COZYDUKE

TLP: WHITE

CONTENTS

This whitepaper provides an overview of **CozyDuke**, a set of tools used by one or more malicious actors for performing targeted attacks against high profile organizations, such as governmental organizations and other entities that work closely with these institutions.

The CozyDuke toolset, which we believe has been under active development since at least 2011, consists of tools for infecting targeted hosts, establishing and maintaining backdoor access to the hosts, gathering information from them and gaining further access to other hosts inside the victim organization.

Based on command and control (C&C) server information found being used by CozyDuke tools, we believe the CozyDuke toolset is used by at least one malicious actor who also uses, or at the least shares, infrastructure with actors using the known threats, **MiniDuke** and **OnionDuke**.

F-SECURE LABS
SECURITY RESPONSE

Malware analysis
Whitepaper

F-Secure

INTRODUCTION

THE COZYDUKE TOOLSET

CozyDuke, as referred to in this document, is a set of tools used by one or more malicious actors for performing targeted attacks against high profile organizations. The core of the CozyDuke toolset is a modular attack platform consisting of a main component augmented by a set of additional modules implementing further functionality. The CozyDuke toolset also includes multiple kinds of droppers for infecting hosts with CozyDuke or for executing additional tools from the CozyDuke toolset. Finally, the CozyDuke toolset includes tools and scripts – some custom-written, others based on openly available tools – that are used for purposes such as gathering further information from infected hosts or for infecting additional hosts in the same target organization.

ATTACK OVERVIEW

Infection vector

We have observed CozyDuke being spread via email, which usually contain a link to a compromised website hosting a ZIP file (although in at least one case, the file was hosted on Dropbox). These files contain an executable that, upon execution, will write to disk and execute CozyDuke, while at the same time presenting the user with a decoy to divert attention. The decoy is usually an uninteresting PDF, but we have also observed a Flash video of monkeys being presented as the decoy.

Target details

We have reason to believe CozyDuke is being used to target governmental organizations and entities that work closely with such bodies.

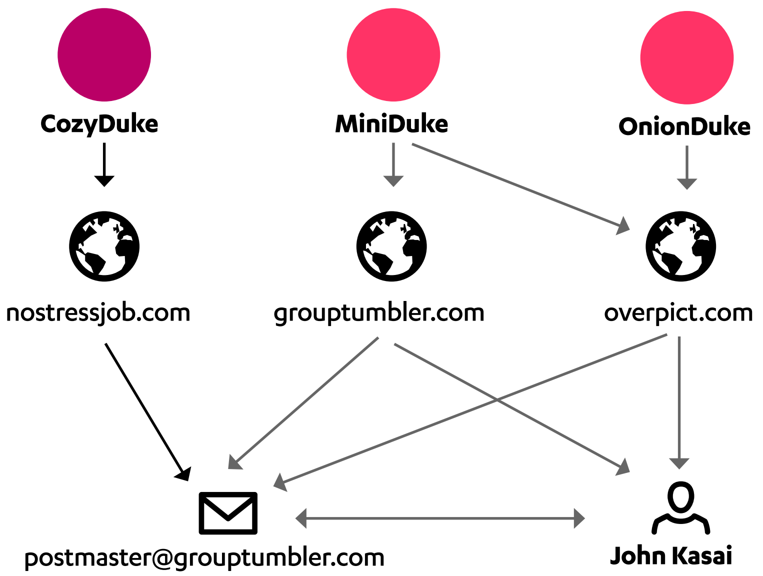
Timeline

We believe the current CozyDuke activity started at the end of January 2015. Most of the samples we have observed were compiled beginning from the end of January. However, based on comparisons of recent CozyDuke samples against older samples we have obtained dating back as far as the end of 2011, we believe CozyDuke has been under active development since at least 2011.

Attribution & links

We have strong evidence suggesting the group using CozyDuke is the same as - or at the least shares command and control infrastructure with - the group or groups using MiniDuke and OnionDuke. Firstly, a CozyDuke sample from February of 2012 attempts to contact a C&C server at *nostressjob.com*. This domain has previously been associated with known MiniDuke C&C infrastructure. This same infrastructure has also been seen in use by OnionDuke. Secondly, we also have reason to believe CozyDuke has, in some instances, downloaded and executed droppers for DLL files reminiscent of OnionDuke. Specifically, the DLLs dropped have used file and export names also used by OnionDuke. Additionally, the strings in the DLLs have been encrypted using the same algorithm used by OnionDuke. This link is, however, not as conclusive as the infrastructure overlap.

