Threat Spotlight: ZeuS (aka Zbot) Infostealer Trojan

[RESEARCH & INTELLIGENCE](https://blogs.blackberry.com/en/category/research-and-intelligence) / 04.29.20 / [T.J. O'Leary](https://blogs.blackberry.com/en/author/t-j-o-leary)

* [Share on Twitter](https://twitter.com/intent/tweet?url=https://blogs.blackberry.com/en/2020/04/threat-spotlight-zeus-infostealer-trojan&text=%20Threat%20Spotlight:%20ZeuS%20(aka%20Zbot)%20Infostealer%20Trojan%20&via=BlackBerry)
* [Share on Facebook](https://www.facebook.com/sharer/sharer.php?u=https://blogs.blackberry.com/en/2020/04/threat-spotlight-zeus-infostealer-trojan)
* [Share on Linked In](https://www.linkedin.com/shareArticle?mini=true&url=https://blogs.blackberry.com/en/2020/04/threat-spotlight-zeus-infostealer-trojan&title=%20Threat%20Spotlight:%20ZeuS%20(aka%20Zbot)%20Infostealer%20Trojan%20&summary=&source=blogs.blackberry.com)
* [Email](mailto:?subject=%20Threat%20Spotlight:%20ZeuS%20(aka%20Zbot)%20Infostealer%20Trojan%20&body=https://blogs.blackberry.com/en/2020/04/threat-spotlight-zeus-infostealer-trojan)



 Threat Spotlight: ZeuS (aka Zbot) Infostealer Trojan

ZeuS (aka Zbot) is an infamous and successful information stealing Trojan. First detected in 2007, the malware’s primary focus is stealing financial/banking information and user credentials from individuals and organizations. Its exploits resulted in [the theft of billions of dollars on a global scale](https://securityintelligence.com/brazil-cant-catch-a-break-after-panda-comes-the-sphinx/)[1]. ZeuS crimeware kits vary in complexity with costs ranging from free to several thousand dollars (for [later versions with added functionality](https://www.theregister.co.uk/2011/05/10/zeus_crimeware_kit_leaked/))[2].

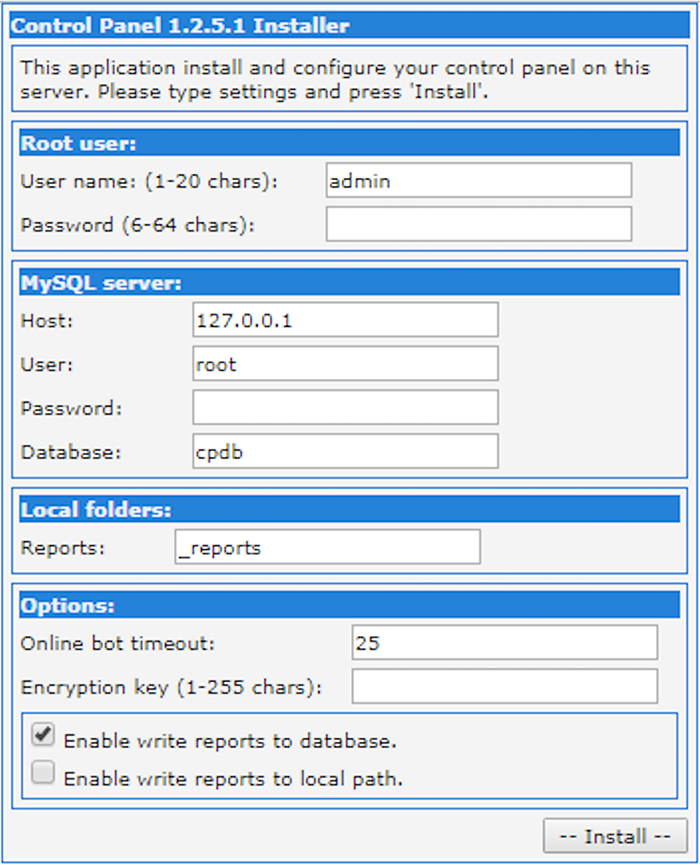
ZeuS offers users a simple-to-install, highly effective, feature packed botnet. Its ease and availability resulted in massive numbers of unrelated ZeuS-built botnets and Zbot variants in the wild. This makes attribution of ZeuS botnet crimes extremely difficult and shifts the focus away from its original developers in Eastern Europe.

