G Data
Red Paper 2014

# Uroburos
Highly complex espionage software with Russian roots

G Data discovers alleged intelligence agency software

G Data SecurityLabs

Contact:
intelligence@gdata.de

G Data. Security Made in Germany.

# Contents

## Executive Summary

G Data Security experts have analyzed a very complex and sophisticated piece of malware, designed to steal confidential data. G Data refers to it as Uroburos, in correspondence with a string found in the malware's code and following an ancient symbol depicting a serpent or dragon eating its own tail.

## What is Uroburos?

Uroburos is a rootkit, composed of two files, a driver and an encrypted virtual file system. The rootkit is able to take control of an infected machine, execute arbitrary commands and hide system activities. It can steal information (most notably: files) and it is also able to capture network traffic. Its modular structure allows extending it with new features easily, which makes it not only highly sophisticated but also highly flexible and dangerous. Uroburos' driver part is extremely complex and is designed to be very discrete and very difficult to identify.

## Technical complexity suggests connections to intelligence agencies

The development of a framework like Uroburos is a huge investment. The development team behind this malware obviously comprises highly skilled computer experts, as you can infer from the structure and the advanced design of the rootkit. We believe that the team behind Uroburos has continued working on even more advanced variants, which are still to be discovered.

Uroburos is designed to work in peer-to-peer mode, meaning that infected machines communicate among each other, commanded by the remote attackers. By commanding one infected machine that has Internet connection, the malware is able to infect further machines within the network, even the ones without Internet connection. It can spy on each and every infected machine and manages to send the exfiltrated information back to the attackers, by relaying this exfiltrated data through infected machines to one machine with Internet connection. This malware behavior is typical for propagation in networks of huge companies or public authorities. The attackers expect that their target does have computers cut off from the Internet and uses this technique as a kind of workaround to achieve their goal.

Uroburos supports 32-bit and 64-bit Microsoft Windows systems. Due to the complexity of this malware and the supposed spying techniques used by it, we assume that this rootkit targets governments, research institutes, or/and big companies.

## Relation to Russian attack against U.S. suspected

Due to many technical details (file name, encryption keys, behavior and more details mentioned in this report), we assume that the group behind Uroburos is the same group that performed a cyberattack against the United States of America in 2008 with a malware called Agent.BTZ. Uroburos checks for the presence of Agent.BTZ and remains inactive if it is installed. It appears that the authors of Uroburos speak Russian (the language appears in a sample), which corroborates the relation to Agent.BTZ. Furthermore, according to public newspaper articles, this fact, the usage of Russian, also applied for the authors of Agent.BTZ.

According to all indications we gathered from the malware analyses and the research, we are sure of the fact that attacks carried out with Uroburos are not targeting John Doe but high profile enterprises, nation states, intelligence agencies and similar targets.

## Probably undiscovered for at least three years

The Uroburos rootkit is one of the most advanced rootkits we have ever analyzed in this environment. The oldest driver we identified was compiled in 2011, which means that the campaign remained undiscovered for at least three years.

## Infection vector still unknown

At the current stage of the investigations it is unknown how Uroburos initially infiltrates high profile networks. Many infection vectors are conceivable. E.g. spear phishing, drive-by-infections, USB sticks, or social engineering attacks.

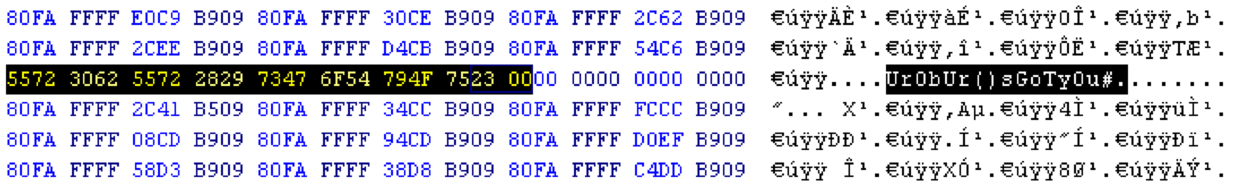
## Analysis

The G Data SecurityLabs discovered the rootkit dubbed Uroburos during 2013. We decided to investigate in depth soon after we identified the following three interesting aspects:

- the usage of virtual file systems
- the complexity of the framework
- the advanced network capabilities