FIGURE 1: C&C INFRASTRUCTURE CONNECTIONS BETWEEN COZYDUKE, MINIDUKE AND ONIONDUKE

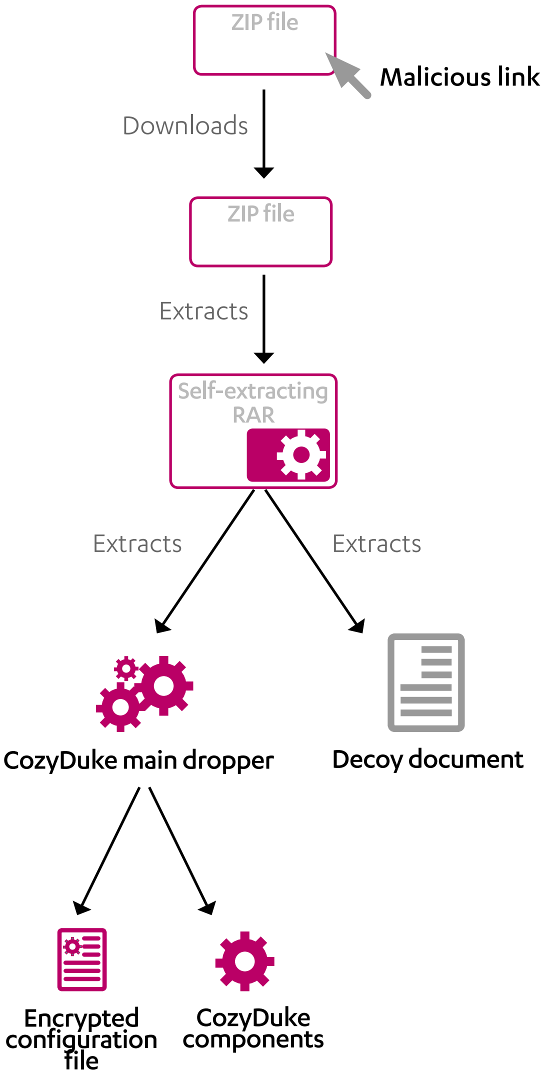


TECHNICAL DETAILS

INITIAL INFECTION

Infection with CozyDuke begins with the victim receiving an email containing a link to a ZIP file. This ZIP file will contain a single executable, usually a self-extracting RAR archive. Upon execution, it will write two files to disk. The first file is a decoy. The decoy has usually been a PDF document but Flash videos have also been observed in some cases. The second file extracted from the archive is a CozyDuke dropper. This dropper will then proceed to write to disk the main CozyDuke components as well as an encrypted configuration file used by CozyDuke.

FIGURE 2: THE COZYDUKE INFECTION FLOW



COZYDUKE MAIN DROPPER

The main CozyDuke dropper, used for infecting hosts with CozyDuke, begins by checking whether the victim has an anti-virus product installed. Should an installed product be found, it will be compared to a predetermined list of product names. If the installed product matches a name on the list, the dropper will immediately exit. Newer versions of the dropper will perform additional checks to ensure the dropper is not being executed inside a virtual machine or a known malware analysis sandbox environment. Should either of these checks fail, the dropper will likewise exit immediately.

Next, the dropper will find and decrypt encrypted data stored as two PE resources embedded in the executable. These resources are named with the hexadecimal identifiers 0x000A and 0x000B. Both resources are structured similarily. They begin with a four-byte value specifying the length of the included decryption key. This decryption key immediately follows the length field. Finally, the rest of the resource is the encrypted payload. The encryption used is a simple XOR with a rotating key.

The first resource, 0x000A, contains as its payload a Microsoft cabinet archive. This archive contains the CozyDuke components that the dropper will later install on the victim system. The second resource, 0x000B, contains as its payload an XML file with instructions for the dropper on where to install the dropped components and what to name them.

The dropper will then proceed to write the CozyDuke components to the specified location. The dropper will additionally copy the system file rundll32.exe to the install location for CozyDuke. This file will also use a name specified in the droppers configuration file. Finally, the dropper will use the copy of rundll32.exe to load and execute the CozyDuke main component.

COZYDUKE MAIN COMPONENT

Overview

The main component of CozyDuke is a DLL file responsible for orchestrating all of CozyDuke's activity on a victim machine. The main component is executed by the CozyDuke dropper using a copy of rundll32.exe. The entrypoint function varies, but is always specified in the dropper configuration.

The most important functionalities offered by the main component are establishment of persistence, gathering of basic system information, communication with the C&C server and the execution of additional tasks, modules or executables as commanded by the C&C server.