This blog examines one of the earlier ZeuS kits. It provides insight into what made ZeuS one of the most effective and persistent pieces of modern malware. It also provides analysis of a Zbot sample built with a kit.

Technical Analysis

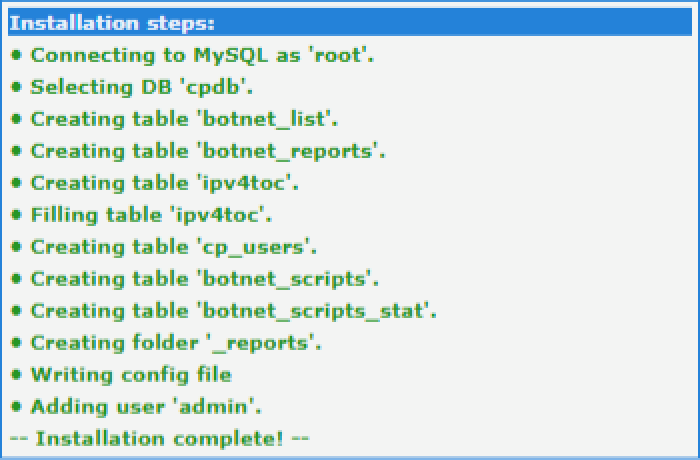
ZeuS Server

The complete installation of the ZeuS Server component is incredibly easy but beyond the scope of this blog. It requires only minimal user settings once the relevant backend services are installed and running:



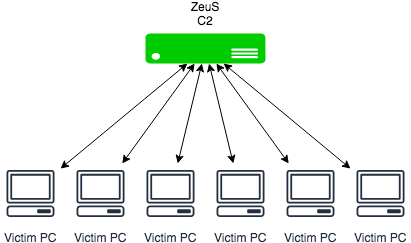
*Figure 1 - ZeuS installation page*

Once users are presented with notification of a successfully completed installation, they may begin to administer and build their network of bots:



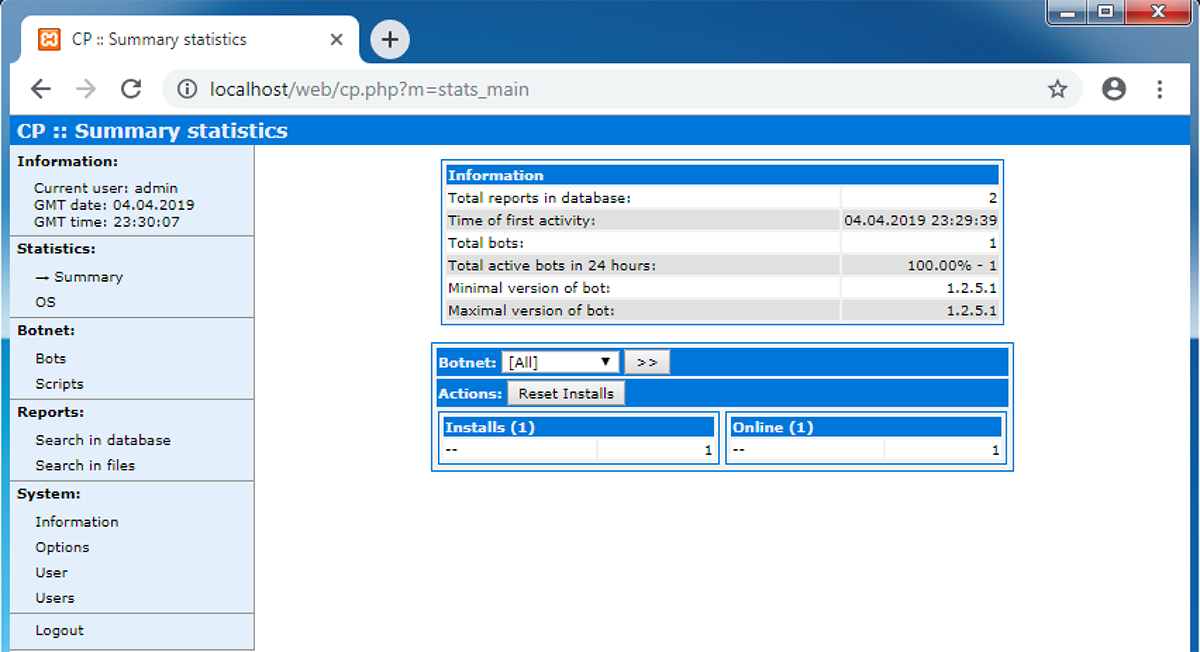
*Figure 2 - ZeuS notification of a completed installation*

