Symantec™
Security Response

Stuxnet 0.5: The Missing Link

Geoff McDonald,
Liam O Murchu,
Stephen Doherty,
Eric Chien

Contents

Overview

In 2010, Symantec reported on a new and highly sophisticated worm called Stuxnet. This worm became known as the first computer software threat that was used as a cyber-weapon. The worm was specifically designed to take control over industrial plant machinery and making them operate outside of their safe or normal performance envelope, causing damage in the process. This was a first in the history of malware.

Clues in the code pointed to other versions of the worm which could potentially perform different actions leaving an open question about Stuxnet and how it came to be. The wait for the missing link is now over. Symantec have now discovered an older version of Stuxnet that can answer the questions about the evolution of Stuxnet. This newly discovered variant has been dissected and analyzed in detail and here is a summary of our key findings:

- Stuxnet 0.5 is the oldest known Stuxnet version to be analyzed, in the wild as early as November 2007 and in development as early as November 2005.
- Stuxnet 0.5 was less aggressive than Stuxnet versions 1.x and only spread through infected Step 7 projects.
- Stuxnet 0.5 contains an alternative attack strategy, closing valves within the uranium enrichment facility at Natanz, Iran, which would have caused serious damage to the centrifuges and uranium enrichment system as a whole.

Whether Stuxnet 0.5 was successful is unclear, but later versions of Stuxnet were developed using a different development framework, became more aggressive, and employed a different attack strategy that changed the speeds of the centrifuges instead instead suggesting Stuxnet 0.5 did not completely fulfill the attacker's goals.

More versions of Stuxnet are known to exist, but have never been recovered.

Evolution

Stuxnet 0.5 was submitted to a malware scanning service in November 2007 and could have began operation as early as November 2005. This version is designed to stop compromising computers on July 4, 2009, and stop communicating with its command-and-control (C&C) servers on an earlier date of January 11 that same year. The compile timestamps found within most of the code appear unreliable and generally are in the range of the year 2001.

Table 1
Evolution of Stuxnet versions

Version	Date	Description
0.500	November 3, 2005	C&C server registration
0.500	November 15, 2007	Submit date to a public scanning service
0.500	July 4, 2009	Infection stop date
1.001	June 22, 2009	Main binary compile timestamp
1.100	March 1, 2010	Main binary compile timestamp
1.101	April 14, 2010	Main binary compile timestamp
1.x	June 24, 2012	Infection stop date

Table 2
Evolution of Stuxnet exploits

Vulnerability	0.500	1.001	1.100	1.101	Description
CVE-2010-3888			X	X	Task scheduler EOP
CVE-2010-2743			X	X	LoadKeyboardLayout EOP
CVE-2010-2729		X	X	X	Print spooler RCE
CVE-2008-4250		X	X	X	Windows Server Service RPC RCE
CVE-2012-3015	X	X	X	X	Step 7 Insecure Library Loading
CVE-2010-2772		X	X	X	WinCC default password
CVE-2010-2568			X	X	Shortcut .lnk RCE
MS09-025		X			NtUserRegisterClassExWow/NtUserMessageCall EOP

Based on an internal version number this version of Stuxnet is 0.5, the earliest known version of the Stuxnet family.

The only method of replication in Stuxnet 0.5 is through infection of Siemens Step 7 project files. Stuxnet 0.5 does not exploit any Microsoft vulnerabilities, unlike versions 1.x which came later.

There are differences in exploited vulnerabilities and spreading mechanisms between Stuxnet versions.

Table 3
Evolution of Stuxnet replication

Replication Technique	0.500	1.001	1.100	1.101
Step 7 project files	X	X	X	X
USB through Step 7 project files	X			
USB through Autorun		X		
USB through CVE-2010-2568			X	X
Network shares		X	X	X
Windows Server RPC		X	X	X
Printer spooler		X	X	X
WinCC servers		X	X	X
Peer-to-peer updating through mailslots	X			
Peer-to-peer updating through RPC		X	X	X

Stuxnet 0.5 is partly based on the Flamer platform whereas 1.x versions were based primarily on the Tilded platform. Over time, the developers appear to have migrated more towards the Tilded platform. The developers actually re-implemented Flamer-platform components using the Tilded platform in later versions.

Both the Flamer and Tilded platform code bases are different enough to suggest different developers were involved.

Stuxnet 0.5 also contains code to attack the valve systems in a uranium enrichment facility rather than modifying centrifuge speeds, as in versions 1.x of Stuxnet.

Installation and load point

Stuxnet 0.5 arrives as an infected Step 7 project archive containing both the s7hkimdb.dll and XR000001.MDX files. Using the Multiple Siemens SIMATIC Products DLL Loading Arbitrary Code Execution Vulnerability (CVE-2012-3015), the S7hkimdb.dll file is executed, which then decrypts and injects the main XR00001.MDX Stuxnet binary file into the services.exe process. Stuxnet is now executing on the system.

Once injected into the services.exe process, a copy of the main Stuxnet binary and a companion DLL that implements the payload are saved to disk in encrypted form along with a MRXCLS.SYS load point driver. The main Stuxnet binary refers to itself as outbreak.dll and is saved to disk as oem7a.pnf. The companion DLL that implements the payload refers to itself as installation.dll and saved to disk as oem7w.pnf. When the system is booted, the MRXCLS.SYS load point driver will decrypt configuration data stored in the registry, decrypt the main Stuxnet binary, and inject it into the Explorer and Step 7 processes. The payload DLL will be decrypted as well and injected into the Explorer process. When loading dynamic-link library (DLL) resources, Stuxnet makes use of a module that mimics `LoadLibrary` rather than calling `LoadLibrary` itself. This technique is likely used to avoid security software and was not seen in versions 1.x of Stuxnet.

