

Статья Black Basta и неприметная доставка

 xss.is/threads/75290

Введение

По данным Check Point в конце первого полугодия 2022 года, 1 из 40 организаций во всем мире пострадала от атак программ-вымогателей, что представляет собой тревожное увеличение на 59% по сравнению с прошлым годом. Бизнес программ-вымогателей продолжает расти в гигантских масштабах из-за прибыльных платежей, требуемых и часто получаемых бандами киберпреступников. С добавлением двойного вымогательства атаки программ-вымогателей стали еще более привлекательными: даже если жертва отказывается платить, украденные личные данные могут быть проданы на форуме даркнета за значительную сумму.

Прошли те времена, когда кибератаки осуществлялись энтузиастами-одиночками, которым иногда помогали друзья и единомышленники. Как стало известно из недавней утечки Conti, бэкэнд современной высококлассной киберпреступной операции напоминает структуру гигантских ИТ-компаний, сотрудники которых могут находиться по всему миру с определенными ролями и обязанностями. Судя по вниманию к деталям, которое мы наблюдали в недавнем инциденте с Black Basta, обнаруженном группой реагирования на инциденты Check Point, операторы, стоящие за этой программой-вымогателем, также имеют впечатляющую организационную структуру.

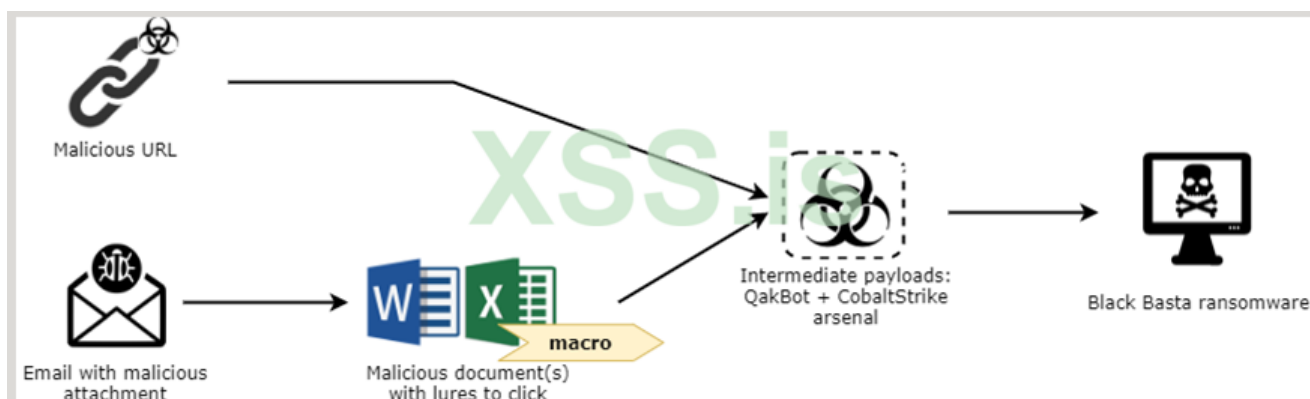
*С мая 2022 года было возбуждено более 89 дел о вымогательстве авторитетных организаций со стороны банды Black Basta. Данные показывают четкую географическую направленность группы на США и Германию; 49% жертв, перечисленных на их сайте, имеют аккаунты в США. Требование выкупа в некоторых случаях превышало 1 миллион долларов США .

Country	Number of victims	Percentage of victims
United States	44	38%
Germany	16	14%
United Kingdom	4	3.5%
Austria	3	2.6%
Canada	3	2.6%
Switzerland	3	2.6%
Denmark	2	1.74%
France	2	1.74%
India	2	1.74%
Italy	2	1.74%
Other	6	5.22%
Total:	87	100%

В статье ниже мы описываем внутреннюю работу кампании Black Basta и уделяем особое внимание этапу доставки, на котором выполняются основные приготовления для беспрепятственного запуска программы-вымогателя. Рассказываем обо всех многочисленных приемах обхода и антианализа, которые мешают эмуляторам и песочницам обнаруживать и анализировать угрозу в автоматическом режиме. Мы даем ссылки на наши энциклопедии Anti-Debug и Evasions в каждой соответствующей статье: эти сайты являются основными источниками многочисленных методов, сгруппированных по категориям, с примерами кода и возможными контрмерами. Наконец, что не менее важно, мы представляем обзор того, как Black Basta шифрует файлы в системе и как он способен к боковому перемещению .