## Uroburos' name

Uroburos is a direct reference to the Greek word Ouroboros (Οὐροβόρος). The Ouroboros is an ancient symbol depicting a serpent or dragon eating its own tail. The name of this rootkit is inspired by a plain text string available in several driver files: Ur0bUr()sGotyOu#



```
80FA FFFF E0C9 B909 80FA FFFF 30CE B909 80FA FFFF 2C62 B909  €úÿÿÀÈ¹.€úÿÿàÉ¹.€úÿÿ0Î¹.€úÿÿ,b¹.
80FA FFFF 2CEE B909 80FA FFFF D4CB B909 80FA FFFF 54C6 B909  €úÿÿ`Ä¹.€úÿÿ,î¹.€úÿÿÔË¹.€úÿÿTÆ¹.
5572 3062 5572 2829 7347 6F54 794F 7523 0000 0000 0000 0000  €úÿÿ....Ur0bUr()sGoTyOu#........
80FA FFFF 2C41 B509 80FA FFFF 34CC B909 80FA FFFF FCCC B909  ˝...  X¹.€úÿÿ,Aµ.€úÿÿ4Ì¹.€úÿÿüÌ¹.
80FA FFFF 08CD B909 80FA FFFF 94CD B909 80FA FFFF D0EF B909  €úÿÿÐÐ¹.€úÿÿ.Í¹.€úÿÿ˝Í¹.€úÿÿÐï¹.
80FA FFFF 58D3 B909 80FA FFFF 38D8 B909 80FA FFFF C4DD B909  €úÿÿ Î¹.€úÿÿXÓ¹.€úÿÿ8Ø¹.€úÿÿÄÝ¹.
```

***Figure 1:*** *Uroburos name string within the driver's code*

Furthermore, we identified other references to the ancient serpent/dragon symbol within the rootkit's code, for example the following strings:

- inj_snake_Win32.dll
- inj_snake_Win64.dll
- snake_alloc
- snake_free
- snake_modules_command

Another interesting notion: The exact spelling, Uroburos, can even be found in a webcomic called Homestuck. In this interactive webcomic, the reader/player needs two codes to receive virtual magic objects (called juju). Those two codes are in fact uROBuROS and UrobUros. We can notice that the uppercase and lowercase character order matches the string found within the malware code.

***Figure 2:*** *Homestuck webcomic*
*http://www.mspaintadventures.com/?s=6*

## Rootkit framework

The rootkit is basically composed of two files:

- a driver (.sys file);
- a virtual file system (.dat file).

We identified several file names for the driver, for example: Ultra3.sys, msw32.sys, vstor32.sys. We have encountered 32-bit and 64-bit driver versions. The two binaries may be installed simultaneously on one system.
The file containing the virtual file system has a random name, followed by the extension .dat. Furthermore, this file is located in the same directory as the driver file. The installation directory does change, but we were able to identify the following pattern:

- %SYSTEMROOT%\$Ntuninstall[Random_ID]$

The malware's persistence is established by the creation of a service which automatically executes during each startup of the system. The service is located in

- HKLM\System\CurrentControlSet\Services\Ultra3

The driver is needed to

- decrypt the virtual file systems
- create several hooks to hide its activities
- inject libraries in the users land
- establish and manage some communication channels

## Hiding malicious activities with the help of hooks

A rootkit naturally tries to hide its activities from the user and so does Uroburos. The driver uses inline patching to perform the hooks, which is a common way to perform this task. Inline patching is carried out by modifying the beginning of a targeted system's function in order to redirect the execution flow to a custom code before jumping back to the original function.

In the current case, the inline patching adds a new interrupt instruction (int 0xc3) at the beginning of the hooked function. Doing this, the malware adds malicious behavior to legitimate functions.