A second driver, PCIBUS.SYS, is also created which causes a forced reboot by generating a BSoD (Blue Screen of Death) 20 days after installation.

A third driver, USBACC11.SYS, is then installed. This driver is similar to the MRXCLS.SYS driver, but instead decrypts and injects DLLs for peer-to-peer and C&C communication into the svchost.exe and Internet Explorer processes.

The structure and organization as well as resource and export listings of each component is available in Appendix D.

Stuxnet 0.5 also checks the current date in a variety of code paths and will not continue to spread after July 4, 2009. Certain modules may also not be created or loaded if security software is present. A list is available in Appendix B.

A variety of additional files are created, including log files and configuration files. A list is available in Appendix A.

Replication

Stuxnet 0.5 uses one form of replication through Step 7 project archives. When a removable drive is inserted in an infected system, Stuxnet 0.5 will infect any Step 7 project archives with a .s7p or .zip file name extension on the drive. In addition, Step 7 project archives on the local disk will also be infected.

Therefore Stuxnet 0.5 spreads to additional machines through removable drives or through human-initiated sharing of infected Step 7 project archives, for example through email.

Stuxnet 0.5 infects Step 7 project archives in the same manner as Stuxnet 1.x versions (as described in W32.

Stuxnet Dossier, *Step 7 Project File Infections*). The following is an example file listing of an infected Step 7 project file.

```
ApiLog/Types – modified to trigger DLL loading vulnerability
XUTILS/links/S7P00001.DBF – configuration file
XUTILS/listen/S7000001.MDX – payload DLL (installation.dll)
XUTILS/listen/XR000000.MDX – main Stuxnet binary (outbreak.dll)
hOmSave7/subfolder/s7hkimdb.dll - loader
```

Command-and-control

Similar to Stuxnet 1.x versions, Stuxnet 0.5 has limited command-and-control ability. In particular, Stuxnet 0.5 does not provide fine grained control to its authors. Instead, Stuxnet 0.5 can only download new code and update itself. Stuxnet needs to ultimately spread on isolated networks with no Internet access, therefore it has been designed to be autonomous to reduce the need for robust and fine grained command-and-control. Stuxnet 0.5 also uses a secondary peer-to-peer mechanism to propagate these code updates to peers on networks inaccessible to the broader Internet.

Command-and-control is implemented by the inetpsp.dll file while peer-to-peer communications are implemented by the netsimp32.dll file. Both files are loaded by the usbacc11.sys driver and then injected into the svchost.exe and iexplore.exe processes.

Stuxnet 0.5 has four C&C servers, all of which are now either unavailable or have since been registered by an unrelated party:

- smartclick.org
- best-advertising.net
- internetadvertising4u.com
- ad-marketing.net

Interestingly, Stuxnet 0.5 is programmed to stop contacting the C&C server after January 11, 2009, even though the threat is programmed to only stop spreading after a later date of July 4, 2009.

The C&C server domains were created in 2005 and all displayed the same front page purporting to be an Internet advertising agency named Media Suffix (figure 1) with the tag line "Believe What the Mind Can Dream".

The servers were hosted on commercial hosting providers in the United States, Canada, France, and Thailand. The fact that these domains were in operation since 2005, suggests that the Stuxnet project started more than seven years ago.

Figure 1

Internet advertising agency homepage for Stuxnet C&C servers

The first request by Stuxnet 0.5 uses the following form:

```
http://<domain>/cgi/link.php?site=xx
```

This notifies the C&C server of an active successful infection. Next, Stuxnet 0.5 sends the following request:

```
http://<domain>/cgi/click.php?xite=xx&num=yy&c=1&j=%x&k=%x&l=%x
```

This may download and execute a file if an update is available.

The final target network for Stuxnet 0.5 was likely isolated from the Internet. To allow updates to reach these machines, Stuxnet 0.5 also used a peer-to-peer mechanism. As long as one updated version was introduced into this network—for example through an infected USB key—all the other infected machines on the network could receive updates or new code modules.

Stuxnet 0.5 uses Windows mailslots for peer-to-peer communication. Mailslots allow a process to pass a message to another process on a remote machine. Stuxnet 0.5 enumerates all machines on the network and attempts to connect to a mailslot named `\\[REMOTE MACHINE NAME]\mailslot\svchost`. It then provides a callback mailslot name of `\\[LOCAL MACHINE NAME]\mailslot\imnotify`.

Stuxnet 0.5 uses these mailslots for peer-to-peer communication and to pass code updates. In addition, Stuxnet 0.5 may configure systems to allow anonymous logins and then provides the following file shares:

- temp$
- msagent$
- SYSADMIN$
- WebFiles$

This allows file retrieval by peer infections. Shared files include:

```
%WinDir%\msagent\agentsb.dll
%WinDir%\msagent\intl\agt0f2e.dll
%WinDir%\system32\complnd.dll
%WinDir%\system32\dllcache\datacprs.dll
%WinDir%\system32\wbem\perfnws.dll
%WinDir%\Installer\{6F716D8C-398F-11D3-85E1-005004838609}\places.dat
```

Payload

Man-in-the-Middle

In order to both fingerprint the target system and inject malicious Programmable Logic Controller (PLC) code, Stuxnet 0.5 replaces two Step 7 DLLs in order to hijack communications with a PLC.

