Critical Infrastructure Security: The Emerging Smart Grid

Cyber Security Lecture 6: Case Study

Carl Hauser & Adam Hahn
How to Design a Secure Grid

Power Grid
Cybersecurity

Secure Cyber Infrastructure

Resilient Power Control Applications

Communication
Authentication
Encryption
Computation
Access control
Attestation
Forensics
Patch management
Software Audits
System Management
Intrusion Detection
Event Monitoring/Analytics
Security Assessment

Generation
Automatic Generation Control
Governor Control
Automatic Voltage Regulation
Protection
Transmission
State Estimation
VAR Compensation
Protection
Distribution
Load Shedding
Protection
Advanced Metering Infrastructure

Source: S Sridhar, A Hahn, M Govindarasu, Cyber–physical system security for the electric power grid. Proceedings of the IEEE
Case Study (Stuxnet)
Stuxnet

- Computer worm discovered in June 2010
- Spreads via Microsoft Windows
- Targets Siemens SCADA systems
- Includes zero-day exploits, a Windows rootkit, the first ever PLC rootkit, antivirus evasion techniques, complex process injection and hooking code, network infection routines, peer-to-peer updates, and a command and control interface
- CVE-2010-2568: Windows Shell in Microsoft Windows XP SP3, Server 2003 SP2, Vista SP1 and SP2, Server 2008 SP2 and R2, and Windows 7 allows local users or remote attackers to execute arbitrary code via a crafted (1) .LNK or (2) .PIF shortcut file, which is not properly handled during icon display in Windows Explorer, as demonstrated in the wild in July 2010, and originally reported for malware that leverages CVE-2010-2772 in Siemens WinCC SCADA systems. (Jun 2010)

- MS10-046: The vulnerability could allow remote code execution if the icon of a specially crafted shortcut is displayed. An attacker who successfully exploited this vulnerability could gain the same user rights as the local user. (Aug 2010)
| Apr 2009 | Security magazine Hakin9 releases details of a remote code execution vulnerability in the Printer Spooler service. |

- **MS10-061:** The vulnerability could allow remote code execution if an attacker sends a specially crafted print request to a vulnerable system that has a print spooler interface exposed over RPC. (Sep 2010)
<table>
<thead>
<tr>
<th>Date</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 2009</td>
<td>Earliest Stuxnet sample seen. Does not exploit MS10-046. Does not have signed driver files.</td>
</tr>
<tr>
<td>Jan 25, 2010</td>
<td>Stuxnet driver signed with a valid certificate belonging to Realtek Semiconductor Corps.</td>
</tr>
<tr>
<td>Mar 2010</td>
<td>First Stuxnet variant to exploit MS10-046.</td>
</tr>
<tr>
<td>Jun 17, 2010</td>
<td>Virusblokada reports W32.Stuxnet (named RootkitTmphider). Reports that it’s using a vulnerability in the processing of shortcuts/.lnk files in order to propagate (MS10-046).</td>
</tr>
<tr>
<td>Date</td>
<td>Event</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Jul 17, 2010</td>
<td>Eset identifies a new Stuxnet driver, this time signed with a certificate from JMicron Technology Corp.</td>
</tr>
<tr>
<td>Jul 19, 2010</td>
<td>Siemens report that they are investigating reports of malware infecting Siemens WinCC SCADA systems.</td>
</tr>
<tr>
<td>Jul 22, 2010</td>
<td>Verisign revokes the JMicron Technology Corps certificate.</td>
</tr>
</tbody>
</table>
Siemens Simatic WinCC and PCS 7 SCADA system uses a hard-coded password, which allows local users to access a back-end database and gain privileges, as demonstrated in the wild in July 2010 by the Stuxnet worm, a different vulnerability than CVE-2010-2568.
Affected Countries

- Iran: 58.85%
- Indonesia: 18.22%
- India: 8.31%
- Ajerbaijan: 2.57%
- USA: 1.56%
- Pakistan: 1.28%
- Others: 9.2%

Infected Computers (%)

Countries: Iran, Indonesia, India, Ajerbaijan, USA, Pakistan, Others
What is So Special about Stuxnet?