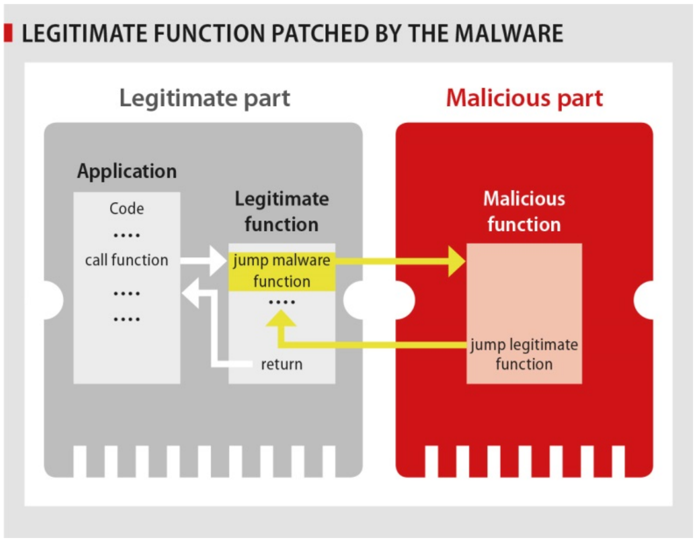
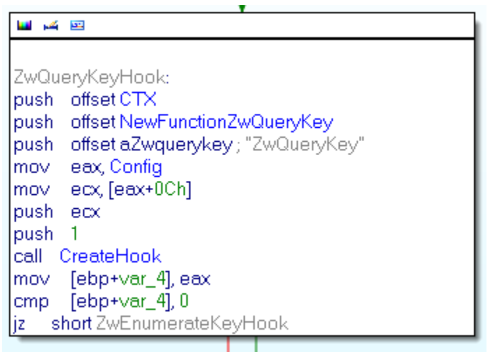


***Figure 3:*** *Hook function is called and calls, in turn, the legitimate function*

The main hooked functions are:

- `ZwQueryKey()`, `ZwEnumerateKey()`, `ZwCreateKey()` and `ZwSaveKey()`
  their purpose is to hide the persistence keys in the registry
- `ZwReadFile()`
  its purpose is to hide the driver and file system files
- `ZwQuerySystemInformation()`
  its purpose is to hide rootkit handles
- `ZwTerminateProcess()`
  its purpose is to terminate cleanly the rootkit during the shutdown of the operating system
- `ObOpenObjectByName()`
  its purpose is to hide the rootkit's virtual file systems



```
ZwQueryKeyHook:
push    offset CTX
push    offset NewFunctionZwQueryKey
push    offset aZwquerykey ; "ZwQueryKey"
mov     eax, Config
mov     ecx, [eax+0Ch]
push    ecx
push    1
call    CreateHook
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short ZwEnumerateKeyHook
```

***Figure 4:*** *ZwQueryKey() hook creation*

## Virtual file systems

The Uroburos rootkit uses two virtual file systems – one NTFS file system and one FAT file system. They are stored locally, on the infected machine. This means that the victim's computer contains an encrypted file, which, in reality, hosts another file system.

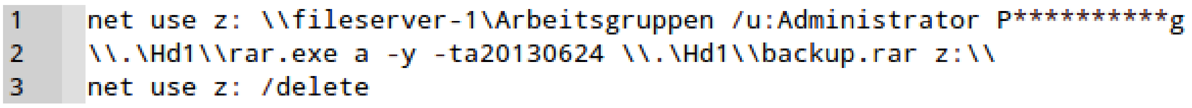
The virtual file systems are used as a work space by the attackers. They can store third party tools, post-exploitation tools, temporary files and binary output. The virtual file systems can be accessed through the devices \Device\RawDisk1 and \Device\RawDisk2 and the volume \\.\Hd1 and \\.\Hd2.

## The NTFS file system

The encryption used for the file systems is CAST-128[1]. The respective encryption key is hardcoded within the driver file. Once decrypted, the virtual file system is a classic NTFS volume, which can be simply accessed through the standard Microsoft file system APIs. During our analysis, we identified several files the file systems contained:

- .bat scripts used by the attackers
- .log files with the output of the execution of the .bat files
- third party tools
- queue files

The .bat scripts contain some net use commands to map a remote file server, netstat commands to have network information, system info commands to get a complete view of the system configuration.



```
net use z: \\fileserver-1\Arbeitsgruppen /u:Administrator P**********g
\\.\Hd1\\rar.exe a -y -ta20130624 \\.\Hd1\\backup.rar z:\\
net use z: /delete
```