The first DLL, s7otbxdx.dll, is hijacked in order to insert the malicious PLC code. The same technique was used in Stuxnet versions 1.x (as described in W32.Stuxnet Dossier, *Modifying PLCs*). Stuxnet 0.5 hooks fewer exports and verifies the CPU is a 417 PLC rather than a 315 PLC, otherwise the behavior remains generally the same.

The second DLL, s7aaapix.dll, is used for fingerprinting the target system and building DB8061, a PLC data block needed to conduct the attack. The export `AUTDoVerb` is hijacked and the malicious s7otbxdx.dll file can call the export with magic values (0x91E55A3D, 0x996AB716, 0x4A5CBB03) in order to build or provide a previously built DB8061 data block for injection. Stuxnet hijacks `AUTDoVerb` in order to monitor any “DOWNLOAD” verb actions, which signifies the fingerprinting and building of DB8061 must occur again in order to ensure the target system is still correctly configured.

Fingerprinting and building DB8061

The building of the DB8061 block is a complicated and lengthy process.

Through the hijacked export, Stuxnet 0.5 will receive a pointer to the most recently used block (PLC programs consist of code and data blocks). Stuxnet 0.5 will then traverse the project structure in order to find the symbols used by the S7 Program in the active project. Symbols are human designated labels representing each device controlled by the PLC. The symbol labels loosely follow the ANSI/ISA S5.1 Instrumentation Symbols and Identification standard used in Piping and Instrumentation Diagrams (P&ID).

Stuxnet 0.5 uses these labels for both fingerprinting and determining the addresses of each device in order to modify the behavior of those devices.

Table 4
Stuxnet 0.5 hooks fewer exports

Stuxnet v0.500	Stuxnet v1.xxx
s7_event	s7_event
	s7ag_bub_cycl_read_create
	s7ag_bub_read_var
	s7ag_bub_read_var_seg
	s7ag_bub_write_var
	s7ag_bub_write_var_seg
	s7ag_link_in
s7ag_read_szl	s7ag_read_szl
s7ag_test	s7ag_test
s7blk_delete	s7blk_delete
s7blk_findfirst	s7blk_findfirst
s7blk_findnext	s7blk_findnext
s7blk_read	s7blk_read
s7blk_write	s7blk_write
	s7db_close
s7db_open	s7db_open

Symbol label parsing

The target system must be a SIMATIC 400 Station (0x14109A) or SIMATIC H-Station (0x141342), which use 417 PLCs. The symbol labels must match the format:

```
<delimiter><FunctionIdentifier><delimiter><CascadeModule><delimiter><CascadeNumber><DeviceNumber>
```

A valve in module A21, also in cascade 8, and associated with centrifuge 160, would have the symbol label PV-A21-8-160, for example.

Each field is defined as follows:

Delimiter

Either space (“ “), hyphen (“-”), underscore (“_”), or not present at all.

FunctionIdentifer

A string that matches a set of strings (available in Appendix C) that loosely follows the ANSI/ISA S5.1 Instrumentation Symbols and Identification standard. If the string is “PIA” (Pressure Indicator Alarm), it is expected to be followed by a one digit number. These strings will represent the device type (a valve, a transducer, or a status light, for instance).

CascadeModule

Must be the string “A21” to “A28” inclusive. These strings match cascade modules in Natanz, Iran, seen publicly described as “A24”, “A26”, and “A28”.

CascadeNumber

Single character that is in the letter range A to R. If it is not in this letter range, it checks to see if it is two digits in the decimal range 00 to 18. This two digit number is the number representation of the letter for A to R.

DeviceNumber

This is parsed in a more complex fashion depending on the device type as determined by the function identifier and caters to three possible cascade arrangements. The device type mappings to function identifiers are available in Appendix C.

Device type 0

A string of digits: If the length of the digits is less than three, the device type is changed to device type 6. If the length of the digits is greater than or equal to three, the device type is changed to device type 7.

Device type 1, 2, 3

"##": A two digit number in the range 1 to 25.

Device type 4, 5, or 7

Device type 4, 5, or 7 has three different formats:

Format 1

"####": Decoded as two separate two-digit numbers representing the stage number and the centrifuge number within the stage, respectively. The stage number must be in the decimal range 1 to 15, which matches with the known Natanz configuration. For each of these 15 stages, the maximum number of expected centrifuges in each corresponding stage is looked up in the following array.

Table 5
Stage numbers with expected numbers of centrifuges

Stage	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Max #	2	2	4	6	8	10	12	16	20	24	20	16	12	8	4

This means, for example, stage 3 is expected to have a second two digit number equal to 4 or less. This requirement is consistent with the centrifuge arrangement within a cascade.

Format 2

"###": A three digit number that must be less than 164, which is the number of centrifuges in a cascade.

Format 3

"##L": A two digit number followed by a letter. The letter must be in the range A to D, and the number must be in the decimal range 1 to 43. This arrangement sub-divides each stage into sub-clusters of four.

Device type 6

"##": A two digit number in the range 1 to 30.

Device type 8, 9, B, or C

"##": A two digit number in the range 1 to 3.

Device type A

"##<delimiter><string>": A two digit number in the range 1 to 3 with an optional string preceded by a delimiter character. The string must start with the letter S and contain the letter P. If the string is present, the device is modified to be device type 0xB instead (Flow Rate Transmitter Controller Output, rather than device type 0xA which is Flow Rate Transmitter Controller Input).

Based on the symbol fingerprinting, the following table summarizes what devices and labels Stuxnet looks for within the symbol table.