- First to exploit multiple zero-day vulnerabilities
- First to use stolen signing keys and valid certificates of two companies
- First to spy on and reprogram industrial control systems
- First PLC rootkit
- First “working and fearsome prototype of a cyber-weapon that will lead to the creation of a new arms race in the world” —Kaspersky Labs
• Specialized assembly code on PLCs
• Controlled by a Windows machine not connected to the Internet
Target

Siemens S7 PLC

Siemens PCS 7
WinCC and STEP7

MS Windows
Initial Infection

Hidden autorun commands

```
[autorun]
objectDescriptor={B315537-63AB-9512-99A9-2F4677235A44}
.Menu\command=./AUTORUN.INF
.Menu=@%windir%\system32\shell32.dll,-8496
UseAutoPLAY=0
```

Two “Open” commands
The .lnk files are created using Resource 240 as a template and four are needed as each specifically targets one or more different versions of Windows including Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows 7. The .lnk files contain an exploit that will automatically execute ~WTR4141.tmp when simply viewing the folder.

~WTR4141.tmp then loads ~WTR4132.tmp, but before doing so, it attempts to hide the files on the removable drive. Hiding the files on the removable drive as early in the infection process as possible is important for the threat since the rootkit functionality is not installed yet, as described in the Windows Rootkit Functionality section. Thus, ~WTR4141.tmp implements its own less-robust technique in the meantime.

~WTR4141.tmp hooks the following APIs from kernel32.dll and Ntdll.dll:

From Kernel32.dll
- FindFirstFileW
- FindNextFileW
- FindFirstFileExW

From Ntdll.dll
- NtQueryDirectoryFile
- ZwQueryDirectoryFile

It replaces the original code for these functions with code that checks for files with the following properties:
- Files with an .lnk extension having a size of 4,171 bytes.
- Files named ~WTRxxxx.TMP, sized between 4Kb and 8 Mb, where xxxx is:
  - 4 decimal digits. (~wtr4132.tmp)
  - The sum of these digits modulo 10 is null. (Example: 4+1+3+2=10=0 mod 10)

If a request is made to list a file with the above properties, the response from these APIs is altered to state that the file does not exist, thereby hiding all files with these properties.

After the DLL APIs are hooked, ~WTR4132.tmp is loaded. To load a .dll file normally, a program calls the "LoadLibrary" API with the file name of the .dll file to be loaded into memory. W32.Stuxnet uses a different approach, not just in the first .dll file but in several different parts of the code. This method is described in the Bypassing Behavior Blocking When Loading DLLs section.

~WTR4132.tmp contains the main Stuxnet DLL in the .stub section. This is extracted into memory and then Export 15 of the DLL is called executing the installation of Stuxnet. Export 15 is described in the Installation section.

The diagram to the right describes the execution flow.
Network Propagation

- Infecting WinCC machines via a hardcoded database server password
- Propagating through network shares
- Propagating through the MS10-061 Print Spooler Zero-Day Vulnerability
- Peer-to-peer communication and updates
- Propagating through the MS08-067 Windows Server Service Vulnerability

Looks for a machine running STEP7
Through Network Shares

Access network resources using each user’s credential

DEFRAG[RANDLNT].tmp

Schedule to execute two minutes after infection
The vulnerability could **allow remote code execution** if an attacker sends a specially crafted print request to a vulnerable system that has a print spooler interface exposed over RPC.

Create and execute `%System%\winsta.exe`
The vulnerability could **allow remote code execution** if an affected system received a specially crafted RPC request.
CREATE TABLE sysbinlog ( abin image ) INSERT INTO sysbinlog VALUES(0x...)

Infected machine \( \rightarrow \) SQL \( \rightarrow \) Target machine (WinCC)

Create and execute
%UserProfile%\sql[RANDOM VALUE].dbi
Peer-to-Peer Communication

Stuxnet has the ability to propagate using a variety of methods. It propagates by infecting removable drives and also by copying itself over the network using a variety of means, including two exploits. In addition, Stuxnet propagates by copying itself to Step 7 projects using a technique that causes Stuxnet to auto-execute when opening the project. The following sections describe the network, removable drive, and Step 7 project propagation routines.

Network propagation routines

Export 22 is responsible for the majority of the network propagation routines that Stuxnet uses. This export builds a "Network Action" class that contains 5 subclasses. Each subclass is responsible for a different method of infecting a remote host.

The functions of the 5 subclasses are:

- Peer-to-peer communication and updates
- Infecting WinCC machines via a hardcoded database server password
- Propagating through network shares
- Propagating through the MS10-061 Print Spooler Zero-Day Vulnerability
- Propagating through the MS08-067 Windows Server Service Vulnerability

Each of these classes is discussed in more detail below.