Persistence

CozyDuke may use multiple techniques for establishing persistence; the following is one technique used. Firstly, CozyDuke may set itself to be executed at system startup by adding a registry value under any of the following registry keys:

- HKLM\Software\Microsoft\Windows\CurrentVersion\Run\
- HKCU\Software\Microsoft\Windows\CurrentVersion\Run\
- HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
- HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

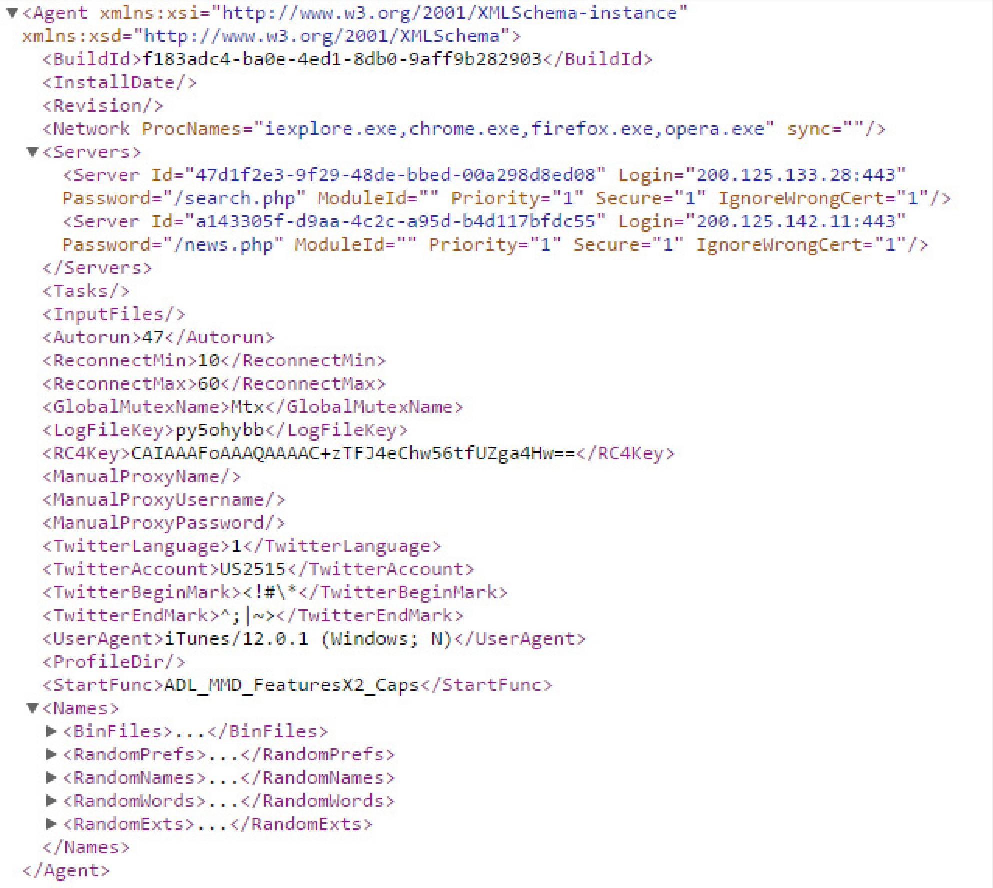
The name of the registry value will usually be the filename, (without the extension) of the CozyDuke main component.

CozyDuke may also register itself as a Windows service or scheduled task. Additionally, CozyDuke may utilize a technique known as COM-object hijacking [1] to establish persistence. To achieve this, CozyDuke will "hijack" the registry entries for the COM object "SharedTaskScheduler". CozyDuke will modify the registry entries in such a way that any loading of the SharedTaskScheduler COM object will first load a special CozyDuke module that will ensure CozyDuke stays active on the victim host.

Configuration data

The configuration data for CozyDuke is stored as a separate RC4-encrypted file that is written to disk by the CozyDuke dropper during initial infection. The name of the encrypted configuration file on disk has been racss.dat in all of the observed cases. The decrypted file is formatted as XML with the UTF-16LE character encoding.

FIGURE 3: SCREENSHOT OF A DECRYPTED COZYDUKE CONFIGURATION FILE



```
▼<Agent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <BuildId>f183adc4-ba0e-4ed1-8db0-9aff9b282903</BuildId>
   <InstallDate/>
   <Revision/>
   <Network ProcNames="iexplore.exe,chrome.exe,firefox.exe,opera.exe" sync=""/>
  ▼<Servers>
     <Server Id="47d1f2e3-9f29-48de-bbed-00a298d8ed08" Login="200.125.133.28:443"
     Password="/search.php" ModuleId="" Priority="1" Secure="1" IgnoreWrongCert="1"/>
     <Server Id="a143305f-d9aa-4c2c-a95d-b4d117bfdc55" Login="200.125.142.11:443"
     Password="/news.php" ModuleId="" Priority="1" Secure="1" IgnoreWrongCert="1"/>
   </Servers>
   <Tasks/>
   <InputFiles/>
   <Autorun>47</Autorun>
   <ReconnectMin>10</ReconnectMin>
   <ReconnectMax>60</ReconnectMax>
   <GlobalMutexName>Mtx</GlobalMutexName>
   <LogFileKey>py5ohybb</LogFileKey>
   <RC4Key>CAIAAAFoAAAQAAAAC+zTFJ4eChw56tfUZga4Hw==</RC4Key>
   <ManualProxyName/>
   <ManualProxyUsername/>
   <ManualProxyPassword/>
   <TwitterLanguage>1</TwitterLanguage>
   <TwitterAccount>US2515</TwitterAccount>
   <TwitterBeginMark><!#\*</TwitterBeginMark>
   <TwitterEndMark>^;|~></TwitterEndMark>
   <UserAgent>iTunes/12.0.1 (Windows; N)</UserAgent>
   <ProfileDir/>
   <StartFunc>ADL_MMD_FeaturesX2_Caps</StartFunc>
  ▼<Names>
    ▶<BinFiles>...</BinFiles>
    ▶<RandomPrefs>...</RandomPrefs>
    ▶<RandomNames>...</RandomNames>
    ▶<RandomWords>...</RandomWords>
    ▶<RandomExts>...</RandomExts>
   </Names>
 </Agent>
```