Технические подробности

Прежде чем начнется фактическое выполнение программы-вымогателя, программа-вымогатель должна быть доставлена на компьютер жертвы. Благодаря творчеству и развитым социальным навыкам членов синдиката киберпреступников у дроппера есть разные способы доставить свою полезную нагрузку на машину выбранной жертвы. Также может быть цепочка выполнения модулей дроппера (мы наблюдали комбинацию полезной нагрузки QakBot и Cobalt Strike), что в конечном итоге приводит к выполнению программы-вымогателя.

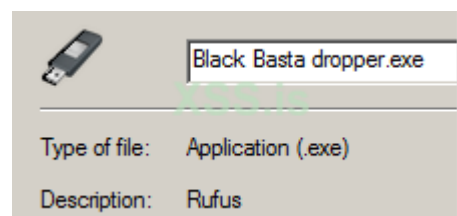


Мы заметили, что дроптеры могут быть гораздо более сложными, чем просто технически более простая полезная нагрузка программы-вымогателя. Ниже мы описываем заключительный этап доставки программы-вымогателя Black Basta.

Этап доставки

Дроптер Black Basta имитирует приложение для создания загрузочных USB-накопителей, размещенное на этом сайте - (<https://rufus.ie/en/>):

Приложение имеет цифровую подпись с тем же сертификатом (выданным "Акео Консалтинг"), который используется для законных исполняемых файлов с веб-сайта Rufus:

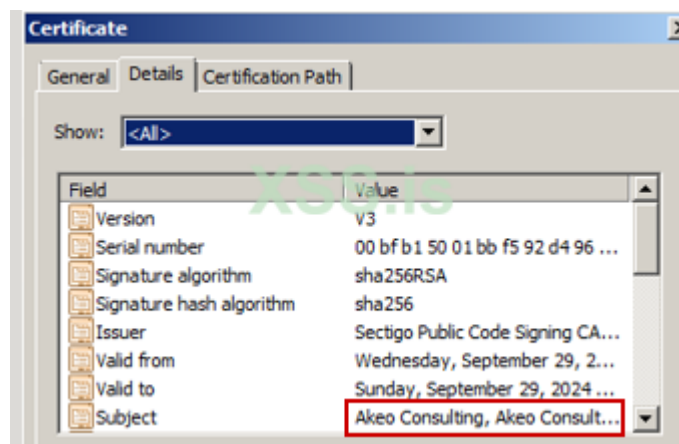


Подробнее о том, как создать вредоносное приложение с проверенной цифровой подписью, см. в специальной статье исследовательской группы Check Point (<https://research.checkpoint.com/202...signature-verification-putting-users-at-risk/>).

Методы уклонения и антианализа

В дроптере Black Basta реализовано немало антиотладочных приемов, перечисленных ниже, сгруппированных по категориям.

Щелкните ссылки для получения дополнительной информации.



<pre> if (r1_ev_peg_being_debugged() != 18534) return 28790; sub_404636(); sub_403009(); if (r1_ev_check_remote_debugger() != 18534) return 6515; if (r1_ev_findwindow_AwtFrame() != 18534) return 8825; if (r1_ev_is_debugger_present() != 18534) return 16465; if (r1_ev_peg_nt_global_flag() != 18534) return 28790; if (r1_ev_check_debug_port() != 18534) return 30295; if (r1_ev_GetWriteWatch() != 18534) return 9286; if (r1_ev_close_bad_handle() != 18534) return 21266; </pre>	<pre> if (r1_ev_eeflags() != 18534) return 30020; if (r1_ev_debugbreak() != 18534) return 30497; if (r1_ev_int_2d() != 18534) return 4969; sub_40552F(); if (r1_ev_int3() != 18534) return 9528; if (r1_ev_outputdebugstring() != 18534) return 17715; if (r1_ev_rdtsc() != 18534) return 26450; if (r1_ev_queryperfcounter() != 18534) return 9730; if (r1_ev_gettickcount() != 18534) return 16722; if (r1_ev_check_hw_regs() != 18534) return 26914; if (r1_ev_check_trap_flag() != 18534) return 22913; </pre>	<pre> sub_4045FC(); if (r1_log_pollute_beep() != 22647) return 24614; sub_401867(); sub_40286E(); sub_40229B(); sub_403EAB(); sub_401834(); if (r1_ev_load_libs_check() != 22647) return 18584; sub_4023F8(); sub_404996(); sub_403A7B(); sub_4030A0(); if (r1_ev_kernel_debugger_check() == 22647) return 22647; else return 28977; </pre>
---	--	---

