances of the ICS-specific IRONGATE components (`scada.exe` and `Step7ProSim.dll`), despite their having been compiled in September of 2014.
- Siemens ProductCERT has confirmed that the code would not work against a standard Siemens control system environment.

Implications for ICS Asset Owners

Even though process operators face no increased risk from the currently identified members of the IRONGATE malware family, IRONGATE provides valuable insight into adversary mindset.

Network security monitoring, indicator of compromise (IoC) matching, and good practice guidance from vendors and other stakeholders represent important defensive techniques for ICS networks.

To specifically counter IRONGATE's process attack techniques, ICS asset owners may, over the longer term, implement solutions that:

- Require integrity checks and code signing for vendor and user generated code. Lacking cryptographic verification facilitates file replacement and MitM attacks against controlled industrial processes.
- Develop mechanisms for sanity checking IO data, such as independent sensing and backhaul, and comparison with expected process state information. Ignorance of expected process state facilitates an attacker's ability to achieve physical consequence without alarming operators.

Technical Malware Analysis

IRONGATE Dropper Family

audiodg.exe in the same directory as the dropper. The dropper then executes the utility using the command audiodg.exe /scomma scxrt2.ini. This command populates the file scxrt2.ini with a comma-separated list of network resources identified by the host system.

The dropper iterates through each entry in scxrt2.ini, looking for paths named move-to-operational or move-to-operational.lnk. If a path is found, the dropper first extracts the Base64-encoded .NET executable scada.exe to the current directory and then moves the file to the path containing move-to-operational or move-to-operational.lnk. The path move-to-operational is interesting as well, perhaps implying that IRONGATE was not seeking the actual running process, but rather a staging area for code promotion. The dropper does not execute the scada.exe payload after moving it.

Anti-Analysis Techniques

Each IRONGATE dropper currently identified deploys the same .NET payload, scada.exe. All but one of the droppers incorporated anti-detection/analysis techniques to identify execution in VMware or the Cuckoo Sandbox. If such environments are detected, the dropper will not deploy the .NET executable (scada.exe) to the host.

Four of the droppers (update.exe$_1$, update_no_pipe.exe$_1$, update_no_pipe.exe$_2$, and update.exe$_3$) detect Cuckoo environments by scanning subdirectories of the %SystemDrive%. Directories with names greater than five, but fewer than ten characters are inspected for the subdirectories drop, files, logs, memory, and shots. If a matching directory is found, the dropper does not attempt to deploy the scada.exe payload.

The update.exe$_1$ and update.exe$_3$ droppers contain code for an additional Cuckoo check using the SysInternals pipelist program, install.exe, but the code is disabled in each.

The update.exe$_2$ dropper includes a check for VMware instead of Cuckoo. The VMWare check looks for the registry key HKLM\SOFTWARE\VMware, Inc.\VMware Tools and the files %WINDIR%\system32\drivers\vmmouse.sys and %WINDIR%\system32\drivers\vmhgfs.sys. If any of these are found, the dropper does not attempt to deploy the scada.exe payload.

The dropper bla.exe does not include an environment check for either Cuckoo or VMware.

scada.exe Payload

We surmise that scada.exe is a user-created payload used for testing the malware. First, our analysis did not indicate what triggers scada.exe to run. Second, Siemens ProductCERT informed us that scada.exe is not a default file name associated with Siemens industrial control software.

When scada.exe executes, it scans drives attached to the system for filenames ending in Step7ProSim.dll. According to the Siemens ProductCERT, Step7ProSim.dll is not part of the Siemens PLCSIM software. We were unable to determine whether this DLL was created specifically by the malware author, or if it was from another source, such as example code or a particular custom ICS implementation. We surmise this DLL simulates generation of IO values, which would normally be provided by an S7-based controller, since the functions it includes appear derived from the Siemens PLCSIM environment.

If scada.exe finds a matching DLL file name, it kills all running processes with the name biogas.exe. The malware then moves Step7ProSim.dll to Step7ConMgr.dll and drops a malicious Step7ProSim.dll – the IRONGATE payload –

return the recorded data.

Simultaneously, the malicious DLL discards all calls to `WriteDataBlockValue` and instead calls `WriteInputPoint(0x110, 0, 0x7763)` and `WriteInputPoint(0x114, 0, 0x7763)` every millisecond. All of these functions are named similarly to Siemens S7ProSim v5.4 COM interface. It appears that other calls to API functions are passed through the malicious DLL to the legitimate DLL with no other modification.

Biogas.exe

As mentioned previously, IRONGATE seeks to manipulate code similar to that found on a blog dealing with simulating PLC communications using PLCSIM, including the use of an executable named `biogas.exe`.

Examination of the executable from that blog's demo code shows that the `WriteInputPoint` function calls with byte indices `0x110` and `0x114` set pressure and temperature values, respectively:

IRONGATE:

```
    WriteInputPoint(0x110, 0, 0x7763)
WriteInputPoint(0x114, 0, 0x7763)
```

Equivalent pseudo code from Biogas.exe:

```
       S7ProSim.WriteInputPoint(0x110, 0, (short)this.Pressure.Value)
    S7ProSim.WriteInputPoint(0x114, 0, (short)this.Temperature.Value)
```

We have been unable to determine the significance of the hardcoded value `0x7763`, which is passed in both instances of the write function.