TABLE 1: DETAILS OF KNOWN COZYDUKE C&C SERVERS

Dropper SHA1	Protocol	Domain/IP	Path	Port
75aeaee253b5c8ae701195e3b0f49308f3d1d932	HTTP	www.sanjosemaristas. com	/app/index.php	80
75aeaee253b5c8ae701195e3b0f49308f3d1d932	HTTP	www.cifss. org	/product_thumb/index.php	80
446daabb7ac2b9f11dc1267fbd192628cc2bac19	HTTP	pvt.relance.fr	/catalogue/json/index.php	80
87668d14910c1e1bb8bbea0c6363f76e664dcd09	HTTPS	200.119.128.45	/mobile.php	443
87668d14910c1e1bb8bbea0c6363f76e664dcd09	HTTPS	202.206.232.20	/rss.php	443
ea0cfe60a7b7168c42c0e86e15feb5b0c9674029	HTTPS	www.getiton.hants. org.uk	/themes/front/img/ajax.php	80
f2ffc4e1d5faec0b7c03a233524bb78e44f0e50b	HTTPS	www.seccionpolitica. com.ar	/galeria/index.php	80
9b56155b82f14000f0ec027f29ff20e6ae5205c2	HTTPS	200.125.133.28	/search.php	443
9b56155b82f14000f0ec027f29ff20e6ae5205c2	HTTPS	200.125.142.11	/news.php	443
bf265227f9a8e22ea1c0035ac4d2449ceed43e2b	HTTPS	203.156.161.49	/plugins/twitter.php	443
32b0c8c46f8baaba0159967c5602f58dd73ebde9	HTTPS	209.40.72.2	/plugins/fsearch.php	443
78e9960cc5819583fb98fb619b33bff7768ee861	HTTPS	210.59.2.20	/search.php	443
78e9960cc5819583fb98fb619b33bff7768ee861	HTTPS	121.193.130.170	/wp-ajax.php	443
ce9d077349638ffd3e1ad68cda76c12cfb024069	HTTPS	208.75.241.246	/msearch.php	443
ac2b5928f46069111f4334f650a7dbf1b5f026d5	HTTPS	183.78.169.5	/search.php	443
ac2b5928f46069111f4334f650a7dbf1b5f026d5	HTTPS	201.76.51.10	/plugins/json.php	443
bf9d3a45273608caf90084c1157de2074322a230	HTTPS	208.77.177.24	/fsearch.php	443

Note: Domain names intentionally broken

In all CozyDuke samples from 2015, the configuration data has been encrypted with the RC4 key B5 78 62 52 98 3E 24 D7 3B C6 EE 7C B9 ED 91 62. In CozyDuke samples from July of 2014, the RC4 key has been embedded in the encrypted configuration file. In this case, the encrypted configuration file will begin with a 4-byte value specifying the length of the included RC4 key. This will be followed by the actual key. Finally, the rest of the file will be the actual configuration data.

Command and control communication

CozyDuke's main method of communicating with its command and control server is using either HTTP or HTTPS. The method of communication, as well as the address to connect to, are specified in CozyDuke's configuration data. In the cases we have observed, the configuration data for any single CozyDuke instance has included the details of either one or two C&C servers. Listed above are details of known CozyDuke C&C servers. We believe all of the ones listed are compromised servers.

In addition to its main communication method, CozyDuke also features the ability to use Twitter as a backup C&C channel. In cases where CozyDuke utilizes this functionality, the twitter account to be used will be specified in CozyDuke's configuration data. We have only observed two samples where a backup Twitter account was actually specified in the configuration data. These accounts were @US2515 and @monkey_drive.

Tasks

CozyDuke's primary purpose is the execution of tasks. These tasks usually involve the execution of modules or executables providing additional functionality. The main difference between the two is that modules are DLL files loaded in memory by the CozyDuke main component, whereas executables are PE executable files that CozyDuke will write to disk and execute.

For the purpose of managing tasks, the main component of CozyDuke implements 6 commands that the C&C server can specify. These commands are briefly described in Table 2 (overleaf).

