From Linux to Windows - New Family of Cross-Platform Desktop Backdoors Discovered

By Stefan Ortloff

Published: 2016-01-29 · Archived: 2026-04-05 15:00:07 UTC

Background

Recently we came across a new family of cross-platform backdoors for desktop environments. First we got the Linux variant, and with information extracted from its binary, we were able to find the variant for Windows desktops, too. Not only that, but the Windows version was additionally equipped with a valid code signing signature. Let´s have a look at both of them.

DropboxCache aka Backdoor.Linux.Mokes.a

This backdoor for Linux-based operating systems comes packed via UPX and is full of features to monitor the victim's activities, including code to capture audio and take screenshots.

```
$ file DropboxCache
DropboxCache: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statical
ly linked, stripped
```

After its first execution, the binary checks its own file path and, if necessary, copies itself to one of the following locations:

- $HOME/$QT-GenericDataLocation/.mozilla/firefox/profiled
- $HOME/$QT-GenericDataLocation/.dropbox/DropboxCache

One example would be this location: $HOME/.local/share/.dropbox/DropboxCache. To achieve persistence, it uses this not very stealthy method: it just creates a .desktop-file in $HOME/.config/autostart/$filename.desktop. Here's the template for this:

```
; EkomsAutorun::service(void)::launchdContextTemplate
_ZZN12EkomsAutorun7serviceEvE22launchdContextTemplate db '[Desktop Entry]',0Ah
db 'Type=Application',0Ah
db 'Name=%1',0Ah
db 'Exec=%2',0Ah
db 'Terminal=false',0Ah,0
```

Next, it connects to its hardcoded C&C Server. From this point, it performs an http request every minute:

```
GET /v1 HTTP/1.1
Connection: Close
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:28.0) Gecko/20100101 Firefox/28.0
Accept-Encoding: gzip, deflate
Accept-Language: en-US,*
Host: xxx.xxx.xxx.xxx
```

This “heartbeat” request replies with a one-byte image. To upload and receive data and commands, it connects to TCP port 433 using a custom protocol and AES encryption. The binary comes with the following hardcoded public keys:

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1jv5Sorh9xbKNKJjlOvP
n8y+zr9Gon//f2Y6fwNcvrRyOiF6m7j4pDZue5kJku0+BqMPIBjenEzDuJBbhwdb
d3CFoWI/pvyR3rcW/XF8m0qjdVqpmvw5trgBGJrSi4TFclgebb0caxbzarfl3+Ws
9pBXtW5t0A3mmCLnxsfEqu+krxFcNJcR/arPpM4zbW+nh4r/aM/ihteL0vjpblnX
4jMdYmT5rXVi3as6dFSO3CqlMaSTC9awQTJVLUfiFX70QyxCvzukBFnNEuARn79n
xurNJxz66QJxwPrZu/Z/tB6JZrLwuuMSdmD97l6UoLqd6ETidGGyNvefvhIFdubi
qQIDAQAB
-----END PUBLIC KEY-----
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAuEQ13RmbpbinxlbqTHvR
c7DhR04EWLujNl9P0wnCrkEfP2eZE+Lm6LgjX2qlgBjfIn4tXUefK4a3i+OCA2X1
vPr2mCSlInc4U5ZRL0swqLbwU6Dp6Yt1TNRqmb19p7gvOwvP7Flka+kxnHM47IHf
9+jeIplk/irkFco7WY7pC9OLLkZ8/1tWB6yhJYmlDvRRRQjiyQm+q2ywk5Vra8pJ
uJohzW23i5u/yrXyQ6/X8/0xFn3WkBpEF2V6VNKg1pDrtQlDmtWSV47of6GMKmxr
t9ESPWie6sWAOzNwFiOtpdnp9ggkIJRN+DEEEfZh7sgoFnEy/Ev2HC5R/zUNjwwV
2QIDAQAB
-----END PUBLIC KEY-----
```

The malware then collects gathered information from the keylogger, audio captures and screenshots in /tmp/. Later it will upload collected data to the C&C.

- /tmp/ss0-DDMMyy-HHmmss-nnn.sst (Screenshots, JPEG, every 30 sec.)
- /tmp/aa0-DDMMyy-HHmmss-nnn.aat (Audiocaptures, WAV)
- /tmp/kk0-DDMMyy-HHmmss-nnn.kkt (Keylogs)
- /tmp/dd0-DDMMyy-HHmmss-nnn.ddt (Arbitrary Data)

DDMMyy = date: 280116 = 2016-01-28
HHmmss = time: 154411 = 15:44:11
nnn = milliseconds.