***Figure 5:*** *Example of one of the .bat scripts*

[1] http://en.wikipedia.org/wiki/CAST-128

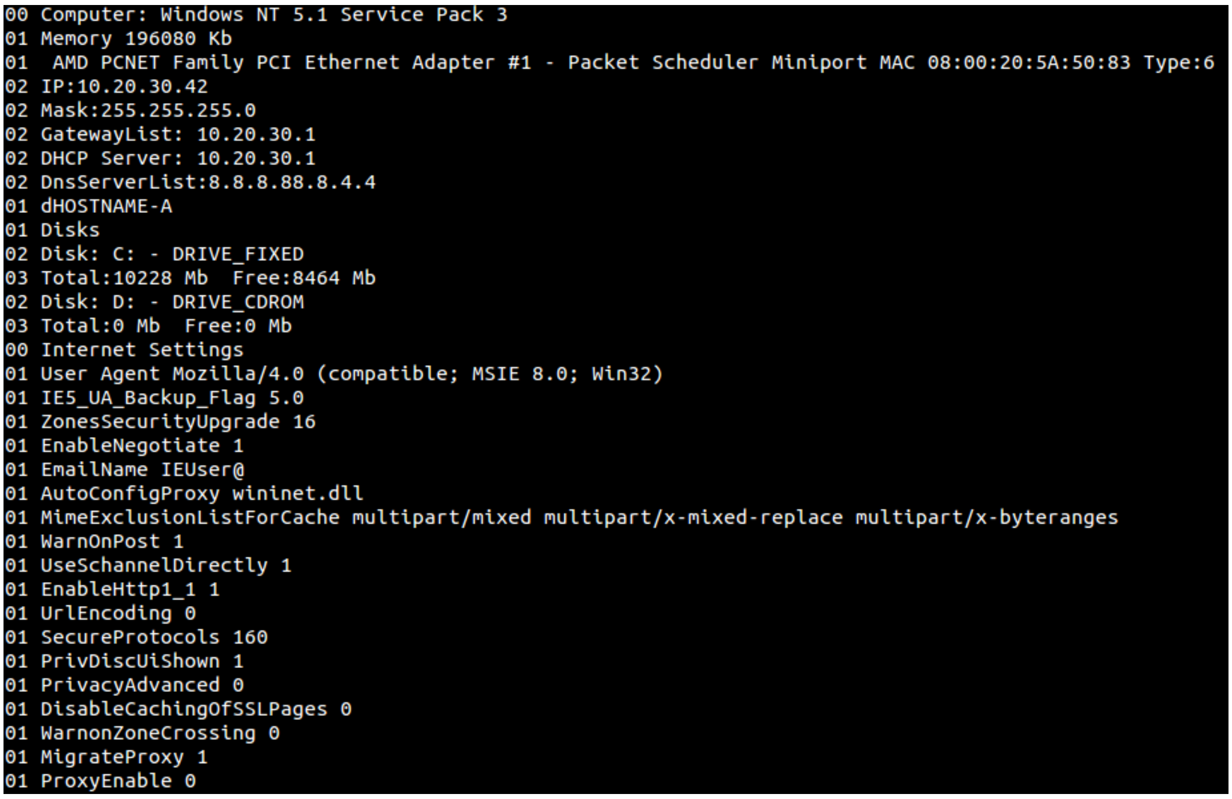
The queue file is the most interesting and complex part of the virtual file system. Each message in the queue contains a unique ID, a type, a timestamp and content. The content is also encrypted using the CAST-128 algorithm and the respective key is stored in a message, too. The messages can contain the following information:

- a key to decrypt other messages
- a configuration
- a file (or library injected in user land)
- …

## Third party tools

We found classic post-exploitation tools, used by a lot of different APT actors. The following list provides an overview of the tools found in the virtual file system:

- Dumper for NTLM (hash of a user's password). This information can be used to perform "pass the hash"[2] attacks, to compromise new systems within the infrastructure
- information gathering tools, to get information on the infected system
- RAR tools, to create archives of stolen documents
- Microsoft Office document stealer
- …



```
00 Computer: Windows NT 5.1 Service Pack 3
01 Memory 196080 Kb
01  AMD PCNET Family PCI Ethernet Adapter #1 - Packet Scheduler Miniport MAC 08:00:20:5A:50:83 Type:6
02 IP:10.20.30.42
02 Mask:255.255.255.0
02 GatewayList: 10.20.30.1
02 DHCP Server: 10.20.30.1
02 DnsServerList:8.8.8.88.8.4.4
01 dHOSTNAME-A
01 Disks
02 Disk: C: - DRIVE_FIXED
03 Total:10228 Mb  Free:8464 Mb
02 Disk: D: - DRIVE_CDROM
03 Total:0 Mb  Free:0 Mb
00 Internet Settings
01 User Agent Mozilla/4.0 (compatible; MSIE 8.0; Win32)
01 IE5_UA_Backup_Flag 5.0
01 ZonesSecurityUpgrade 16
01 EnableNegotiate 1
01 EmailName IEUser@
01 AutoConfigProxy wininet.dll
01 MimeExclusionListForCache multipart/mixed multipart/x-mixed-replace multipart/x-byteranges
01 WarnOnPost 1
01 UseSchannelDirectly 1
01 EnableHttp1_1 1
01 UrlEncoding 0
01 SecureProtocols 160
01 PrivDiscUiShown 1
01 PrivacyAdvanced 0
01 DisableCachingOfSSLPages 0
01 WarnonZoneCrossing 0
01 MigrateProxy 1
01 ProxyEnable 0
```