REFERENCE

I. GData; Paul Rascagneres; *COM Object hijacking: the discreet way of persistence: An Analysis of a new persistence mechanism in the wild*; published 30.10.2014; https://blog.gdatasoftware.com/blog/article/com-object-hijacking-the-discreet-way-of-persistence.html

TABLE 2: 6 COZYDUKE C&C SERVER COMMANDS

Command	Purpose
Add	Add task
Delete	Delete task
Stop	Stop task
Modify	Modify task or configuration
Upload	Upload data
Download	Download data

The following chapter, *CozyDuke Tasks*, provides further information on the tasks CozyDuke has been observed executing.

COZYDUKE TASKS

Based on samples obtained from our own collections and generously shared to us by a trusted source, we have been able to identify some of the tasks executed recently by CozyDuke. These tasks can be divided into two categories. The first consists of CozyDuke modules which are DLL files loaded in memory by the main component of CozyDuke for the purposes of extending CozyDuke's functionality. The second category consists of PE executable files that CozyDuke will write to disk and execute on the infected host. These executables are usually droppers similar to those used to infect a host with CozyDuke. In this case, however, instead of dropping CozyDuke, the executables may drop other executables, scripts or DLLs. Once executed by CozyDuke, these executables function independently of CozyDuke's main component.

Modules

Listed below are the modules we have observed being used:

TABLE 3: COZYDUKE MODULES

Module Type	Purpose
Command execution module	Can be used to execute arbitrary commands by invoking C:\Windows\System32\cmd.exe
Password stealer module	Will attempt to harvest stored credentials from the victim
NTLM stealer module	Will attempt to harvest credentials stored on the victim host that are used as part of Windows NTLM user authentication
System info module	Will attempt to gather comprehensive information on the victim host's configuration
Screenshot module	Will take a screenshot of the victim host

Executables

We have observed executables executed by CozyDuke for the following purposes:

- Dropping and executing scripts for collecting information and credentials from the victim organization's Active Directory environment
- Dropping and executing scripts for further penetrating the victim organization with the help of commonly available tools such as Mimikatz and PSExec
- Dropping and executing additional malware with the same file and export naming conventions as OnionDuke and the same string encryption algorithm as OnionDuke

These executables will implement their own methods of C&C communication and data exfiltration. In many cases, the scripts utilized Microsoft OneDrive accounts for data transfer.

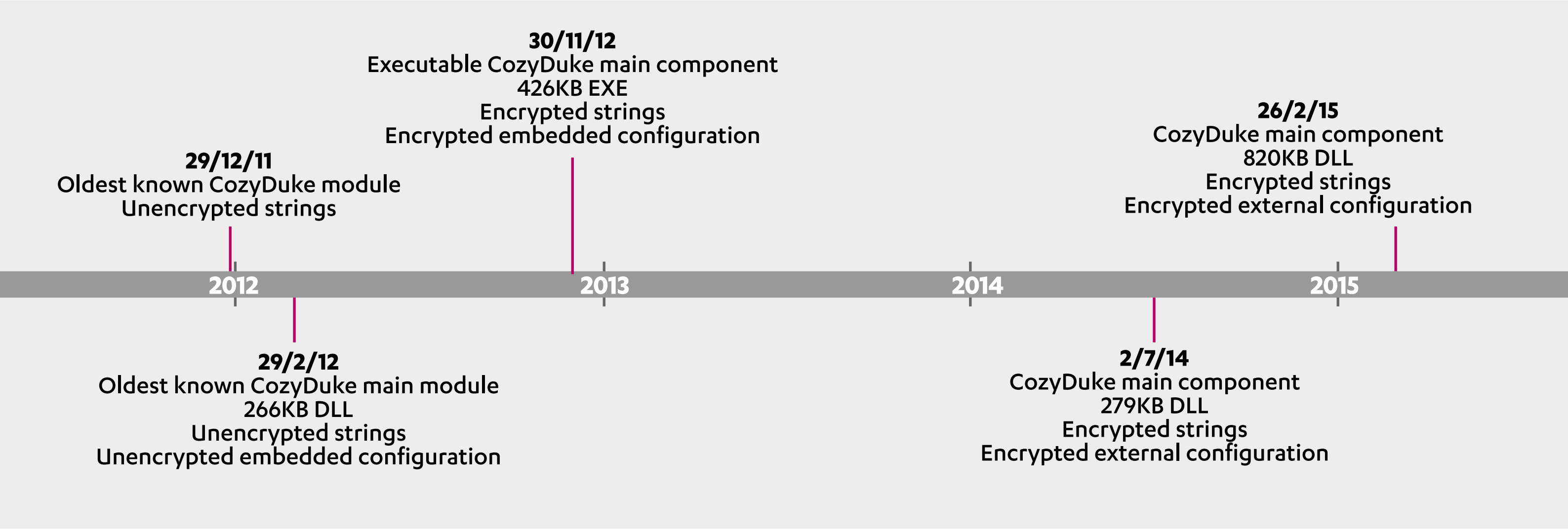
COZYDUKE EVOLUTION

The earliest CozyDuke sample we have observed so far was compiled on the 29th of December, 2011. The sample in question was not an actual CozyDuke main component, but a CozyDuke module. The earliest main component we observed was compiled on the 29th of February, 2012. The main component in question stores its strings and configuration in an unencrypted form. Additionally, the configuration is stored embedded in the binary, not as a separate XML file in the manner of newer CozyDuke versions. Even this oldest main component does however use XML for other purposes via the open-source Pugi-XML library.