Table 6
Summary of Stuxnet symbol label parsing

Device type	P&ID function identifier	Devices per cascade	
		Min	Max
Auxiliary valve	{HS, HV, PV, EP}, {ZLO,ZO},{ZLC,ZC}	2	25
Centrifuge valve	{MVS, RVS, VS}, {MV,RV,SV,YV}	163	164
Stage pressure transducer	PT, PCV, PIA#, PIT, PIC, PI, PS	3	30
Centrifuge pressure transducer	PT, PCV, PIA#, PIT, PIC, PI, PS	0	164
Flow rate sensor	{FIA}, {FIT}, {FITC},FIC,FT,MFC,MFM}	0	3

Symbol address parsing

Each symbol label will have two corresponding addresses: -- the address in the process image area and the direct peripheral address of the device represented by the symbol. Modifying the memory at these addresses allows the PLC to control and read the behavior of the associated device. For example, the value may be a Boolean value turning a switch on or off, or a 16-bit value representing the current temperature of the system. Addresses can be either outputs (the PLC sets the value to modify the behavior of the device) or inputs (the PLC reads the value to determine the current state of the device).

Device types 0, 1, 5, and B must be output addresses and device types 2, 3, 4, 5, 6, 7, 8, 9, A, and C must be input addresses.

Values at addresses for device types 0, 1, 2, 3, 4, and 5 must be bit values. Values at addresses for device types 6, 7, 8, 9, A, B, and C must be 16-bit values.

Cascade rating and building DB8061

After parsing the symbols and addresses for each cascade, the code inspects the configuration of each cascade. Depending on the configuration, a rating is calculated. Certain devices in certain configurations will result in higher ratings. When complete, only the six highest-rated cascades have data written to DB8061.

Finally, a flag is set signifying DB8061 has been built. This flag is reset to 0 every time the "DOWNLOAD" verb is executed.

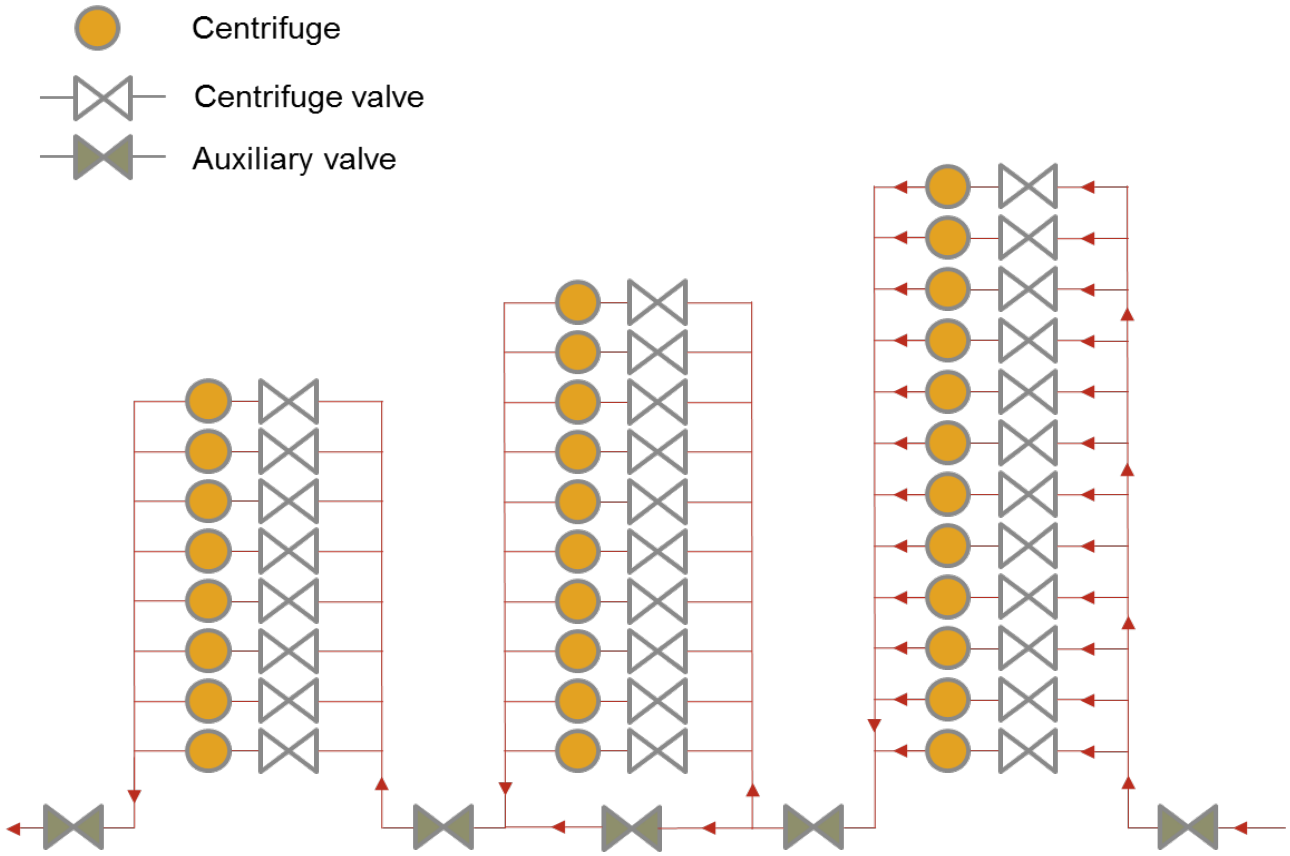
PLC device attack code

The code conducts an attack by closing valves in the six top rated cascades out of the possible 18 cascades. The states of two types of valves are modified:

- Centrifuge valves – a set of three valves (feed, product, tails) that work in unison per centrifuge to control uranium hexafluoride (UF6) flow into each centrifugeStage valves – one per stage to control UF6 flow into each stage
- Auxiliary valves – valves that control UF6 flow into or out of each stage (stage valve) or the cascade as a whole

Figure 2

Example valve configuration showing both valve types in three stages of a cascade



Similar to version 1.x of Stuxnet, the PLC device attack code consists of a state machine with eight possible states:

State 0 (Wait): Perform system identification and wait for the enrichment process to reach steady state before attack. This can take approximately 30 days.