Если какой-либо из этих методов успешно обнаруживает отладчик и/или среду эмуляции, дроппер останавливает свое выполнение и завершает работу, не запуская Black Basta.

Системные флаги

Эта группа методов защиты от отладки опирается на внутрипроцессные структуры для проверки состояния: выполняется ли отладка.

- **PEB: is debugger present**
- **PEB: being debugged**
- **PEB: NtGlobalFlag**
- **CheckRemoteDebugger**
- **Check kernel debugger**

Регистры процессора

Методы, сгруппированные ниже, используют регистры ЦП для проверки того, отлаживается ли процесс.

- **Set trap flag**
- **Check trap flag, same as above. Flag is not set, just checked**
- **Check HW breakpoints (method 1 in the link)**

Инструкции процессора

Эти методы используют инструкции ЦП через прямые вызовы или оболочки, чтобы проверить, отлаживается ли процесс.

- **DebugBreak**
- **INT 2D**
- **INT3**

Проверка времени:

Эти методы выполняют проверки времени, чтобы увидеть различия между отлаживаемым процессом и тем, который выполняется без отладчика.

- **RDTSC**
- **QueryPerformanceCounter**
- **GetTickCount**

Библиотечные проверки:

Этот метод основан на предположении, что в обычной системе есть некоторые общие системные библиотеки, которые могут быть загружены без проблем, и что есть также некоторые необычные библиотеки, которые не должны присутствовать в типичной системе. Однако в среде песочницы при попытке загрузить необычную библиотеку может быть возвращен предопределенный код вместо того, что в этих случаях возвращается на неэмулируемой машине. Различия в возвращаемом коде может быть достаточно для обнаружения песочницы.

Библиотеки, которые необходимо загрузить:

- **kernel32.dll**
- **networkexplorer.dll**
- **NlsData0000.dll**

Библиотеки, которые нельзя загружать:

- **NetProjW.dll**
- **Ghofr.dll**
- **fg122.dll**

Проверки API Windows

Следующая группа методов использует функции Windows API для определения того, отлаживается ли процесс.

- **VirtualAlloc в сочетании с GetWriteWatch**
- **CloseHandle с неверным дескриптором**
- **OutputDebugString для проверки последней системной ошибки**

Загрязнение журнала:

Этот метод на самом деле не является анти-отладчиком, но усложняет анализ журнала. Суть в том, чтобы сделать случайное количество вызовов функции kernel32.beep. Вы можете увидеть больше в этом анализе песочницы (<https://www.joesecurity.org/blog/9048980422564630717#>).

Сбой проверки из-за ошибки кода

Предполагается, что эти проверки используют особенности либо среды эмуляции, либо отлаженного процесса, но не работают должным образом из-за ошибок в коде.

- FindWindow (имя класса: `—unAwtFrame`) — первый символ в имени неправильный; это должен быть `SunAwtFrame`

- NtQueryInformationProcess, проверьте DebugPort — не работает из-за неправильного имени dll

Непонятный дамп

После успешного прохождения этапа техники уклонения у дроппера есть еще одна хитрость. Полезная нагрузка Black Basta не просто распаковывается и выполняется в памяти; есть данные, расположенные перед PE-заголовком программы-вымогателя, чтобы автоматические сканеры не могли легко идентифицировать вредоносную полезную нагрузку.