```
loc_411D48:             ; QAudioInput::stateChanged(QAudio::State)
lea     rax, _ZN11QAudioInput12stateChangedEN6QAudio5StateE
mov     edi, 20h
mov     r12, [rbx+18h]
mov     [rsp+98h+var_60], 0
mov     [rsp+98h+var_50], 0
mov     [rsp+98h+var_68], rax
lea     rax, _ZN14AbAudioCapture14onStateChangedEN6QAudio5StateE ; AbAudioCapture:
mov     [rsp+98h+var_58], rax
call    _Znwm           ; operator new(ulong)
mov     r9, rax
mov     rax, [rsp+98h+var_50]
mov     rdx, [rsp+98h+var_58]
lea     rcx, _ZN9QtPrivate11QSlotObjectIM14AbAudioCaptureFvN6QAudio5StateEENS_4Lis
lea     r8, [rsp+98h+var_58]
mov     dword ptr [r9], 1
mov     rsi, r12
mov     rdi, rbp
mov     [r9+18h], rax
lea     rax, _ZN11QAudioInput16staticMetaObjectE ; QAudioInput::staticMetaObject
mov     [r9+10h], rdx
lea     rdx, [rsp+98h+var_68]
mov     [r9+8], rcx
mov     rcx, rbx
mov     [rsp+98h+var_90], 0
mov     [rsp+98h+var_98], 0
mov     [rsp+98h+var_88], rax
call    _ZN7QObject11connectImplEPKS_PPvS1_S3_PN9QtPrivate15QSlotObjectBaseEN2Qt14
mov     rdi, rbp
call    _ZN11QMetaObject10ConnectionD2Ev ; QMetaObject::Connection::~Connection()
mov     rdi, [rbx+18h]
movsd   xmm0, cs:qword_B84530
mov     dword ptr [rbx+0C0h], 0
call    _ZN11QAudioInput9setVolumeEd ; QAudioInput::setVolume(double)
mov     rdi, [rbx+18h]
call    _ZN11QAudioInput5startEv ; QAudioInput::start(void)
test    rax, rax
mov     [rbx+20h], rax
jz      loc_411F18
```

This part of the code is able to capture audio from the victim's box.

However, audio capturing is not activated in the event timer of this binary, just like the keylogging feature. Since the authors have statically linked libqt, xkbcommon (the library to handle keyboard descriptions) and OpenSSL (1.0.2c) to the binary, the size of the binary is over 13MB. The criminals also didn't make any effort to obfuscate the binary in any way. In fact, the binary contains almost all symbols, which is very useful during analysis.

Function name	Segment
EkomsAutorun::EkomsAutorun(void)	.text
EkomsAutorun::service(void)	.text
EkomsAutorun::~EkomsAutorun()	.text
EkomsAutorun::~EkomsAutorun()	.text
EkomsCcClient::EkomsCcClient(void)	.text
EkomsCcClient::executeCommand(QString const&)	.text
EkomsCcClient::fileToPostData(QFileInfo const&)	.text
EkomsCcClient::runSession(QList<QFileInfo>,EkomsConnection::Met...	.text
EkomsCcClient::sendRequest(EkomsConnection::Method,EkomsCon...	.text
EkomsCcClient::service(void)	.text
EkomsCcClient::~EkomsCcClient()	.text
EkomsCcClient::~EkomsCcClient()	.text
EkomsConnection::EkomsConnection(EkomsConnection::Method)	.text
EkomsConnection::close(void)	.text
EkomsConnection::closeQt(void)	.text
EkomsConnection::connect(EkomsConnection::ProxyType,QString c...	.text
EkomsConnection::connectQt(EkomsConnection::ProxyType)	.text
EkomsConnection::createFullUri(void)	.text
EkomsConnection::createRequestHeaders(bool)	.text
EkomsConnection::getSystemUserAgent(void)	.text
EkomsConnection::recvResponse(QByteArray &)	.text
EkomsConnection::recvResponseQt(QByteArray &)	.text

There are also references to the author’s source files:

```
.init:0000000000409748 ; Source File : 'main.cpp'
.init:0000000000409748 ; Source File : 'ekomsinstaller.cpp'
.init:0000000000409748 ; Source File : 'ekomsautorun.cpp'
.init:0000000000409748 ; Source File : 'ekomsccclient.cpp'
.init:0000000000409748 ; Source File : 'ekomsutils.cpp'
.init:0000000000409748 ; Source File : 'ekomsconnection.cpp'
.init:0000000000409748 ; Source File : 'ekomsuseractivity.cpp'
.init:0000000000409748 ; Source File : 'audiocapture.cpp'
.init:0000000000409748 ; Source File : 'screencapture.cpp'
.init:0000000000409748 ; Source File : 'keyboardcapture.cpp'
.init:0000000000409748 ; Source File : 'serverconnection.cpp'
.init:0000000000409748 ; Source File : 'abstractservice.cpp'
.init:0000000000409748 ; Source File : 'proxyconnection.cpp'
.init:0000000000409748 ; Source File : 'serverconnectionthread.cpp'
.init:0000000000409748 ; Source File : 'localsettings.cpp'
.init:0000000000409748 ; Source File : 'serveruploader.cpp'
.init:0000000000409748 ; Source File : 'serveruploaderthread.cpp'
.init:0000000000409748 ; Source File : 'serveruploaderfile.cpp'
.init:0000000000409748 ; Source File : 'serverdownloaderfile.cpp'
.init:0000000000409748 ; Source File : 'moc_audiocapture.cpp'
.init:0000000000409748 ; Source File : 'moc_serverconnection.cpp'
.init:0000000000409748 ; Source File : 'evpmdwrapper.cpp'
.init:0000000000409748 ; Source File : 'networkmessagefactory.cpp'
.init:0000000000409748 ; Source File : 'networkmessage.cpp'
.init:0000000000409748 ; Source File : 'util.cpp'
.init:0000000000409748 ; Source File : 'opensslcore.cpp'
.init:0000000000409748 ; Source File : 'zlibwrapper.cpp'
.init:0000000000409748 ; Source File : 'referencecounter.cpp'
.init:0000000000409748 ; Source File : 'abstractapplication.cpp'
.init:0000000000409748 ; Source File : 'abstractnetworkconnection.cpp'
.init:0000000000409748 ; Source File : 'tcpsocketconnection.cpp'
.init:0000000000409748 ; Source File : 'abstracteventsthread.cpp'
.init:0000000000409748 ; Source File : 'networkdata.cpp'
```

Apparently, it's written in C++ and Qt, a cross-platform application framework. According to the binary's metadata it was compiled using "GCC 4.8.4 (Ubuntu 4.8.4-2ubuntu1~14.04)" on Ubuntu 14.04 LTS "Trusty Tahr". According to the qt_instdate timestamp, the last time the Qt sources were configured was on 2015-09-26 (qt/qtbase.git: deprecated), which implies the compilation time of the malware to be not earlier than end of September 2015.

We detect this type of malware as **Backdoor.Linux.Mokes.a**.

OLMyJuxM.exe aka Backdoor.Win32.Mokes.imv

Just a few days ago, we came across a rather familiar looking sample, although it was compiled for machines running Microsoft Windows. It quickly turned out to be a 32-bit Windows variant of **Backdoor.Linux.Mokes.a**.

After execution, the malware randomly chooses one of nine different locations in %AppData% to persistently install itself on the machine. The binary also creates a "version"-file in the same folder. As its name implies, it stores just version information, together with the full installation path of the malware itself:

```
%AppData%/Skype/SkypeHelper.exe
%AppData%/Dropbox/bin/DropboxHelper.exe
%AppData%/Google/Chrome/nacl32.exe
%AppData%/Google/Chrome/nacl64.exe
%AppData%/Mozilla/Firefox/mozillacache.exe
%AppData%/Hewlett-Packard/hpqcore.exe
%AppData%/Hewlett-Packard/hpprint.exe
%AppData%/Hewlett-Packard/hpscan.exe
%AppData%/Adobe/Acrobat/AcroBroker.exe

%AppData%/Adobe/Acrobat/version


"version"-Example:
00000000  00 05 01 00 43 3a 2f 44  6f 63 75 6d 65 6e 74 73  |....C:/Documents|
00000010  20 61 6e 64 20 53 65 74  74 69 6e 67 73 2f [redacted]  | and Settings/[redacted]|
00000020  [redacted] 2f 4c 6f  63 61 6c 20 53 65 74 74  |[redacted]/Local Sett|
00000030  69 6e 67 73 2f 41 70 70  6c 69 63 61 74 69 6f 6e  |ings/Application|
00000040  20 44 61 74 61 2f 41 64  6f 62 65 2f 41 63 72 6f  | Data/Adobe/Acro|
00000050  62 61 74 2f 41 63 72 6f  42 72 6f 6b 65 72 2e 65  |bat/AcroBroker.e|
00000060  78 65                                             |xe|
00000062
```

Then the corresponding registry keys are created in HKCU\Software\Microsoft\Windows\CurrentVersion\Run to ensure persistence in the system.

After the malware has executed its own copy in the new location, the SetWindowsHook API is utilized to establish keylogger functionality and to monitor mouse inputs and internal messages posted to the message queue.

The next stage in its operation is to contact the hardcoded C&C server. Besides the different IP addresses and encryption key, we see almost identical behavior.

```
-----BEGIN PUBLIC KEY-----
MIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKCAYEAtJY9Zmq93ocE0cdIVTKH
KOt0ThAG0xq1Xuci8DNORvO/ztEtlKbSCdBNHDh9I8dXKfWBBsN99eI11gxfoGPs
F3OV5fiva55lh/l8VMqmXI2z1u9McEuzxKJwUMLOk3My3v5zklkqzufVSrMNeVOH
uOQwr7pArgNagYToUqSCT6JAVCSJdt5h5oaJj6bnlWisxDOWsSSsVVn/0YeUEItB
sDVHp/0R3tcNXLkqdxTv0Zq+t7PkeaZDpwo/FbCEKQV+6JWd+YxbLWvO7nDGXVyx
7n3fkygoHLsnGzlWOMm2zWJsqvX4mbZbY+IfBdRaRlzzD6PqquDHkIrdfOiM3vwt
8iaqc1RxlJiAgiXF2F+seJwGh49ZXcmglvkhW6yLP33b/74s4BGxr/22oP8QRoPW
acfcxmV1boAiUnBDd+1VMUpo6GFTlJQPfXYd4O09YEeYspsqSKxYjWJp4s1fDmHf
w4FiOH3c1aHf6DPwmsFz1KA69COprKgjsrN8oZdZvGwpAgMBAAE=
-----END PUBLIC KEY-----
```

However, this particular variant uses a slightly different implementation and tries to obtain the default Windows user-agent string.

```
call    ds:LoadLibraryW
mov     ebx, eax
test    ebx, ebx
jz      loc_405815
push    offset ProcName ; "ObtainUserAgentString"
push    ebx             ; hModule
call    ds:GetProcAddress
mov     [esp+1Ch+ptr_ObtainUserAgentString], eax
```

If this is not successful, the sample uses its hardcoded version:

```
aMozilla5_0Comp db 'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Triden'
                                        ; DATA XREF: sub_405730:HTTP_request↑o
                db 't/6.0)',0
```

Like the Linux variant, it connects to its C&C server in the same way: once per minute it sends a heartbeat signal via HTTP (GET /v1). To retrieve commands or to upload or download additional resources, it uses TCP Port 433.

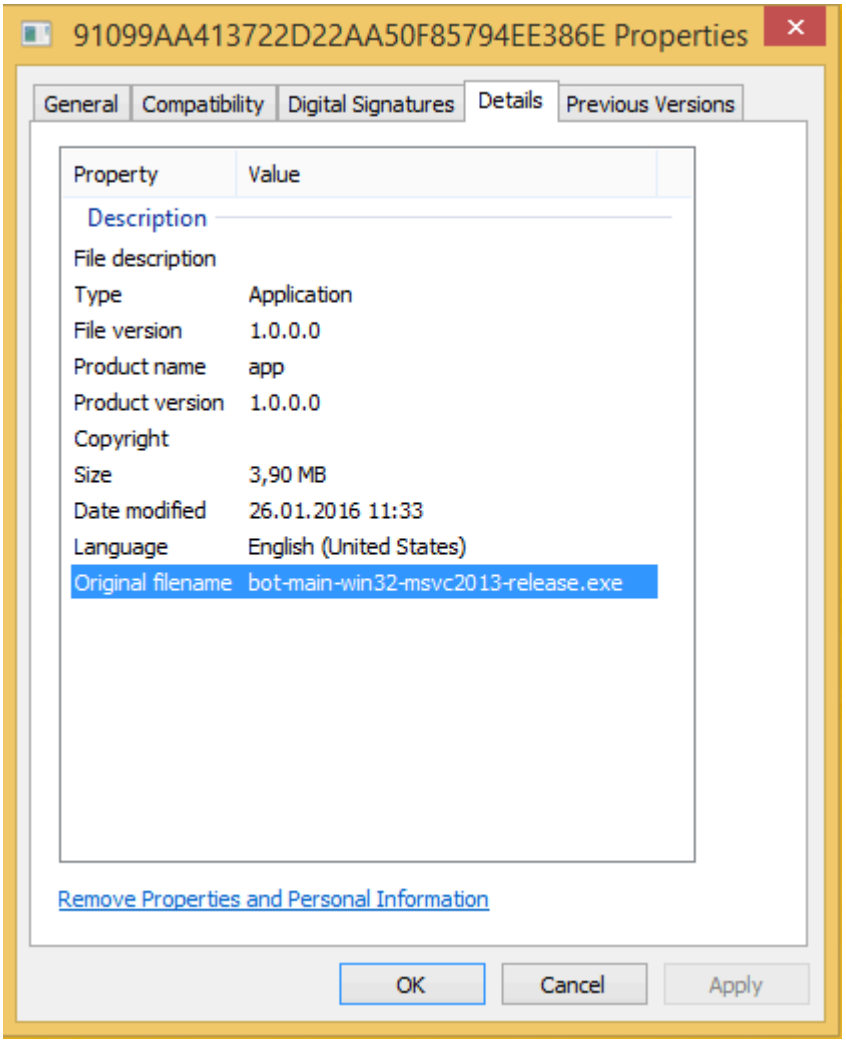
It uses almost the same filename templates to save the obtained screenshots, audiocaptures, keylogs and other arbitrary data. Unlike the Linux variant, in this sample the keylogger is active. Below you can see the content of a keystroke logfile, located in %TEMP% and created by this sample:

```
[Task Manager]
{left-ctrl}

[                                    ]
{home}{right-shift}{end}c{right-shift}:{left-ctrl}{right-alt}{↓}{left-ctrl}{left
-shift}{←}{←}{delete}{enter}{right-shift}%Appdata{right-shift}${backspace}{right
-shift}%{enter}

[Chrome]
c{right-shift}:{left-ctrl}{right-alt}ßt{backspace}{right-shift}{home}{right-shif
t}%TEMP%{enter}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl
}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{l
eft-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left
-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ct
rl}{left-ctrl}{left-shift}{left-shift}{left-shift}{left-shift}{left-shift}{left-
shift}{left-shift}{left-shift}{left-shift}{left-shift}{left-shift}{left-shift}{l
eft-shift}

[                                    ]
{left-ctrl}{left-shift}N{right-shift}{right-shift}{right-shift}Mokes{backspace}s
.ivm.{right-shift}Temp{enter}{enter}{left-ctrl}v

[Temp]
{left-shift}{↓}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl
}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{l
eft-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}{left
-ctrl}{left-ctrl}{left-ctrl}{left-ctrl}x{left-ctrl}v{left-ctrl}{left-shift}Nneu{
enter}{enter}{left-ctrl}v