State 1 (Record): Take peripheral snapshots and build fake input blocks for replaying later.

State 2 (Attack centrifuge valves): Begin replaying fake input signals. Close valves on most centrifuges.

State 3 (Secondary pressure reading): Open both centrifuge and feed stage valves in the final stage of a single cascade to obtain a low pressure reading.

State 4 (Wait for pressure change): Wait for desired pressure change or time limit. This can take up to approximately two hours.

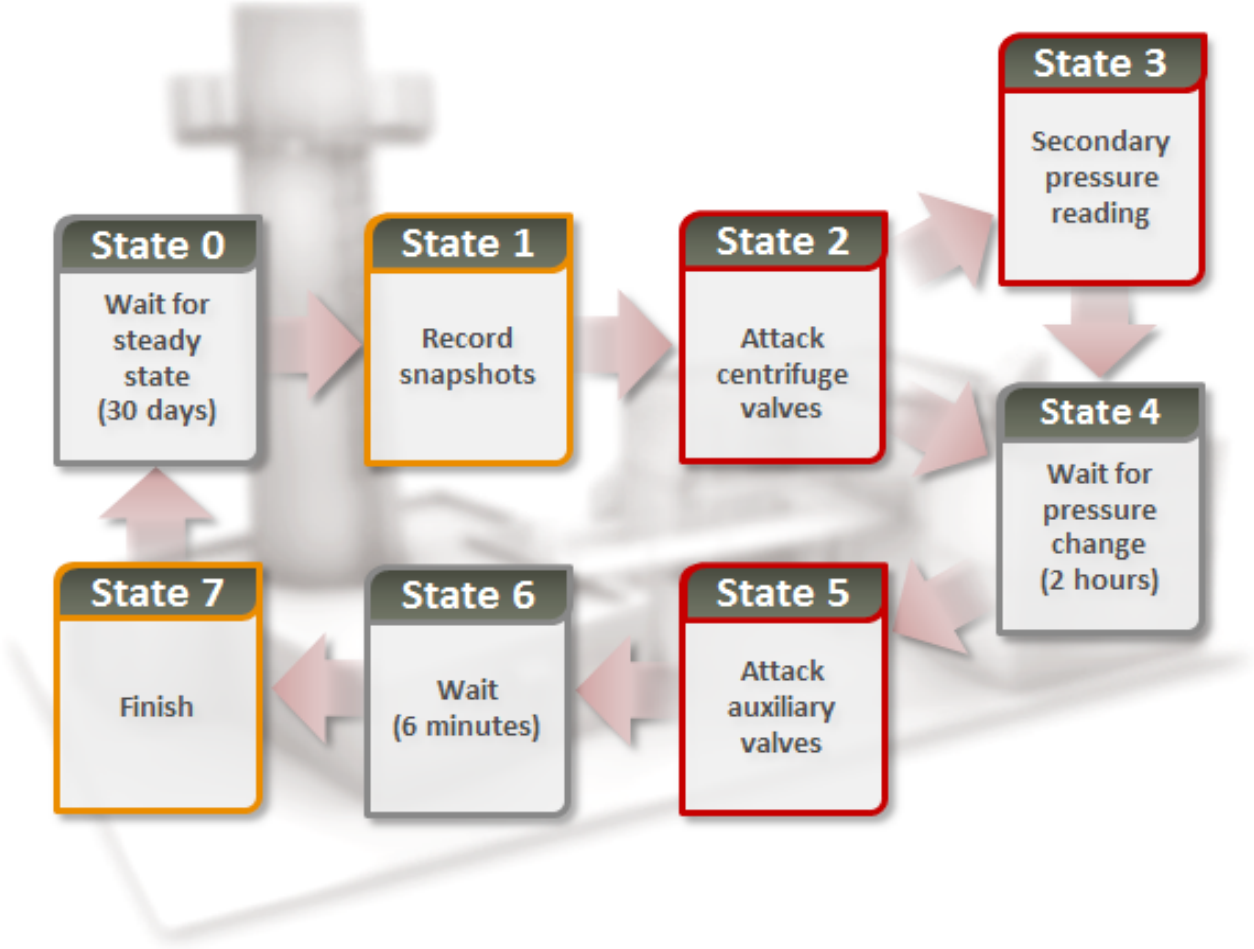
State 5 (Attack auxiliary valves): Open all auxiliary valves except valves believed to be near the first feed stage (stage 10). Wait for three minutes in this state.

State 6 (Wait for attack completion): Wait for six minutes while preventing any state changes.

State 7 (Finish): Reset and return to state zero.

Figure 3

State flow diagram of 417 PLC device attack code



State 0:

The code verifies the system has reached steady state by monitoring the state of each auxiliary valve and the amount of elapsed time.