00170000	00040000					Priv	RWE	RWE	
00180000	00067000					Map	R	R	\De
00220000	00005000					Map	R	R	
002E0000	00003000					Map	R	R	
003F0000	00001000					Priv	RW	RW	
00300000	00002000					Map	R	R	
00320000	00002000					Map	R	R	
00350000	00030000					Priv	RW	RW	
00450000	00101000					Map	R	R	
00600000	00009000					Priv	RW	RW	
00610000	0007C000					Map	R	R	
01210000	00012000					Priv	RW	RW	
01380000	00001000								
01381000	00058000	.text		PE header		Imag	R	RWE	
01309000	00020000	.rdata		SFX,code		Imag	R	RWE	
013F9000	000B5000	.data		data,import		Imag	R	RWE	
014AE000	00004000	.rsrc		resources		Imag	R	RWE	
01482000	00003000	.reloc				Imag	R	RWE	
014C0000	00003000					Priv	RW	RW	
01580000	00003000					Priv	RW	RW	
016B0000	00001000					Priv	RW	RW	
016F0000	00073000					Priv	RW	RW	
017F0000	00008000					Priv	RW	RW	
01930000	00087000					Priv	RW	RW	
67410000	00001000	NetProjW		PE header		Imag	R	RWE	
67411000	00018000	NetProjW	.text	SFX,code,im		Imag	R	RWE	
67429000	00001000	NetProjW	.data	data		Imag	R	RWE	
6742A000	000F6000	NetProjW	.rsrc	resources		Imag	R	RWE	
67520000	00003000	NetProjW	.reloc			Imag	R	RWE	
6B180000	00001000	webservi		PE header		Imag	R	RWE	

Как и ожидалось, команда .imgscan в WinDbg не может обнаружить PE-модуль Black Basta в памяти процесса дроппера.

```

0:000> .imgscan
MZ at 01380000, prot 00000002, type 01000000 - size 135000
MZ at 67410000, prot 00000002, type 01000000 - size 113000 ← no valid PE module detected here
  Name: NetProjW.dll
MZ at 6b180000, prot 00000002, type 01000000 - size c2000
  Name: webservices.dll
MZ at 6b250000, prot 00000002, type 01000000 - size 73000

```

После того, как все эти шаги пройдены, выполняется фактическая полезная нагрузка Black Basta.

Полезная нагрузка Black Basta

В начале выполнения программы-вымогателя создается мьютекс, чтобы убедиться, что активна только одна копия вредоносного ПО:

```

.text:0040C9C3 push offset Name ; "dsajdhas.0"
.text:0040C9C8 push eax ; bInitialOwner
.text:0040C9C9 push eax ; lpMutexAttributes
.text:0040C9CA call ds:CreateMutexW

```

В примере, который мы описываем, имя мьютекса — «dsajdhas.o».

Затем вредоносная программа устанавливает обои и присваивает файлам с расширением ".basta" собственный значок.



Образы взяты из каталога TEMP, куда их распаковывает Black Basta.

Программа-вымогатель также пытается удалить все копии теневого тома, как показано на рисунке ниже:

```
.rdata:0046... 0000003D C C:\\Windows\\SysNative\\vssadmin.exe delete shadows /all /quiet  
.rdata:0046... 0000003C C C:\\Windows\\System32\\vssadmin.exe delete shadows /all /quiet
```

Шифрование

Несколько потоков создаются для создания многопоточного процесса шифрования:

Вредоносная программа шифрует все файлы, найденные на дисках, за исключением тех, в путях которых есть следующие строки:

- Windows
- Documents and Settings
- Local Settings
- Application Data
- txt
- Boot
- txt
- jpg
- DAT
- ico



Ident	Entry	Data block
00000118	0105E8E0	7FFD7000
0000065C	0105E8E0	7FFDA000
000006D0	0105E8E0	7FFDD000
00000760	0105E8E0	7FFD6000
000009B4	0105E8E0	7FFDB000
00000C08	0105E8E0	7FFD9000
00000CD8	0105E8E0	7FFDC000
00000DB4 (main)	0105512B	7FFDE000
00000FD8	0105E8E0	7FFD8000