The next CozyDuke main component we observed was compiled on the 30th of November, 2012. By then, the authors had switched to encrypting the strings and the configuration. However, the configuration was still stored embedded in the binary. Design-wise, this sample is a bit of an outlier. Instead of being a DLL file, the main component is actually an EXE that, in addition to the configuration, embeds multiple DLLs that provide additional functionality to the main component. Apart from the different design, functionally the main component is very similar to other CozyDuke main components. It is possible the authors of CozyDuke were trying out a new design, but eventually decided to go back to the original.

The next CozyDuke main component we observed, compiled on the 2nd of July, 2014, is again a DLL file. By now, the authors of CozyDuke had switched from

FIGURE 4: TIMELINE OF COZYDUKE EVOLUTION


30/11/12
Executable CozyDuke main component
426KB EXE
Encrypted strings
Encrypted embedded configuration
29/12/11
Oldest known CozyDuke module
Unencrypted strings
26/2/15
CozyDuke main component
820KB DLL
Encrypted strings
Encrypted external configuration
2012
2013
2014
2015
29/2/12
Oldest known CozyDuke main module
266KB DLL
Unencrypted strings
Unencrypted embedded configuration
2/7/14
CozyDuke main component
279KB DLL
Encrypted strings
Encrypted external configuration

an embedded configuration to the external XML-formatted configuration file seen today. We have yet to observe samples from 2013, but we believe CozyDuke to have been under active development as well during that year. Between 2012 and 2014, the authors of CozyDuke appear to have performed significant refactoring of the CozyDuke codebase, even though functionally the differences are smaller. Examples of this refactoring include for instance the switch from using the Pugi-XML library to using Microsoft's MSXML 3.0 for XML-related functionality.

The latest CozyDuke main component we observed was compiled on the 26th of February, 2015. The core functionality of the latest sample is very similar to the previous sample from July of 2014, but the authors of CozyDuke have implemented a lot of additional functionality, as the increase in size from 279KB to 820KB would also suggest.

COZYDUKE TERMINOLOGY AND NAMING

Based on logging strings, variable naming and PDB strings found in CozyDuke samples, we observed the following:

- The internal name for CozyDuke is "Agent"
- The CozyDuke main component's functionality revolves around the execution of "tasks" that are often associated with "modules"
- It is possible that the name "Agent" is not the original internal name of CozyDuke and that the name was changed sometime in 2011 with the original project name being "Agent_NextGen"
- Sometime in 2011, CozyDuke was identified internally as being version 3

PDB strings found in early CozyDuke samples are listed in Table 4 (below).

TABLE 4: COZYDUKE PDB STRINGS

Compilation timestamp	PDB string
Mon Feb 13 13:07:04 2012 (UTC)	E:\Visual Studio 2010\Projects\Agent_NextGen\Agent2011v3\Agent2011\Agent\tasks\bin\GetPasswords\exe\GetPasswords.pdb
Wed Dec 28 13:23:04 2011 (UTC)	D:\Projects\Agent2011\Agent2011\Agent\tasks\bin\systeminfo\exe\systeminfo.pdb
Thu Jan 26 13:57:00 2012 (UTC)	\\192.168.56.101\true\soft\Agent\tasks\Screenshots\agent_screeshots\Release\agent_screeshots.pdb