[This PC]
{right-shift}%TEMP%{enter}{left-ctrl}{left-ctrl}x{left-alt}{tab}

[neu]
{left-ctrl}v
```

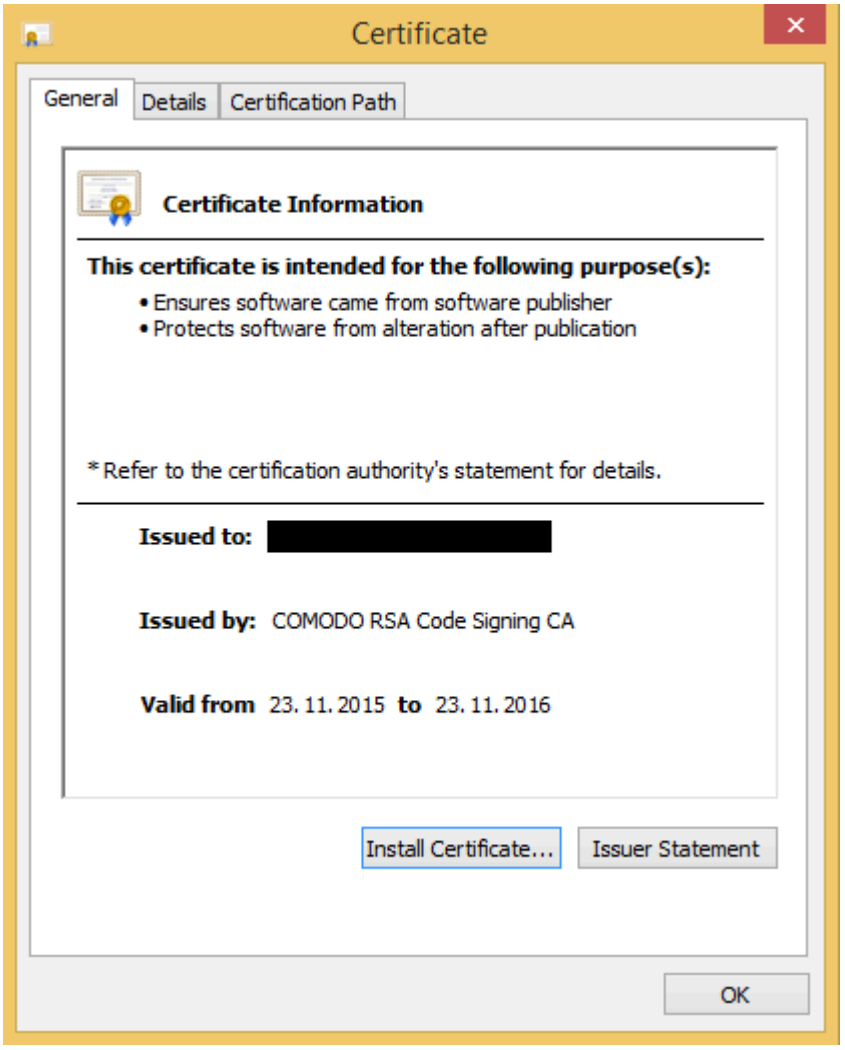
And again, we spotted some unexpected code. The following screenshot shows references to code which is able to capture images from a connected camera, such as a built-in webcam.