Потоковый шифр ChaCha20 (который, как сообщается в независимых исследованиях, быстрее, чем AES) <https://crypto.stackexchange.com/questions/34455/whats-the-appeal-of-using-chacha20-instead-of-aes> используется для шифрования с ключом, сгенерированным случайным образом для каждого зашифрованного файла. Затем этот ключ передается в шифрование RSA с жестко закодированным открытым ключом для извлечения 512 байт зашифрованного ключа ChaCha20. Этот ключ добавляется в конец зашифрованного файла:

exe.1.basta		orig.exe.bad			
Compare	Next difference	Previous difference	Font		
<input type="checkbox"/> Show only differences, with extra lines:		2	ANSI<->ANSI		
266938:	00 00 00 00 00 00 00 00	266938:	00 00 00 00 00 00 00 00
266940:	EC 8D 35 2B C8 ED 16 CF	i5+Èi.I	266940:	00 00 00 00 00 00 00 00
266948:	32 67 77 D2 2E A2 58 15	2gwò.çX.	266948:	00 00 00 00 00 00 00 00
266950:	C7 AC 7C 0A 5A 8D A3 C5	ç~ .ZfÅ	266950:	00 00 00 00 00 00 00 00
266958:	5A 56 86 32 6A 99 71 6C	ZV+2j~q1	266958:	00 00 00 00 00 00 00 00
266960:	38 B2 99 A0 30 8E 36 B2	8~" 0ž6"	266960:	00 00 00 00 00 00 00 00
266968:	08 CB E4 25 99 F7 43 19	.Ea%~+C.	266968:	00 00 00 00 00 00 00 00
266970:	A1 52 72 EF C9 57 CB DF	;RriEWÈB	266970:	00 00 00 00 00 00 00 00
266978:	07 EA 20 34 4C 12 03 06	.ê 4L...	266978:	00 00 00 00 00 00 00 00
266980:	00 00 00 00 00 00 00 00	266980:	00 00 00 00 00 00 00 00
266988:	00 00 00 00 00 00 00 00	266988:	00 00 00 00 00 00 00 00
266990:	00 00 00 00 00 00 00 00	266990:	00 00 00 00 00 00 00 00
266998:	00 00 00 00 00 00 00 00	266998:	00 00 00 00 00 00 00 00
2669A0:	00 00 00 00 00 00 00 00	2669A0:	00 00 00 00 00 00 00 00
2669A8:	00 00 00 00 00 00 00 00	2669A8:	00 00 00 00 00 00 00 00
2669B0:	00 00 00 00 00 00 00 00	2669B0:	00 00 00 00 00 00 00 00
2669B8:	00 00 00 00 00 00 00 00	2669B8:	00 00 00 00 00 00 00 00
2669C0:	00 00 00 00 00 00 00 00	2669C0:	00 00 00 00 00 00 00 00
2669C8:	00 00 00 00 00 00 00 00	2669C8:	00 00 00 00 00 00 00 00
2669D0:	00 00 00 00 00 00 00 00	2669D0:	00 00 00 00 00 00 00 00
2669D8:	00 00 00 00 00 00 00 00	2669D8:	00 00 00 00 00 00 00 00
2669E0:	00 00 00 00 00 00 00 00	2669E0:	00 00 00 00 00 00 00 00
2669E8:	00 00 00 00 00 00 00 00	2669E8:	00 00 00 00 00 00 00 00
2669F0:	00 00 00 00 00 00 00 00	2669F0:	00 00 00 00 00 00 00 00
2669F8:	00 00 00 00 00 00 00 00	2669F8:	00 00 00 00 00 00 00 00
266A00:	8E 66 A6 FF 30 8C 9A 45	žf y0GŠE	266A00:		
266A08:	C3 85 22 B2 E5 FC 65 2A	Å...~šåue*	266A08:		
266A10:	9F 23 7C 15 72 69 71 4E	Y# .riqN	266A10:		
266A18:	57 04 6A E7 57 EE D4 B4	W.jçWiÓ'	266A18:		
266A20:	58 75 0D 52 F8 4A C4 C4	Xu.ŘeJÅÅ	266A20:		
266A28:	3E E5 55 EE CD EF 6E 42	>ÅUilnB	266A28:		
266A30:	49 75 3A 11 D8 C1 C9 E5	Iu;.øÅÉÅ	266A30:		
266A38:	BC CD C6 AE 5D A2 C7 D8	WižB]ççø	266A38:		
266A40:	CB 83 63 7F D4 BD AF E8	EfcíÔ~'é	266A40:		
266A48:	6E 42 30 07 01 DC 27 6A	nB0..Ü'j	266A48:		
266A50:	4A 0C E8 01 D6 18 1C 3F	Jæ.Ö..?	266A50:		
266A58:	53 5A 28 A6 E1 BA 7E C5	SZ (iÁ°~Å	266A58:		