- The valves must not change state for a period of 300 snapshots. In addition, the code determines if most of the centrifuge valves are in the open or closed position.
- All cascades must be operational for three or more days, or currently be in the down state.
- At least one cascade must have been operating for more than 35 days, or collectively all cascades must have been operating for more than 297 days.
- Between 3 and 7 of the first 21 auxiliary valves must have been opened in the last 2 days.

- Most of the pressure readings associated with the auxiliary valves must be within an expected range.

Only if these all conditions are met does the code proceed to state 1.

State 1:

There are 21 snapshots of the peripheral I/O values that are taken one second apart. These values are stored for replay during the attack. This prevents systems and technicians from realizing the system is no longer operating as expected.

State 2:

First, the normal operating pressure is obtained and stored for replay later. For each stage a portion of all the centrifuge valves are closed, except in the feed stage (stage 10). The centrifuge valves in the feed stage remain completely open, while the centrifuge valves at both the product end and tails end are completely closed.

The particular centrifuge valves closed per stage are randomly chosen. The code will randomly chose a starting centrifuge valve and then close the next one in order until the last centrifuge valve in the stage. If the total desired number of valves to close for that stage has not been reached, the code will continue from the first centrifuge valve in the stage until the maximum valves to close are reached.

Table 7
Processing stages and valves states

Stage	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Centrifuges	2	2	4	6	8	10	12	16	20	24	20	16	12	8	4
Centrifuge valves to close	2	2	2	4	6	8	10	13	14	0	14	13	10	8	4
Percentage closed	100%	100%	50%	67%	75%	80%	83%	81%	70%	0%	70%	81%	83%	100%	100%

State 3:

In state 3, in a single cascade, both centrifuge valves in stage 1 are opened and it is likely the stage valve of stage 1 is also opened. Then, the code obtains a pressure reading at stage 1. The pressure should be relatively low. This value is used for replay in later stages. If the normal pressure operating pressure wasn't obtained properly in state 2, state 3 is actually skipped and hardcoded default values are used instead.

State 4:

State 4 waits for the desired pressure change or other predetermined time limits before proceeding to state 5. If any of the following conditions are met, the code will continue to state 5:

- The pressure of the stage 10 or stage 11 transducer (these are likely transducers for or near the feed stage) has an absolute value greater than 280 units above the expected value and greater than five times the expected value.
- 46 minutes after the state of an auxiliary valve has been modified from the original state recorded in state 1, with the exception of auxiliary valve number 17 which is likely a stage valve near the product end.
- 2 hours and 3 minutes after the attack started (since state 2) without any centrifuge valve state changes.
- 2 hours and 3 minutes since at least four centrifuge valve states have been modified from the original state recorded in state 1.

State 5:

In state 5, all the auxiliary valves are opened except valve numbers 17, 18, and 20. Before continuing to state 6, the

code waits for at least 2 minutes and 53 seconds.

State 6:

During state 6, fake values continue to be replayed and any attempts to change device values are prevented for 6 minutes and 58 seconds.

State 7:

Data is reset and the code returns to state 0.

By closing all valves except the initial feed stage valves, UF6 will continue to flow into the system. This act alone may cause damage to the centrifuges themselves. However, the attack expects the pressure to reach five times the normal operating pressure. At this pressure, significant damage to the uranium enrichment system could occur and the UF6 gas could even revert to a solid.

Whether the attack succeeded in this manner or not remains unclear. Even if the attack did succeed, the attackers decided to move to a different strategy in the Stuxnet 1.x versions, attacking the speed of the centrifuges instead.

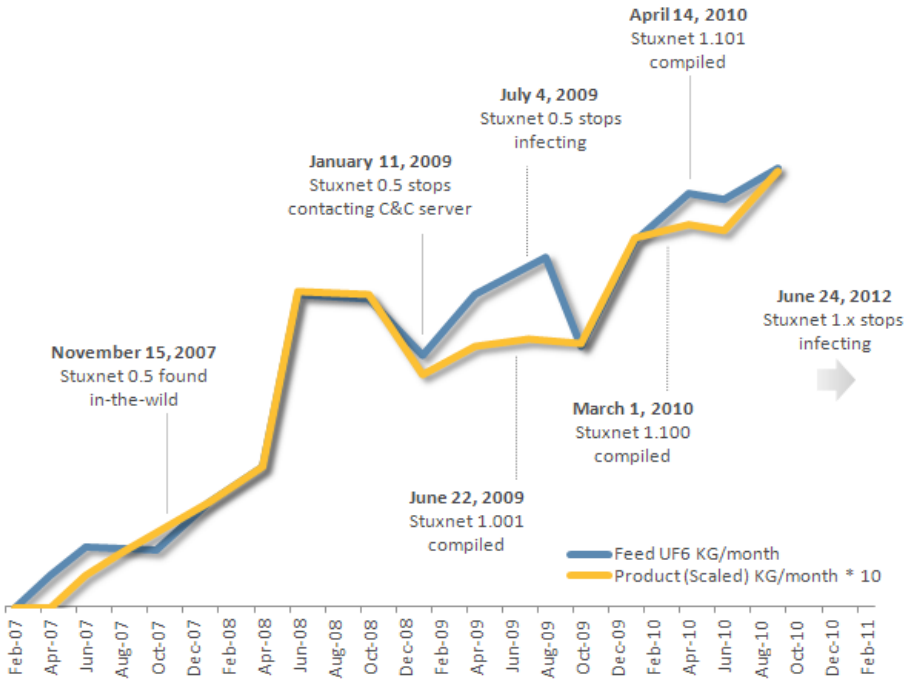
Conclusion

Stuxnet 0.5 clarifies the evolution and history of Stuxnet. Stuxnet clearly became more aggressive over time and switched development platforms as it evolved from 0.5 versions to later 1.x versions.