Because of the noted indications that IRONGATE is a proof of concept, we cannot conclude IRONGATE's author intends to manipulate specific temperature or pressure values associated with the specific biogas.exe process, but find the similarities to this example code striking.

Artifacts and Indicators

PyInstaller Artifacts

The IRONGATE droppers are Python scripts converted to executables using PyInstaller. The compiled droppers contain PyInstaller artifacts from the system the executables were created on. These artifacts may link other samples compiled on the same system. Five of the six file droppers (`bla.exe`, `update.exe`$_1$, `update_no_pipe.exe`$_1$, `update_no_pipe.exe`$_2$ and `update.exe`$_3$) all share the same PyInstaller artifacts listed in Table 1.

C:\Users\Main\Desktop\pyinstaller-2.0\PyInstaller\loader\iu.py	BIN	NA	18,120
A79596BCCA537FA3FA45037F4855FD00			2014-10-27 17:33:57Z
C:\Users\Main\Desktop\pyinstaller-2.0\PyInstaller\loader\archive.py	BIN	NA	16,506
026BC58300DE02455937CEF46405F065			2014-10-27 17:33:57Z

Table 1: Pyinstaller Artifacts

The remaining dropper, `update.exe`$_2$, contains the artifacts listed in Table 2.

Source Path	File Type	Architecture	Size
MD5			**Compile Time**
C:\Python27\Lib\site-packages\PyInstaller\loader\pyi_os_path.py	BIN	NA	2,445
EC07A5ECB182960777007AFE2C077A1D			2014-09-24 07:59:14Z
C:\Python27\Lib\site-packages\PyInstaller\loader\pyi_archive.py	BIN	NA	12,159
9B588ADB1D0AE72CEB4051031FD1F1F3			2014-09-24 07:59:14Z
C:\Python27\Lib\site-packages\PyInstaller\loader\pyi_importers.py	BIN	NA	12,882
EDA021ACACA81AE99E39ECCDA0163295			2014-09-24 07:59:14Z

Table 2: Pyinstaller Artifacts for update.exe$_2$

Unique Strings

Figure 1 and 2 list the unique strings discovered in the `scada.exe` and `Step7ProSim.dll binaries.`

```
dllFilename
newDllFilename
PackagingModule.Step7ProSim.dll
  2014
$ccc64bc5-ef95-4217-adc4-5bf0d448c272
c:\Users\Main\Desktop\PackagingModule\PackagingModule\obj\Release\Packa
gingModule.pdb
```

Figure 1: Scada.exe Unique Strings

```
Step7ProSim.dll
IStep7ProSim
Step7ProSim.Interfaces
waitBeforeRecordingTimeInMilliSeconds
waitBeforePlayingRecordsTimeInMilliSeconds
payloadExecutionTimeInMilliSeconds
waitBeforePlayingRecordsTimer
waitBeforeRecordingTimer
payloadExecutionTimer
Step7ProSimProxy
$863d8af0-cee6-4676-96ad-13e8540f4d47
c:\Users\Main\Desktop\Step7ProSimProxy\Step7ProSimProxy\obj\Release\Ste
p7ProSim.pdb
```

Figure 2: Step7ProSim.dll Unique Strings

File Hashes

Table 3 contains the MD5 hashes, file and architecture type, and compile times for the malware analyzed in this report.

			09:26:27Z
update.exe$_1$	PE.EXE	X86	3,377,275
75D118996F5190EDAFCA1B1904A7EEA8			2012-05-25 09:26:27Z
update_no_pipe.exe$_1$	PE.EXE	X86	3,319,829
9F37E1EA08E6A4AE03E9FEBA6D1F6259			2012-05-25 09:26:27Z
update_no_pipe.exe$_2$	PE.EXE	X86	3,319,825
3152F21D701A2397E7B22711B8019B82			2012-05-25 09:26:27Z
update.exe$_2$	PE.EXE	X86	4,283,142
EF2A97512FDB45CD26089AD2FF61F1CC			2013-03-23 22:26:54Z
update.exe$_3$	PE.EXE	X86	2,768,872
41906403206EA5C7DCDBFAE230ADD9FA			2012-05-25 09:26:27Z
audiodg.exe	PE.EXE	X86	44,544
7A0C1017E6B5BB5DC776B3B883A1D0E0			2013-10-31 20:17:26Z
scada.exe	PE.EXE	X86	18,432
874F7BCAB71F4745EA6CDA2E2FB5A78C			2014-09-28 23:57:13Z
install.exe	PE.EXE	X86	150,328
1F338BDD92F08803A2AC7022A34D98FD			2006-08-16 17:28:36Z
Step7ProSim.dll	PE.DLL	X86	9,728
7C51474E6560C51DFC815D4A227BA1AA			2014-09-28 23:52:13Z

Table 3: File MD5 Hashes and Compile Times

FireEye detects IRONGATE. A list of indicators can be found here.

Special thanks to the Siemens ProductCERT for providing support and context to this investigation.

This entry was posted on Thu Jun 02 08:00:00 EDT 2016 and filed under Advanced Malware, Blog, ICS Security, Ics, Josh Homan, Latest Blog Posts, Rob Caldwell, Scada, Scada System Security, Sean McBride and Threat Research.