Peer-to-peer communication

The P2P component works by installing an RPC server and client. When the threat infects a computer it starts the RPC server and listens for connections. Any other compromised computer on the network can connect to the RPC server and ask what version of the threat is installed on the remote computer.

If the remote version is newer then the local computer will make a request for the new version and will update itself with that. If the remote version is older the local computer will prepare a copy of itself and send it to the remote computer so that it can update itself. In this way an update can be introduced to any compromised computer on a network and it will eventually spread to all other compromised computers.

All of the P2P requests take place over RPC as outlined below.

The RPC server offers the following routines. (Note that RPC methods 7, 8, 9 are not used by Stuxnet.)

0: Returns the version number of Stuxnet
1: Receive an .exe file and execute it (through injection)
2: Load module and executed export
3: Inject code into lsass.exe and run it
4: Builds the latest version of Stuxnet and sends to compromised computer
5: Create process
6: Read file
7: Drop file
8: Delete file
9: Write data records

![Peer-to-Peer Communication Diagram](image-url)
Stuxnet Architecture

Organization

Stuxnet has a complex architecture that is worth outlining before continuing with our analysis.

The heart of Stuxnet consists of a large .dll file that contains many different exports and resources. In addition to the large .dll file, Stuxnet also contains two encrypted configuration blocks.

The dropper component of Stuxnet is a wrapper program that contains all of the above components stored inside itself in a section name "stub". This stub section is integral to the working of Stuxnet. When the threat is executed, the wrapper extracts the .dll file from the stub section, maps it into memory as a module, and calls one of the exports.

A pointer to the original stub section is passed to this export as a parameter. This export in turn will extract the .dll file from the stub section, which was passed as a parameter, map it into memory and call another different export from inside the mapped .dll file. The pointer to the original stub section is again passed as a parameter. This occurs continuously throughout the execution of the threat, so the original stub section is continuously passed around between different processes and functions as a parameter to the main payload. In this way every layer of the threat always has access to the main .dll and the configuration blocks.

In addition to loading the .dll file into memory and calling an export directly, Stuxnet also uses another technique to call exports from the main .dll file. This technique is to read an executable template from its own resources, populate the template with appropriate data, such as which .dll file to load and which export to call, and then to inject this newly populated executable into another process and execute it. The newly populated executable template will load the original .dll file and call whatever export the template was populated with.

Although the threat uses these two different techniques to call exports in the main .dll file, it should be clear that all the functionality of the threat can be ascertained by analyzing all of the exports from the main .dll file.

Exports

As mentioned above, the main .dll file contains all of the code to control the worm. Each export from this .dll file has a different purpose in controlling the threat as outlined in table 3.

<table>
<thead>
<tr>
<th>Export #</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Infect connected removable drives, starts RPC server</td>
</tr>
<tr>
<td>2</td>
<td>Hooks APIs for Step 7 project file infections</td>
</tr>
<tr>
<td>4</td>
<td>Calls the removal routine (export 18)</td>
</tr>
<tr>
<td>5</td>
<td>Verifies if the threat is installed correctly</td>
</tr>
<tr>
<td>6</td>
<td>Verifies version information</td>
</tr>
<tr>
<td>7</td>
<td>Calls Export 6</td>
</tr>
<tr>
<td>9</td>
<td>Updates itself from infected Step 7 projects</td>
</tr>
<tr>
<td>10</td>
<td>Updates itself from infected Step 7 projects</td>
</tr>
<tr>
<td>14</td>
<td>Step 7 project file infection routine</td>
</tr>
<tr>
<td>15</td>
<td>Initial entry point</td>
</tr>
<tr>
<td>16</td>
<td>Main installation</td>
</tr>
<tr>
<td>17</td>
<td>Replaces Step 7 DLL</td>
</tr>
<tr>
<td>18</td>
<td>Uninstalls Stuxnet</td>
</tr>
<tr>
<td>19</td>
<td>Infects removable drives</td>
</tr>
<tr>
<td>22</td>
<td>Network propagation routines</td>
</tr>
<tr>
<td>24</td>
<td>Check Internet connection</td>
</tr>
<tr>
<td>27</td>
<td>RPC Server</td>
</tr>
<tr>
<td>28</td>
<td>Command and control routine</td>
</tr>
<tr>
<td>29</td>
<td>Command and control routine</td>
</tr>
<tr>
<td>31</td>
<td>Updates itself from infected Step 7 projects</td>
</tr>
<tr>
<td>32</td>
<td>Same as 1</td>
</tr>
</tbody>
</table>
The main .dll file also contains many different resources that the exports above use in the course of controlling the worm. The resources vary from full .dll files to template executables to configuration files and exploit modules.