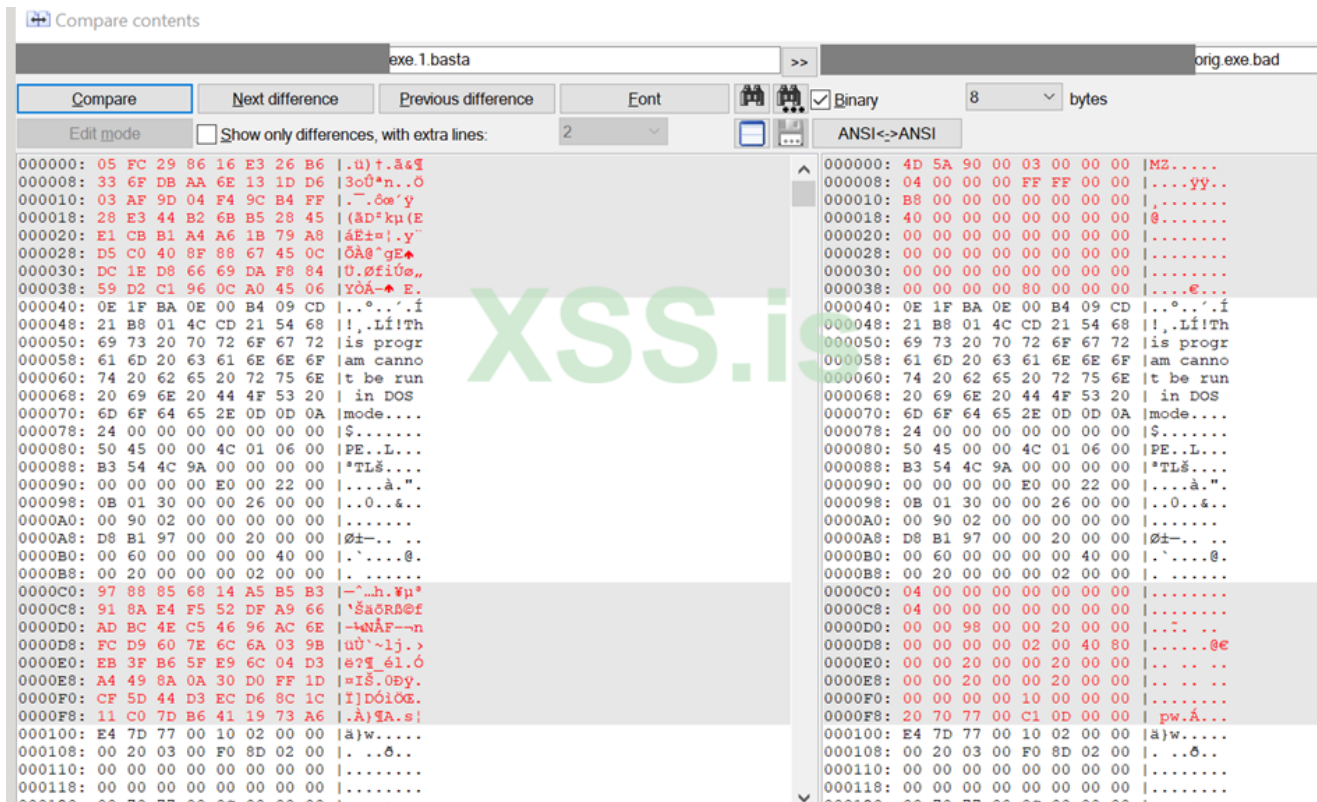
В конце блока также указана длина зашифрованного ключа (0x200):

```

266BE0: B0 D1 65 09 88 87 C7 42 | °Ñe. ^+ÇB
266BE8: 48 F6 4C 37 0E 38 CB 51 | HøL7.8ÈQ
266BF0: CC D8 2E 47 F4 69 91 40 | Ìø.Gði '\@
266BF8: 72 22 8C 31 61 45 7B F8 | r"Ø1aE{ø
266C00: 00 02 00 00 | ....