APPENDIX A | SAMPLE HASHES

- 00f67deb6e435c68f8a39336c9effc45d395b134
- 01d3973e1bb46e2b75034736991c567862a11263
- 034481acd945028f4521cf0eaa3685c6202f9e19
- 04aefbf1527536159d72d20dea907cbd080793e3
- 08facf0ae484f5bc7b066bbdd382e683fdfcba77
- 0a38765d599865dabc394287e61f5e8f6ac442c5
- 1051f814b33991a1f8e551759ead44b8ee7fc2c9
- 1a3825ef1064c2bbea5169671ef62030b00875ca
- 1d734a26184005603605aab67eba76d7d5ec3b8c
- 1e02eea130d17b9afb712d846612ab8bd6972183
- 210bc99275368df7ea179055737cffc3a12a6614
- 23e20c523b9970686d913360d438c88e6067c157
- 2564d7d42384bd3dce7257ef4a0a4b0cedac635b
- 259b4679c26625c452141861014fe2f2c336462b
- 26d030c93c517d63147f502bf6536c3914698821
- 29686320a3f06030f7192ca5b4f3eb47e73cb470
- 29a91e7823046f4ec3fd6b3fd1b442eaa92f3565
- 31163d35c5a3caa5e82e1d9b0d1b4db8fbdd79fa
- 32b0c8c46f8baaba0159967c5602f58dd73ebde9
- 33beb7a410f1cd699733000b5b30b5e4eb2062ba
- 3583647ef8158e29e3c18413ece70c2851720926
- 365cbfe32a79ce41b049dd85bb30afc51ba1ea6f
- 37144694cfa953ab7acd376c033beda45cc95f4d
- 3a624b196576b03d327b43247a975da44688ffda
- 3b297f0ca7750c0c74e5f931fec1528fe1ba6bc9
- 3c8ba7ca3675ecc75855a58b9c0527d067c88f86
- 3f0be1751afa9cb0fdd6bc6fc9874dd880bc8c1b
- 41bb403d2549db95cfc6c851ef92ad26bdf2e906
- 42cfe068b0f476198b93393840d400424fd77f0c
- 42fadc443025a132f833a4a5ed8a5350f79a86cc
- 43a979aa6ab08685d9ce949c67e19bebbb3c3559
- 443bc2e77b10ae64af6321c2c7bfd311c0772503
- 44406a80f13045442ce6a28ee62a923ac8f8c56a
- 482d1624f9450ca1c99926ceec2606260e7ce544
- 4975293c49ca223013088e51b8378e935322fe93
- 49fb759d133eeaab3fcc78cec64418e44ed649ab
- 4a16674c799fae6535c82f878f6a37f94ee9a49b
- 5150174a4d5e5bb0bccc568e82dbb86406487510
- 55bd71353408cdda1bdbbd54bc70b4c595d70e56
- 56ac317ed78f8016d59cb41e9283b1c08cbf149f
- 5bcd74e0c3c661580201e7d8122d7525a1480b4c
- 5d3b82cdea4ae066efd5d127c7dd222adee62d0b
- 5d4535df615a30b87b57facf4babf8d506e86a07
- 5ffe420a3cc848024884db8e2cfed68c47368dae
- 6502bffbd1324071c7461c50a2552e48084560ae
- 662d3cb303450abae2b88699c7f48d74f84f0d5a
- 669b7c98f0f697b91e95804dacdfe55fae3f0a85
- 69c82f6ca382bd2205d55b89f2e842b4790bda62
- 6b5ef7b76b35203dd323af49bfa27cfa7e1b6376
- 6b64ed0f4e39a1c320c7cbd342a93faed9f5df86
- 71c59eaa445346251467942bac489a9d4e807f7f
- 75aeaee253b5c8ae701195e3b0f49308f3d1d932
- 75e03a17d49d1b052770a21520bc13b14fc6c607
- 7765a0869530c1a17b8fd339bbe55cc4c1bdba30
- 78e9960cc5819583fb98fb619b33bff7768ee861
- 7c710cf31f20ef7e0ad1809672255d4edfdff052
- 7c79e3205323b9917f9eedcd3d5a891d87ddf256
- 7cda99eefb5150b87278f9bcf6ac0bde534b99e8
- 80935ac2ab3cf5b2900b49f6982a6a3f4575367c
- 81affba765aa87a0d0b12b5a213f09fd51e1e9a1
- 87668d14910c1e1bb8bbea0c6363f76e664dcd09
- 883292f00e5836f99a1943a6e0164d8c6c124478
- 8ad2003b99d92dfb9d85912ee6a39c46b1ec8137
- 8b357ff017df3ed882b278d0dbbdf129235d123d
- 8ba7932a40008881a4ed975f52271c0b679eaff2
- 8bc2d5aa1f384d56f3e921bce5326de8ff4dce2d
- 8c3ed0bbdc77aec299c77f666c21659840f5ce23
- 8cc326473fd30ab5c97709e5a91fb04e18e72e96
- 8f1ac45360196a7b5a1680ff839a131394e9d9b4
- 8f467b32f1ec0f3b2efe10b3fed2a14b16075702
- 9319bf72000f8e468c182947dd5c82fb8b9ae419
- 93d53be2c3e7961bc01e0bfa5065a2390305268c
- 93ee1c714fad9cc1bf2cba19f3de9d1e83c665e2
- 94520b93510db0dc10387a65e0a46f45ab501226
- 975b86c329c537f763f94a3f12610304dd358ff8
- 9b56155b82f14000f0ec027f29ff20e6ae5205c2
- 9dc6bbc34933ffecbfbb454788bab4230fcc2c65
- 9e156f41ff9c17692c9eba5bdb67ac14f0c0473f
- 9f8f1672594a6fbac43793c857dd7718e75f328a
- a38ea2533e3dfa6339726aafd4bc2bc7e3eec529
- a7a00f35797db2db9302625be456671911896d27
- a99d8313876015fcf1b783d38fee9e9c3cde088c
- ac2b5928f46069111f4334f650a7dbf1b5f026d5
- b26bc0a3e35c474f7099bd2b066f1680f3394b14
- b2b2e5c5a6f8a07f051aab14fbec1f6607888b50
- b47e711845d03c389004c912b3fbfc59228bb18c
- b5e973df0a159ab583fc8923c796c8cbf5b535df
- ba29768a2452a0e3abde02a903e53a181ee05bc8
- bdd2bae83c3bab9ba0c199492fe57e70c6425dd3
- bf265227f9a8e22ea1c0035ac4d2449ceed43e2b
- bf9d3a45273608caf90084c1157de2074322a230
- c02b8c2bc15dd8a7110e5f1765716464bf421591
- c117608dab3ab632de8110f8981dd7e773c61d05
- c3d8a548fa0525e1e55aa592e14303fc6964d28d
- c3fde950fe7d668805b40b1680d519f20c18b899
- c5ef4c31693845d492285e5f1c7ff3c293f99976
- c62e840ffe4bba50f6584b33a877475f0ebcf558
- c6472898e9085e563cd56baeb6b6e21928c5486d
- c7b91ff3cc69dab807016aa76d0c261411ccf27d
- c8fe2296565c211e019cdad3918a5736d4b12d44
- caa1083d2f20be0858e8d3d0671c042d0455a657
- cb7652aede9b1b7d756019f44c25fb0263498313
- ccf83cd713e0f078697f9e842a06d624f8b9757e
- ce9d077349638ffd3e1ad68cda76c12cfb024069
- cebcf2f495c3b95138128d0577dcac5cde29490d
- d12e4f164a4734e8136da85001750157014d012c
- d3254f1f4c4def8c023982dfb28fa31e91b69ab5
- d5cbf554e4e700b37ddcb026d4407fcd87032d87
- d89fc09f1aa72547d4b7f022470b6c8362997a5f
- daa651188610fd9c5a6987109e7ee5504d72a35d
- e0779ac6e5cc76e91fca71efeade2a5d7f099c80
- e2d0edf2e7d4a09fad732d4113d970a56e9a6667
- e76da232ec020d133530fdd52ffcc38b7c1d7662
- e99a03ebe3462d2399f1b819f48384f6714dcba1
- ea0cfe60a7b7168c42c0e86e15feb5b0c9674029
- eb851adfada7b40fc4f6c0ae348694500f878493
- efd41300ccf4143d04664715e1de98cb416ffdd1
- f2ffc4e1d5faec0b7c03a233524bb78e44f0e50b
- f33c980d4b6aaab1dc401226ab452ce840ad4f40
- f38040c70024fe9e305af5a3687e0d5993bb9e96
- f7693e5d39db067d97cd91fb22522f94c59fda3d
- f7d47c38eca7ec68aa478c06b1ba983d9bf02e15
- fb1b1dc288d68f695f88c5ac036b3ab1c4f5e850
- feb9424386af47d550b13614c78530bc06ec876e