```
BC9D7C aDefault         db 'default',0          ; DATA XREF: CAM_SETUP_INIT+
BC9D7C                                          ; sub_421270+1F2↑o ...
BC9D84 a2statuschanged db '2statusChanged(QCamera::Status)',0
BC9D84                                          ; DATA XREF: CAM_SETUP_INIT+
BC9D84                                          ; sub_4223F0+61↑o ...
BC9DA4 a1updatereadyfo db '1updateReadyForCapture()',0
BC9DA4                                          ; DATA XREF: CAM_SETUP_INIT+
BC9DBD                  align 10h
BC9DC0 off_BC9DC0       dd offset loc_677064+6  ; DATA XREF: CAM_CAPTURE+97↑
BC9DC0                                          ; sub_6FFD20+278↑o ...
BC9DC4 aCameraNotReady db 'Camera not ready for capture',0
BC9DC4                                          ; DATA XREF: CAM_CAPTURE+1F↑
BC9DE1                  align 4
BC9DE4 aPresentframe    db 'presentFrame',0     ; DATA XREF: CAM_FRAMES+20E↑
BC9DF1                  align 4
BC9DF4 aImg_            db 'IMG_',0             ; DATA XREF: CAM_CAPTURE+AA↑
```

Similar to the Linux version, the author left quite a number of suspicious strings in the binary. The following string is surprisingly honest.



From the criminal's point of view, it's important that the software looks legitimate and that Windows doesn't asks the user for confirmation prior to execution of unknown software. On Windows machines this can be achieved by using Trusted Code Signing Certificates. In this particular case, the criminal managed to sign the binary with a trusted certificate from "COMODO RSA Code Signing CA".



We detect this type of malware as **Backdoor.Win32.Mokes.imv**.

What's next

Since this software was intentionally designed to be platform independent, we might see also corresponding Mac OS X samples in the future. **Update: We found it. See Update section below.**

Update

(2016-02-01 10:45 UTC): We just got **Backdoor.Win32.Mokes.imw**. This is the first time we see a variant of Mokes, which comes with the audio capture module activated. The malware creates a new audio file every 5 minutes.

```
$ file *.aat
aa29683-010216-093228-514.aat: empty
aa29683-010216-100727-488.aat: RIFF (little-endian) data, WAVE audio, Microsoft
PCM, 8 bit, mono 8000 Hz
aa29683-010216-101227-878.aat: RIFF (little-endian) data, WAVE audio, Microsoft
PCM, 8 bit, mono 8000 Hz
aa29683-010216-101727-893.aat: RIFF (little-endian) data, WAVE audio, Microsoft
PCM, 8 bit, mono 8000 Hz
$
```

(2016-09-07 13:19 UTC): We just come across the OS X variant of this malware and posted an analysis on this blog.

IOCs

Backdoor.Linux.Mokes.a

c9e0e5e2aeaecb232120e8573e97a6b8

$HOME/$QT-GenericDataLocation/.mozilla/firefox/profiled
$HOME/$QT-GenericDataLocation/.dropbox/DropboxCache
$HOME/.config/autostart/profiled.desktop
$HOME/.config/autostart/DropboxCache.desktop

/tmp/ss0-$date-$time-$ms.sst

Backdoor.Win32.Mokes.imv & .imw

f2407fd12ec0d4f3e82484c027c7d149 (imw)
91099aa413722d22aa50f85794ee386e (imv)

%AppData%\Skype\SkypeHelper.exe
%AppData%\Skype\version
%AppData%\Dropbox\bin\DropboxHelper.exe
%AppData%\Dropbox\bin\version
%AppData%\Google\Chrome\nacl32.exe
%AppData%\Google\Chrome\version
%AppData%\Google\Chrome\nacl64.exe
%AppData%\Google\Chrome\version
%AppData%\Mozilla\Firefox\mozillacache.exe
%AppData%\Mozilla\Firefox\version
%AppData%\Hewlett-Packard\hpqcore.exe
%AppData%\Hewlett-Packard\version
%AppData%\Hewlett-Packard\hpprint.exe
%AppData%\Hewlett-Packard\version
%AppData%\Hewlett-Packard\hpscan.exe
%AppData%\Hewlett-Packard\version
%AppData%\Adobe\Acrobat\AcroBroker.exe
%AppData%\Adobe\Acrobat\version

%TEMP%\aa$n-$date-$time-$ms.aat (imw)
where $n is a decimal hash-value calculated from the soundcard's name

%TEMP%\ss0-$date-$time-$ms.sst
%TEMP%\dd0-$date-$time-$ms.ddt

%TEMP%\kk$date.kkt

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run “%PERSISTENT-FILENAME%”, “%PERSISTENT-FILEPATH%”

where %PERSISTENT-FILENAME% is one of the filenames above
and %PERSISTENT-FILEPATH% is the corresponding path