Key parts of the 417 attack code missing from versions 1.x is fully implemented in Stuxnet 0.5. This demonstrates that the 417 attack code was the first attack strategy implemented by Stuxnet. This original 417 attack code attempted to modify valve states during the uranium enrichment process at Natanz, Iran, to cause damage to the centrifuges and the system as a whole.

Figure 4

Low-enriched uranium production (source ISIS)



The success of Stuxnet 0.5 remains unknown. However, the chart in figure 4 references uranium enrichment production at Natanz to key milestones of Stuxnet development. Interesting events are dips in feed or production amounts and lower levels of production given the same or greater feed amounts (shown as gaps between the two lines).

While the discovery of Stuxnet 0.5 helps to deepen our overall understanding of Stuxnet and what its goals are, versions remain unrecovered. If these are located, they may expose other secrets behind this operation and more clues to its origins, but obtaining these other samples may prove to be next to impossible.

Appendix A

The following registry entries are indicators of compromise:

- `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MRxCls`
- `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\usbacc11`
- `HKEY_USERS\<SID>\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellRecoveryState`

The following files are indicators of compromise:

- %WinDir%\inf\mdmcpq3.PNF – Encrypted installation.dll
- %WinDir%\inf\mdmeric3.PNF – P2P configuration file
- %WinDir%\inf\oem6C.PNF – Log file
- %WinDir%\inf\oem7A.PNF – Main Stuxnet component (outbreak.dll)
- %WinDir%\inf\oem7F.pnf
- %WinDir%\inf\oem7w.pnf – Encrypted installation.dll
- %WinDir%\inf\~67.tmp – Encrypted installation.dll
- %System%\drivers\mrxcls.sys – Load point driver
- %System%\drivers\usbacc11.sys – Load point driver for C&C server modules
- %System%\drivers\PCIBUS.SYS – Timer driver for generating BSoD
- %System%\comuid.dat
- %System%\netsimp32.dll – P2P communication
- %System%\inetpsp.dll – C&C server communication
- %System%\perfg009.dat
- %WinDir%\msagent\agentsb.dll
- %WinDir%\msagent\intl\agt0f2e.dll
- %System%\compInd.dll
- %System%\dllcache\datacprs.dll
- %System%\wbem\perfnws.dll
- %WinDir%\Installer\{6F716D8C-398F-11D3-85E1-005004838609}\places.dat
- %System%\dssbase.dat – Log file
- %AllUsersProfile%\Application Data\Microsoft\HTML Help\hhorcslt.dat
- %Temp%/DF419a.tmp
- %WinDir%\help\winmic.fts – Configuration file for Step 7 infections

Appendix B

Processes checked for and the assumed security product associated with the process:

- umxagent, Tiny Personal Firewall
- cfgintpr, Tiny Personal Firewall
- umxldra, Tiny Personal Firewall
- amon, Tiny Activity Monitor
- UmxCfg, Tiny Personal Firewall
- UmxPol, Tiny Personal Firewall
- UmxTray, Tiny Personal Firewall
- vsmon, ZoneAlarm Personal Firewall
- zapro, ZoneAlarm Personal Firewall
- zlclient, ZoneAlarm Personal Firewall
- tds-3,TDS3 Trojan Defense Suite
- avp, Kaspersky
- avpcc, Kaspersky
- avpm, Kaspersky
- kavpf, Kaspersky
- kavi, Kaspersky
- safensec,SafenSoft
- snsmcon, SafenSoft
- filemon, Sysinternals Filemon
- regmon, Sysinternals Filemon
- FrameworkService, McAfee
- UpdaterUI, McAfee
- shstat, McAfee
- naPrdMgr, McAfee
- rapapp.exe, Blackice Firewall
- blackice.exe, Blackice Firewall
- blackd.exe, Blackice Firewall
- rcfgsvc.exe
- pfwcfgsurrogate.exe, Tiny Personal Firewall
- pfwadmin.exe, Tiny Personal Firewall
- persfw.exe, Kerio Personal Firewall
- agentw.exe, Kerio Personal Firewall
- agenta.exe, Kerio Personal Firewall
- msascui.exe, Windows Defender
- msmpeng.exe, Windows Defender
- fssm32.exe, F-Secure
- fsgk32st.exe, F-Secure
- fsdfwd.exe, F-Secure
- fsaw.exe, F-Secure
- fsavgui.exe, F-Secure
- fsav32.exe, F-Secure
- fsav.exe, F-Secure
- fsma32.exe, F-Secure
- fsm32.exe, F-Secure
- fsgk32.exe, F-Secure

Appendix C

Table of each allowed function identifier, the corresponding device type, and the assumed device name.

Table 8

Function identifiers, device types, and assumed device name

Function identifier	Device type	Device name
PT	0	Pressure Transmitter
PCV	0	Pressure Control Valve
PIA	0	Pressure Indicator Alarm
PIT	0	Pressure Indicator Transmitter
PIC	0	Pressure Indicator Controller
PI	0	Pressure Indicator
PS	0	Pressure Switch
HS	1	Hand Switch
HV	1	Hand Valve
PV	1	Pressure Valve
EP	1	Voltage (Test) Point
ZLO	2	Light Position Open (Status light)
ZO	2	Position Open
ZLC	3	Light Position Closed
ZC	3	Position Closed
MVS	4	Manual Valve Switch
RVS	4	Relief Valve Switch
VS	4	Valve Switch
SHS	4	High Frequency Switch
MV	5	Manual Valve
RV	5	Relief Valve
SV	5	Frequency Control Valve
YV	5	Valve State Indicator
FIA	8	Flow Rate Indicator Alarm
FITC	A	Flow Rate Indicator Transmitter Controller
FIT	9	Flow Rate Indicator Transmitter
FIC	C	Flow Rate Indicator Controller
FT	C	Flow Rate Transmitter
MFC	C	Mass Flow Controller
MFM	C	Mass Flow Meter

Appendix D

Organization of Stuxnet 0.5 components and behavior of each export.