***Figure 6:*** *Information gathering example*

[2] http://en.wikipedia.org/wiki/Pass_the_hash

## Injected libraries - controlling the activities

The driver injects several libraries into user land. These libraries are stored in encrypted form in the queue file. These files are used to create a kind of "proxy" between the kernel land and the user land. The driver injects two noteworthy libraries:

- inj_services_Win32.dll
- inj_snake_Win32.dll

If the infected system is a 64-bit system, *Win32* is replaced by *Win64*. The libraries are very huge (more than 150 functions) and contain a lot of features. They are able to manipulate the queue file from the user land. Following, a list of functions dedicated to the queue management (*qm*):

- `qm_create()`
- `qm_enum()`
- `qm_find_first()`
- `qm_free()`
- `qm_move()`
- `qm_pop()`
- `qm_push()`
- `qm_read()`
- `qm_read_hdr()`
- `qm_reset_len()`
- `qm_rm()`
- `qm_rm_list()`
- `qm_set_dates()`
- `qm_set_parem()`
- `qm_write()`

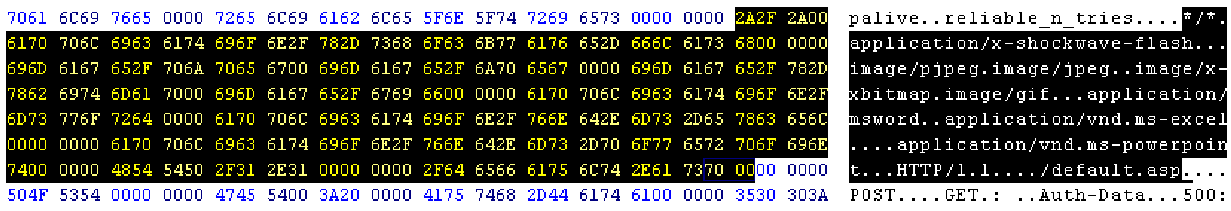
The libraries have the capability to create and manage a pcap[3] capture. The purpose of this feature is to generate a snapshot of the network traffic.

The libraries are furthermore used to exfiltrate data to the outside world, namely the attackers. We identified several protocols to perform this task: generally, the configuration needed for each protocol is stored in the queue file and not within the library itself.

- HTTP protocol
  the attackers can choose to use a website to exfiltrate data. The rootkit supports GET and POST requests and proxy authentication, too. The default URI is http://%s/default.asp but it is configurable. The media type of the request is chosen from the following list:
  - application/vnd.ms-powerpoint
  - application/vnd.ms-excel
  - application/msword
  - image/gif
  - image/x-bitmap
  - image/jpeg
  - image/pjpeg
  - application/x-shockwave-flash
  - or */*

[3] http://en.wikipedia.org/wiki/Pcap

- ICMP protocol
  the attackers can choose to use ICMP (ping) to exfiltrate data
- SMTP protocol
  the attackers can send exfiltrated data by email
- Named pipe
  the attackers can use Microsoft's named pipe to communicate to another infected machine. This case will be described in the next chapter



```
7061 6C69 7665 0000 7265 6C69 6162 6C65 5F6E 5F74 7269 6573 0000 0000 2A2F 2A00  palive..reliable_n_tries....*/*.
6170 706C 6963 6174 696F 6E2F 782D 7368 6F63 6B77 6176 652D 666C 6173 6800 0000  application/x-shockwave-flash...
696D 6167 652F 706A 7065 6700 696D 6167 652F 6A70 6567 0000 696D 6167 652F 782D  image/pjpeg.image/jpeg..image/x-
7862 6974 6D61 7000 696D 6167 652F 6769 6600 0000 6170 706C 6963 6174 696F 6E2F  xbitmap.image/gif...application/
6D73 776F 7264 0000 6170 706C 6963 6174 696F 6E2F 766E 642E 6D73 2D65 7863 656C  msword..application/vnd.ms-excel
0000 0000 6170 706C 6963 6174 696F 6E2F 766E 642E 6D73 2D70 6F77 6572 706F 696E  ....application/vnd.ms-powerpoin
7400 0000 4854 5450 2F31 2E31 0000 0000 2F64 6566 6175 6C74 2E61 7370 0000 0000  t...HTTP/1.1..../default.asp....
504F 5354 0000 0000 4745 5400 3A20 0000 4175 7468 2D44 6174 6100 0000 3530 303A  POST....GET.: ..Auth-Data...500:
```