Both the exports and resources are discussed in the sections below.

**Bypassing Behavior Blocking When Loading DLLs**

Whenever Stuxnet needs to load a DLL, including itself, it uses a special method designed to bypass behavior-blocking and host intrusion-protection based technologies that monitor LoadLibrary calls. Stuxnet calls LoadLibrary with a specially crafted file name that does not exist on disk and normally causes LoadLibrary to fail. However, W32.Stuxnet has hooked Ntdll.dll to monitor for requests to load specially crafted file names. These specially crafted filenames are mapped to another location instead—a location specified by W32.Stuxnet. That location is generally an area in memory where a .dll file has been decrypted and stored by the threat previously. The filenames used have the pattern of KERNEL32.DLL.ASLR.[HEXADECIMAL] or SHELL32.DLL.ASLR. [HEXADECIMAL], where the variable [HEXADECIMAL] is a hexadecimal value.

The functions hooked for this purpose in Ntdll.dll are:

- ZwMapViewOfSection
- ZwCreateSection
- ZwOpenFile
- ZwCloseFile
- ZwQueryAttributesFile
- ZwQuerySection

Once a .dll file has been loaded via the method shown above, GetProcAddress is used to find the address of a specific export from the .dll file and that export is called, handing control to that new .dll file.

**Table 4: DLL Resources**

<table>
<thead>
<tr>
<th>Resource ID</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>MrxNet.sys load driver, signed by Realtek</td>
</tr>
<tr>
<td>202</td>
<td>DLL for Step 7 infections</td>
</tr>
<tr>
<td>203</td>
<td>CAB file for WinCC infections</td>
</tr>
<tr>
<td>205</td>
<td>Data file for Resource 201</td>
</tr>
<tr>
<td>207</td>
<td>Autorun version of Stuxnet</td>
</tr>
<tr>
<td>208</td>
<td>Step 7 replacement DLL</td>
</tr>
<tr>
<td>209</td>
<td>Data file (%windows%\help\winmic.fts)</td>
</tr>
<tr>
<td>210</td>
<td>Template PE file used for injection</td>
</tr>
<tr>
<td>221</td>
<td>Exploits MS08-067 to spread via SMB.</td>
</tr>
<tr>
<td>222</td>
<td>Exploits MS10-061 Print Spooler Vulnerability</td>
</tr>
<tr>
<td>231</td>
<td>Internet connection check</td>
</tr>
<tr>
<td>240</td>
<td>LNK template file used to build LNK exploit</td>
</tr>
<tr>
<td>241</td>
<td>USB Loader DLL ~WTR4141.tmp</td>
</tr>
<tr>
<td>242</td>
<td>MRxnet.sys rootkit driver</td>
</tr>
<tr>
<td>250</td>
<td>Exploits Windows Win32k.sys Local Privilege Escalation (MS10-073)</td>
</tr>
</tbody>
</table>
Bypassing Host IDS

- IDS monitors LoadLibrary calls
- Stuxnet calls LoadLibrary with a specially crafted file name that does not exist
- LoadLibrary fails
- W32.Stuxnet has hooked Ntdll.dll to monitor for requests to load specially crafted file names, mapped to another location
- That location is generally an area in memory where a .dll file has been decrypted and stored by the threat previously
Injection Technique

- Inject into a trusted process
- Lsass.exe, Winlogon.exe, Svchost.exe
- Installed security product process