APPENDIX B: IOCS

Filenames

(Note: we believe many of these to be borrowed from legitimate files)

- agent_wininet_dl.exe
- amdh264enc32.bin
- amdh264enc32.dll
- amdhcp32.dll
- amdhdl32.dll
- amdmftdecoder_32.dll
- amdmftvideodecoder_32.bin
- amdmftvideodecoder_32.dll
- amdmiracast.dll
- amdocl_as32.exe
- amdocl.bin
- amdocl_ld32.exe
- amd_opencl32.dll
- amdpcom32.bin
- atiadlxx.bin
- atiadlxx.dll
- atiapfxx.exe
- atibtmon.exe
- aticalcl.dll
- aticaldd.dll
- aticalrt.dll
- aticfx32.bin
- aticfx32.dll
- atidemgy.bin
- atidxx32.bin
- atidxx32.dll
- atieclxx.exe
- atiesrxx.exe
- atiglpxx.dll
- atiicdxx.dat
- atikmdag.sys
- atimuixx.dll
- atiodcli.exe
- atiode.exe
- atioglxx.bin
- atisamu32.dll
- atiu9pag.bin
- atiuxpag.dll
- ativce02.dat
- ativvaxy_cik.dat
- ativvaxy_cik_nd.dat
- ativvsva.dat
- ativvsvl.dat
- autorun.dll
- autorun_com.dll
- autorun_curver.dll
- clinfo.exe
- coinst_13.152.dll
- observers.dll
- ovdecode.dll
- wininetp.dll

User agent strings

- Java/1.8.0_25
- Java/1.8.0_26
- iTunes/12.0.1 (Windows; N)
- Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
- Mozilla/5.0 (Windows NT 6.1; WOW64; rv:32.0) Gecko/20100101 Firefox/32.0
- Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome /34.0.1847.137 Safari/537.36

Mutexes

- Mtx
- qdfrty
- AgentMutex

F-Secure®