***Figure 7:*** *HTTP media type list*

The design chosen by the developers is truly efficient: to add a new protocol and a new capability, the attackers do not need to recompile (or reinstall) the entire rootkit. They simply need to adjust the library and replace the library in the queue file with the adjusted one. The library usage results in modularity well thought out.

## Network capabilities

Thanks to the protocol described previously, the attacker can even target victims not directly connected to the Internet. The following figure shows an example of a network scheme we discovered in 2013:

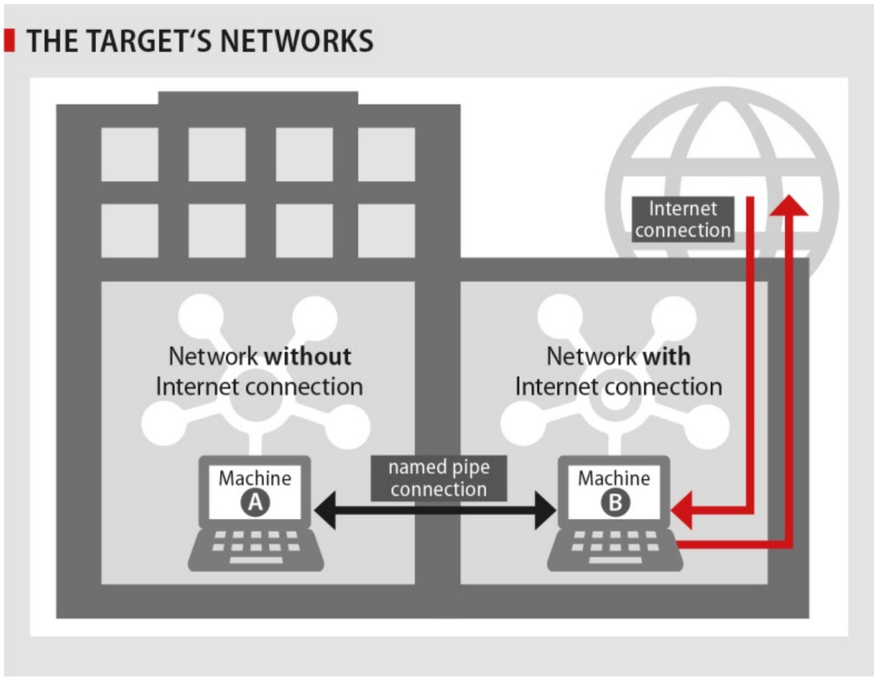


***Figure 8:*** *Uroburos' communication capabilities*

The targeted machine (A) is a machine with access to sensitive data, e.g. a server. The rootkit installed on the system opens a Microsoft named pipe and waits for an incoming connection. This machine can be named "spied-on node".

The second machine (B) is an office machine with the capability to connect to the Internet. The rootkit is configured to connect to system (A), with the help of the named pipe, and administrate the machine remotely. Finally, machine (B) is able to pass on all data received from machine (A) to the Internet. This machine (B) could be named “proxy node”.

This peer-to-peer design is really efficient, scalable and resilient. In case a “proxy node” is not available/detected, the attackers can use another infected one. The advantage for the attackers: even if a security specialist finds one “spied-on node”, he cannot easily find the “proxy node”, due to the fact that this node is a passive node. Furthermore, the analyst does not automatically have the command and control URL. In case of incident response, this design is complicated to apprehend and it is hard to contain the infection.