```

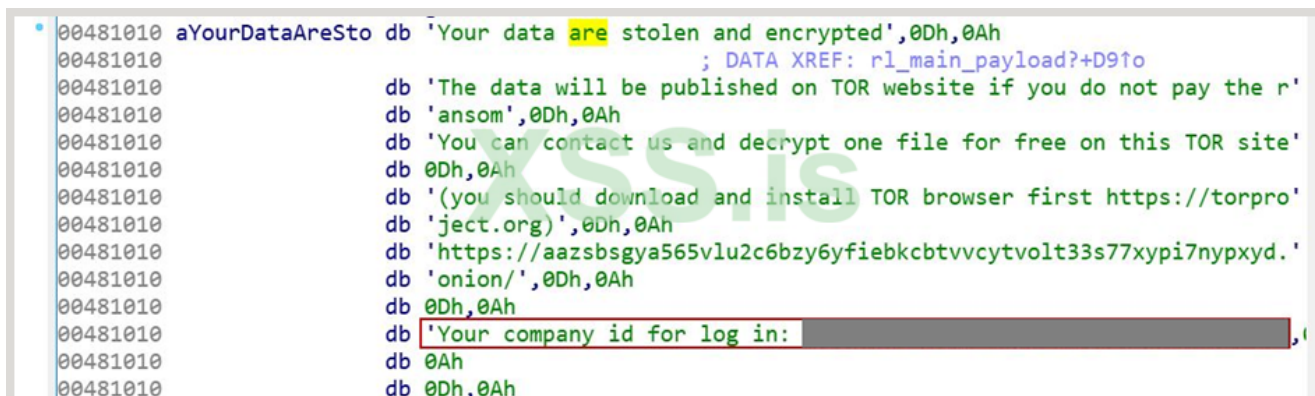
Обратите внимание, что не весь файл шифруется. Зловред нацелен на каждый третий блок из 64 байт:



Для обработки файла используются обычные функции kernel32:

- CreateFile
- ReadFile
- WriteFile
- MoveFile (to rename an encrypted file)

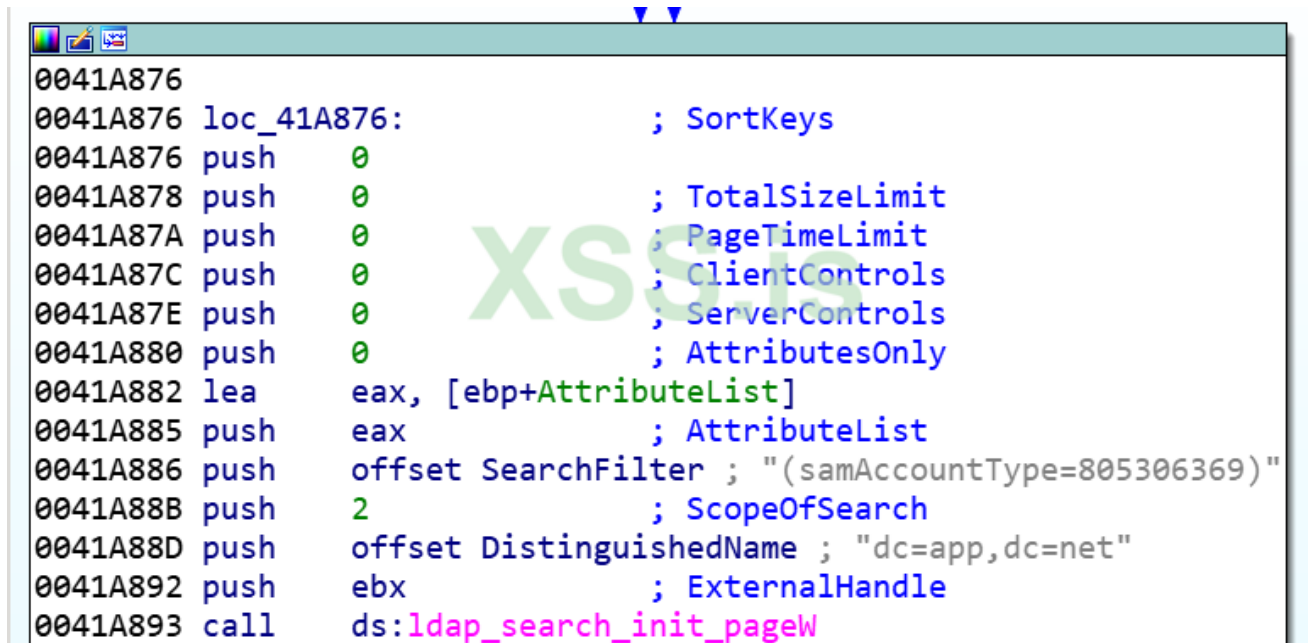
После завершения шифрования программа-вымогатель помещает примечание о выкупе в файл "readme.txt" на рабочем столе. В записке с требованием выкупа жестко прописан идентификатор компании, что является признаком целенаправленной и подготовленной атаки:



Нет очевидного способа расшифровать файлы, не зная закрытого ключа RSA.

Автоматическое распространение

В Black Basta встроен функционал автоматической раздачи в сети, если функций дропперов недостаточно для поставленной задачи. Программа-вымогатель пытается подключиться к AD с помощью LDAP API и перебирает подключенные рабочие станции, используя строку фильтра (samAccountType=805306369):



```
0041A876
0041A876 loc_41A876:                ; SortKeys
0041A876 push    0
0041A878 push    0                    ; TotalSizeLimit
0041A87A push    0                    ; PageTimeLimit
0041A87C push    0                    ; ClientControls
0041A87E push    0                    ; ServerControls
0041A880 push    0                    ; AttributesOnly
0041A882 lea    eax, [ebp+AttributeList]
0041A885 push    eax                    ; AttributeList
0041A886 push    offset SearchFilter ; "(samAccountType=805306369)"
0041A88B push    2                    ; ScopeOfSearch
0041A88D push    offset DistinguishedName ; "dc=app,dc=net"
0041A892 push    ebx                    ; ExternalHandle
0041A893 call   ds:ldap_search_init_pageW
```

После получения списка рабочих станций вымогатель пытается скопировать себя на удаленные машины по пути \\c\$\\Windows\\tmp.exe. Затем с помощью COM-объектов objectIWbemClassObject (CLSID: 4590F812-1D3A-11D0-891F-00AA004B2E24) и IWbemServices->Win32_Process через метод Create запускается скопированный на предыдущем этапе исполняемый файл .

Вывод

Атаки программ-вымогателей являются одной из самых серьезных угроз, с которыми может столкнуться жертва. Современные атаки программ-вымогателей имеют опыт многочисленных успешных вымогательств и могут перемещаться по сети в горизонтальном направлении, что приводит к все большему и большему гарантированному вознаграждению при использовании схемы двойного вымогательства.

Недавно появившаяся Black Basta уже является успешным игроком в программах-вымогателях, который принимает различные меры предосторожности и выполняет фактическое шифрование данных, о чем свидетельствуют применяемые методы

защиты от отладки и уклонения. Сочетание технических навыков, продемонстрированных бандой Black Basta, при успешном применении в атаке программ-вымогателей может привести к поистине разрушительным результатам.

Как видно из статьи, программа-вымогатель не только сама спроектирована таким образом, чтобы нанести максимальный ущерб за минимально возможное время, но и этап ее доставки скрыт, изолирован и эффективен. Black Basta без сомнения знает, что окружающая среда безопасна, и имеет все шансы выполнить шифрование.

Чтобы снизить вероятность стать жертвой этой и подобных атак, работодатели должны принять следующие меры:

- **Обучите своих сотрудников тому, как оставаться в безопасности в сфере кибербезопасности.**
- **Не открывайте некорпоративные вложения от неожиданных отправителей.**
- **Обновите и улучшите безопасность вашей киберинфраструктуры.**

Регулярно делайте резервные копии конфиденциальных данных и храните их на внешних дисках.

- **Держите свои системы в актуальном состоянии с помощью последних доступных обновлений.**

Переведено специально для XSS.IS

Автор перевода: yashechka

Источник: <https://research.checkpoint.com/2022/black-basta-and-the-unnoticed-delivery/>