<table>
<thead>
<tr>
<th>Security Product Installed</th>
<th>Injection target</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAV v1 to v7</td>
<td>LSASS.EXE</td>
</tr>
<tr>
<td>KAV v8 to v9</td>
<td>KAV Process</td>
</tr>
<tr>
<td>McAfee</td>
<td>Winlogon.exe</td>
</tr>
<tr>
<td>AntiVir</td>
<td>Lsass.exe</td>
</tr>
<tr>
<td>BitDefender</td>
<td>Lsass.exe</td>
</tr>
<tr>
<td>ETrust v5 to v6</td>
<td>Fails to Inject</td>
</tr>
<tr>
<td>ETrust (Other)</td>
<td>Lsass.exe</td>
</tr>
<tr>
<td>F-Secure</td>
<td>Lsass.exe</td>
</tr>
<tr>
<td>Symantec</td>
<td>Lsass.exe</td>
</tr>
<tr>
<td>ESET NOD32</td>
<td>Lsass.exe</td>
</tr>
<tr>
<td>Trend PC Cillin</td>
<td>Trend Process</td>
</tr>
</tbody>
</table>
Installation (Export 15)

Export 15 is the first export called when the .dll file is loaded for the first time. It is responsible for checking that the threat is running on a compatible version of Windows, checking whether the computer is already infected or not, elevating the privilege of the current process to system, checking what antivirus products are installed, and what the best process to inject into is. It then injects the .dll file into the chosen process using a unique injection technique described in the Injection Technique section and calls export 16.

The first task in export 15 is to check if the configuration data is up-to-date. The configuration data can be stored in two locations. Stuxnet checks which is most up-to-date and proceeds with that configuration data. Next, Stuxnet determines if it is running on a 64-bit machine or not; if the machine is 64-bit the threat exits. At this point it also checks to see what operating system it is running on. Stuxnet will only run on the following operating systems:

- Win2K
- WinXP
- Windows 2003
- Vista
- Windows Server 2008
- Windows 7
- Windows Server 2008 R2

If it is not running on one of these operating systems it will exit.

Next, Stuxnet checks if it has Administrator rights on the computer. Stuxnet wants to run with the highest privilege possible so that it will have permission to take whatever actions it likes on the computer. If it does not have Administrator rights, it will execute one of the two zero-day escalation of privilege attacks described below.

![Diagram of the control flow for export 15](image)

Figure 10

Control flow for export 15
Attaining Admin Privileges

• MS10-073: The most severe of these vulnerabilities could allow elevation of privilege if an attacker logs on to an affected system and runs a specially crafted application.

• The currently undisclosed Task Scheduler Escalation of Privilege vulnerability
If the process already has the rights it requires it proceeds to prepare to call export 16 in the main .dll file. It calls export 16 by using the injection techniques described in the Injection Technique section.

When the process does not have Adminstrator rights on the system it will try to attain these privileges by using one of two zero-day escalation of privilege attacks. The attack vector used is based on the operating system of the compromised computer. If the operating system is Windows Vista, Windows 7, or Windows Server 2008 R2 the currently undisclosed Task Scheduler Escalation of Privilege vulnerability is exploited. If the operating system is Windows XP or Windows 2000 the Windows Win32k.sys Local Privilege Escalation vulnerability (MS10-073) is exploited.

If exploited, both of these vulnerabilities result in the main .dll file running as a new process, either within the csrss.exe process in the case of the win32k.sys vulnerability or as a new task with Adminstrator rights in the case of the Task Scheduler vulnerability.

The code to exploit the win32k.sys vulnerability is stored in resource 250. Details of the Task Scheduler vulnerability currently are not released as patches are not yet available. The Win32k.sys vulnerability is described in the Windows Win32k.sys Local Privilege Escalation vulnerability (MS10-073) section.

After export 15 completes the required checks, export 16 is called.

Export 16 is the main installer for Stuxnet. It checks the date and the version number of the compromised computer; decrypts, creates and installs the rootkit files and registry keys; injects itself into the services.exe process to infect removable drives; injects itself into the Step7 process to infect all Step 7 projects; sets up the global mutexes that are used to communicate between different components; and connects to the RPC server.

Export 16 first checks that the configuration data is valid, after that it checks the value "NTVDM TRACE" in the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\MS-DOS Emulation

Figure 11

Infection Routine Flow
MrxCls.sys

- Resource 242
- Driver digitally signed with a certificate
- Allows Stuxnet to be executed at boot
- Injects and executes Stuxnet into specific processes

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MrxCls\“Data”
Digital Signature

- Two certificates stolen from Hsinchu Science and Industrial Park
  - Realtek
  - JMicron
Command and Control