Figure 5

Organization of Stuxnet 0.5 components

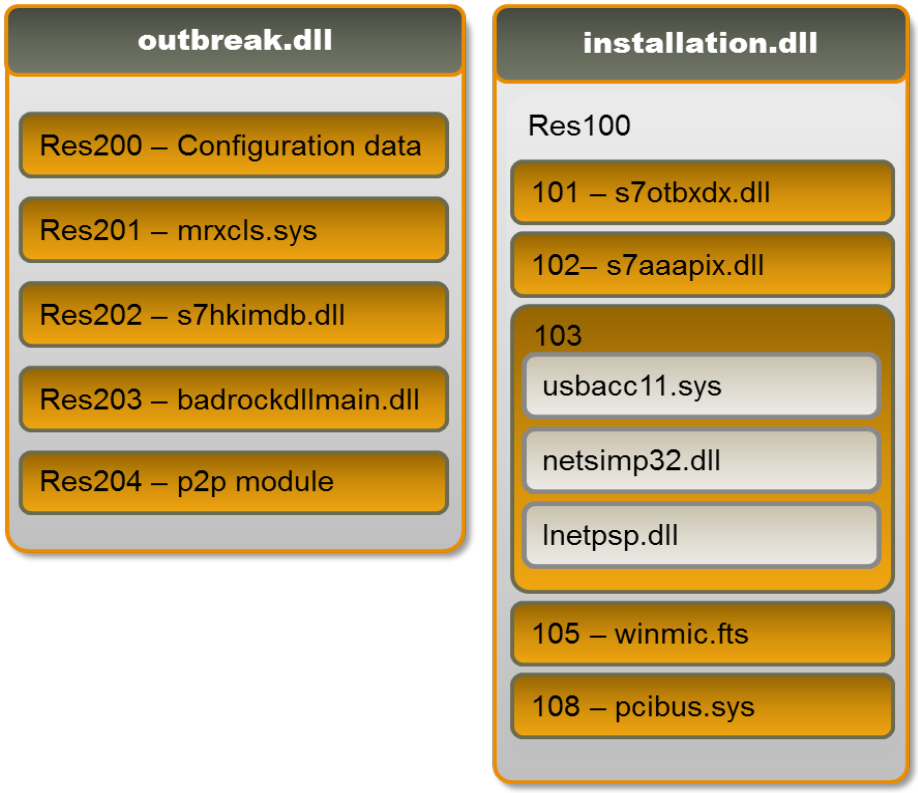


Table 9

Payload exports for Outbreak.dll

Payload exports	Description
Export 1	Infect Step 7 projects on insertion of removable drives
Export 2	Hook Step 7 for Step 7 project infection
Export 4	Uninstall routine
Export 5	Verifies installation
Export 6	Returns version number
Export 7	Loads peer-to-peer communication data file
Export 8, 9, and 10	Updates Stuxnet from infected Step 7 project archives
Export 11	Inject module into services.exe
Export 12	Install routine
Export 13	Call export 1

Resources

W32.Duqu

http://www.symantec.com/security_response/writeup.jsp?docid=2011-101814-1119-99

W32.Flamer

http://www.symantec.com/security_response/writeup.jsp?docid=2012-052811-0308-99

W32.Stuxnet

http://www.symantec.com/security_response/writeup.jsp?docid=2010-071400-3123-99

Multiple Siemens SIMATIC Products DLL Loading Arbitrary Code Execution Vulnerability (CVE-2012-3015)

http://www.securityfocus.com/bid/54651

Stuxnet 0.5: The Missing Link

http://www.symantec.com/connect/blogs/stuxnet-05-missing-link

Stuxnet 0.5: Disrupting Uranium Processing At Natanz

http://www.symantec.com/connect/blogs/stuxnet-05-disrupting-uranium-processing-natanz

Stuxnet 0.5: How it Evolved

http://www.symantec.com/connect/blogs/stuxnet-05-how-it-evolved

Stuxnet 0.5: Command-and-Control Capabilities

http://www.symantec.com/connect/blogs/stuxnet-05-command-and-control-capabilities

Community credits

Symantec would like to thank the Institute for Science and International Security (ISIS) for their continued assistance in understanding centrifugal uranium enrichment systems.

Symantec
Security Response

About Symantec

Symantec protects the world's information, and is a global leader in security, backup and availability solutions. Our innovative products and services protect people and information in any environment – from the smallest mobile device, to the enterprise data center, to cloud-based systems. Our world-renowned expertise in protecting data, identities and interactions gives our customers confidence in a connected world. More information is available at www.symantec.com or by connecting with Symantec at: go.symantec.com/socialmedia.

About the authors

Geoff McDonald - Threat Analysis Engineer
Liam O Murchu - Development Manager
Stephen Doherty - Sr Threat Intelligence Analyst
Eric Chien - Distinguished Engineer

For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 (800) 745 6054.

Symantec Corporation
World Headquarters
350 Ellis Street
Mountain View, CA 94043 USA
+1 (650) 527-8000
www.symantec.com