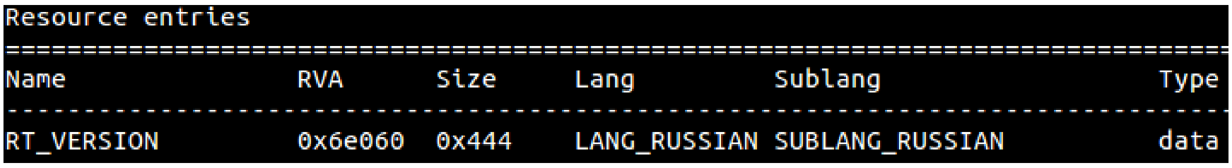
## Victims and attribution

Due to the complexity of the Uroburos rootkit, we estimate that it was designed to target government institutions, research institutions or companies dealing with sensitive information as well as similar high-profile targets.
Concerning the attribution, we found some technical information which allows us to link the Uroburos rootkit to a cyber-attack against the United States of America, carried out in 2008[4] and, particularly, to the worm used by the attackers, called Agent.BTZ. During this 2008 campaign, a USB stick was deliberately "lost" in the parking lot of the United States Department of Defense. This USB stick contained malicious code and infected the military’s network.

The following leads make us link what we discovered during our analysis with the cyber-attack carried out in 2008:

- the usage of the same obfuscation key in Uroburos and Agent.BTZ (1dM3uu4j7Fw4sjnbcwlDqet4m5Imnxl1pzxI6as80cbLnmz54cs5Ldn4ri3do5L6gs923HL34x2f5cvd0fk6c1a0s)
- the usage of the same file name to store logs: winview.ocx
- Uroburos actually checks whether Agent.BTZ is already present on the attacked system, before its installation. In case Agent.BTZ is installed, Uroburos will not be installed on the system.
- the usage of Russian language in both codes

In an article published by Reuters, in 2011, the journalist mentioned that “U.S. government strongly suspects that the original attack was crafted by Russian Intelligence.”[5] We found Uroburos samples with a resource in Russian language:



```
Resource entries
================================================================================
Name                RVA       Size      Lang          Sublang                Type
--------------------------------------------------------------------------------
RT_VERSION          0x6e060   0x444     LANG_RUSSIAN  SUBLANG_RUSSIAN        data
```

***Figure 9:*** *Resource with Russian language*

In case someone from the audience of this report notices an infection caused by the Uroburos rootkit and needs help, would like to receive further technical information or would like to contribute any information about this case, please feel free to contact us by email using the following mailbox: intelligence@gdata.de

[4] http://en.wikipedia.org/wiki/2008_cyberattack_on_United_States
[5] http://www.reuters.com/article/2011/06/17/us-usa-cybersecurity-worm-idUSTRE75F5TB20110617

## Conclusion

The Uroburos rootkit is one of the most advanced rootkits we have ever analyzed. The oldest driver we identified was compiled in 2011, which means that the campaign remained undiscovered for at least three years.

The investment to develop a complete framework such as Uroburos is extremely high. The developer team behind the development and the design of such an enhanced framework is really skilled. We believe that, until today, the team behind Uroburos has developed an even more sophisticated framework, which still remains undiscovered.

The design is highly professional; the fact the attackers use a driver and a virtual file system in two separate files which can only work in combination, makes the analysis really complicated. One needs to have the two components to correctly analyze the framework. The driver contains all of the necessary functionality and the file system alone simply cannot be decrypted.

The network design is extraordinarily efficient, too; for an incident response team, it is always complicated to deal with peer-to-peer infrastructure. It is also hard to handle passive nodes, because one cannot quickly identify the link between the different infected machines.

This kind of data stealing software is too expensive to be used as common spyware. We assume that the attackers reserve the Uroburos framework for dedicated and critical targets. This is the main reason why the rootkit was only detected many years after the suspected first infection. Furthermore, we assume that the framework is designed to perform cyber espionage within governments and high profile enterprises but, due to its modularity, it can be easily extended to gain new features and perform further attacks as long as it remains undetected within its target.

There are some strong indications which suggest that the group behind Uroburos is the same as the one behind Agent.BTZ, which allegedly was part of an intelligence agency cyberattack targeting US military bases in 2008. Notable hints include the usage of the exact same encryption key then and now, as well as the presence of Russian language in both cases.

## Technical details

SHA256: BF1CFC65B78F5222D35DC3BD2F0A87C9798BCE5A48348649DD271CE395656341

MD5: 320F4E6EE421C1616BD058E73CFEA282

Filesize: 210944

For further information contact intelligence@gdata.de