- Test network connectivity
  - www.windowsupdate.com
  - www.msn.com

- Send basic information about the compromised machine
  - www.mypremierfutbol.com at Malaysia
  - www.todaysfutbol.com at Denmark
1 & 2: Check internet connectivity
3: Send system information to C&C
4a: C&C response to execute RPC routine
4b: C&C response to execute encrypted binary code
Hiding Itself

- MrxNet.sys signed with a certificate
- Intercepts IRP requests
  - Writes/reads to NTFS, FAT, CD-ROM
- Filters out
  - “.LNK”, 4,171 bytes long
  - “~WTR[FOUR NUMBERS].TMP
    - 4Kb–8Mb
    - Sum of four numbers \( \equiv 0 \pmod{10} \)
Step7 Infections

- Hook specific APIs that are used to open project files inside the s7tgtopx.exe process
- Cause Stuxnet to execute when the project is loaded
Step 7 and PLC Communication
Modifying PLCs

Step 7
request code block from PLC
show code block from PLC to user
modified STL code block

stuxnet
s7otbxdx.dll

original but renamed
s7otbxsx.dll

s7blk_read
STL code block

s7blk_read
STL code block

PLC
STL code block
Criteria for Targets

- Specific slave variable-frequency drives (frequency converter drives) to be attached to the targeted Siemens S7-300 system
- Variable-frequency drives from: Vacon (Finland) and Fararo Paya (Iran)
- Frequency of the attached motors between 807 Hz and 1210 Hz (pumps and gas centrifuges)
Attacks

• Installs malware into PLC memory
  • Monitors the Profibus messaging bus of the system
• Modifies the frequency to 1410 Hz and then to 2 Hz and then to 1064 Hz
  • Affects the operation of the connected motors by changing their rotational speed
• Installs a rootkit
  • Masks the changes in rotational speed from monitoring systems
Natanz Nuclear Facilities

• Early 2009: a "serious nuclear accident" (supposedly the shutdown of some of its centrifuges) occurred

• Sep 2010: the centrifuge operational capacity has dropped over the past year by 30 percent

• Nov 2010: uranium enrichment at Natanz had ceased several times because of a series of major technical problems
Propagation History

- Targeted attack on five organizations (12,000 infections)

### Attack Waves Against the Initial Targets

<table>
<thead>
<tr>
<th>Attack Wave</th>
<th>Site</th>
<th>Compile Time</th>
<th>Infection Time</th>
<th>Time to Infect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack Wave 1</td>
<td>Domain A</td>
<td>June, 22 2009 16:31:47</td>
<td>June 23, 2009 4:40:16</td>
<td>0 days 12 hours</td>
</tr>
<tr>
<td></td>
<td>Domain B</td>
<td>June, 22 2009 16:31:47</td>
<td>June 28, 2009 23:18:14</td>
<td>6 days 6 hours</td>
</tr>
<tr>
<td></td>
<td>Domain C</td>
<td>June, 22 2009 16:31:47</td>
<td>July 7, 2009 5:09:28</td>
<td>14 days 12 hours</td>
</tr>
<tr>
<td>Attack Wave 2</td>
<td>Domain B</td>
<td>March, 1 2010 5:52:35</td>
<td>March 23, 2010 6:06:07</td>
<td>22 days 0 hours</td>
</tr>
<tr>
<td>Attack Wave 3</td>
<td>Domain A</td>
<td>April, 14 2010 10:56:22</td>
<td>April 26, 2010 9:37:36</td>
<td>11 days 22 hours</td>
</tr>
<tr>
<td></td>
<td>Domain E</td>
<td>April, 14 2010 10:56:22</td>
<td>May 11, 2010 6:36:32</td>
<td>26 days 19 hours</td>
</tr>
<tr>
<td></td>
<td>Domain E</td>
<td>April, 14 2010 10:56:22</td>
<td>May 11, 2010 11:45:53</td>
<td>27 days 0 hours</td>
</tr>
<tr>
<td></td>
<td>Domain E</td>
<td>April, 14 2010 10:56:22</td>
<td>May 11, 2010 11:46:10</td>
<td>27 days 0 hours</td>
</tr>
<tr>
<td></td>
<td>Domain B</td>
<td>April, 14 2010 10:56:22</td>
<td>May 13, 2010 5:02:23</td>
<td>28 days 18 hours</td>
</tr>
</tbody>
</table>

The shortest span between compile time and initial infection was 12 hours
Symantec cautions readers on drawing any attribution conclusions. Attackers would have the natural desire to implicate another party.