The **command and control** (C2) server allows for the easy administration of an army of bots. After the bots are installed on the target PCs the C2 server provides various methods of reporting on and interacting with them. Early variants of Zeus utilized a simplistic client-server botnet model. Later variants used more complex botnet implementations featuring Domain Generation Algorithms (DGA) or Fast Fluxing along with multi-tier or peer-to-peer C2 infrastructure. Early ZeuS botnets consisted of a single centralized C2 server that controlled several hundred, if not thousands, of bots. This allowed C2 communication to be easily traced and botnets (not configured for failover to a secondary pre-configured C2 server or utilizing a Bullet Proof hosting provider) to be taken down:



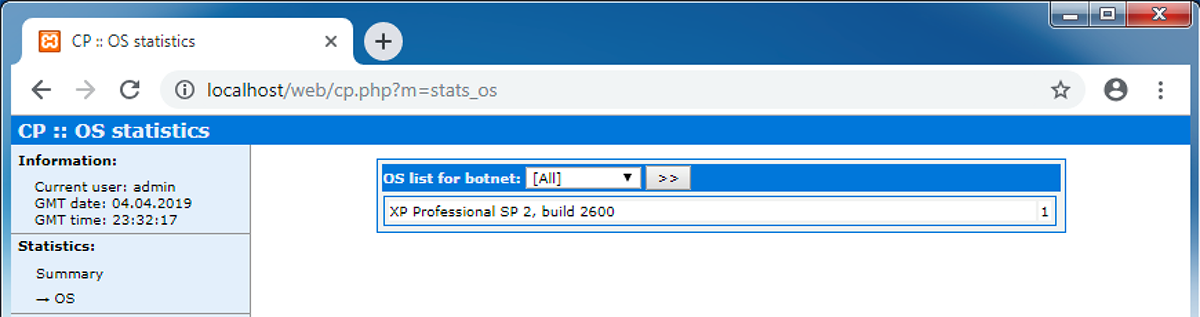
*Figure 3 - ZeuS basic C2 infrastructure*

The control panel contains information on the host system running the C2 and offers options such as user account creation, etc. It also offers some detailed insights into the botnets and the individual bots that comprise them. The summary page lists the number of bots deployed, number of reports received from infected computers, bot activity level, and bot versions:



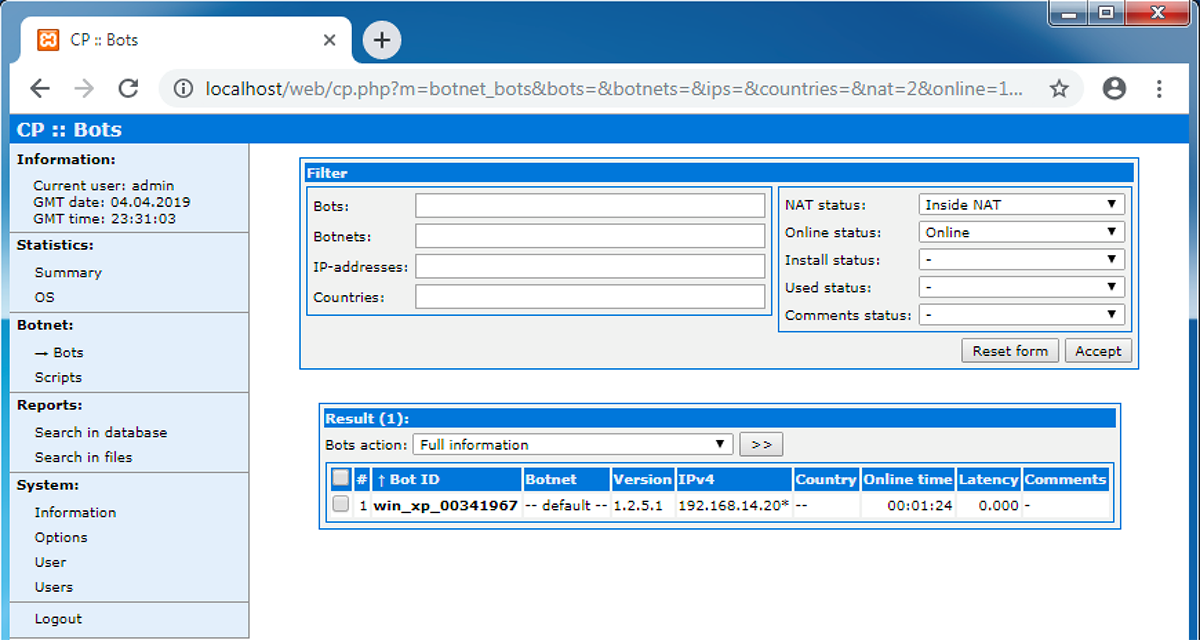
*Figure 4 - Zeus Server - summary page*

The OS version of the victim PCs are documented and searchable:



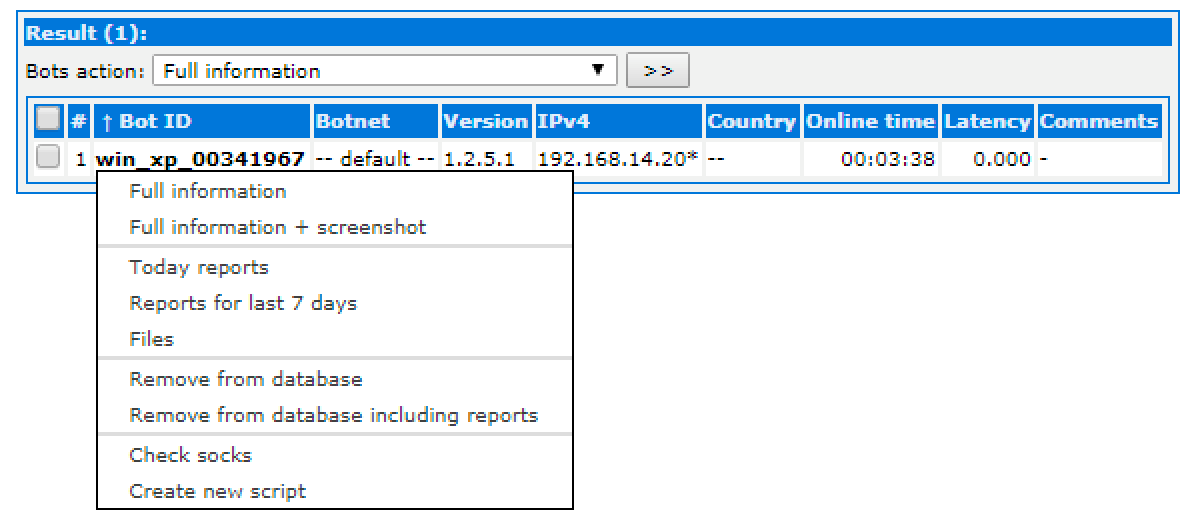
*Figure 5 - ZeuS Server - OS listings*

Fine grain filtering and searching is also available. The need for such a capability speaks to the expected size and diversity of the botnets ZeuS is intended to control. Filters exist for I.P. address, country, botnet, NAT status, and more:



*Figure 6 - ZeuS Server - bot search functionality*

Simplified bot administration and reporting allows the threat actor to pull detailed information from their botnet(s) including system information, reports, and screenshots. It also allows the creation of command scripts that target a particular botnet, subsets of bots, or bots located in a particular country:



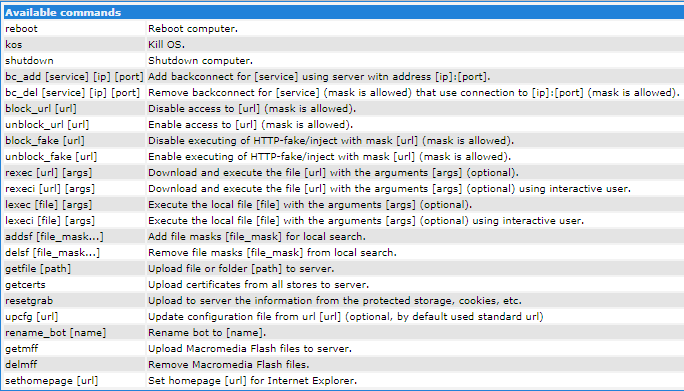
*Figure 7 - Zeus Server - bot reporting*

Various commands are understood by the bot and can be scripted together and executed on the victim PC.

The commands let attackers:

* Reboot the PC
* Kill or shutdown the OS
* Download and execute files from the Internet
* Execute local files
* Configure *BackConnect*

*BackConnect*allows attackers to cover their tracks by proxying traffic to the Internet through a victim PC:



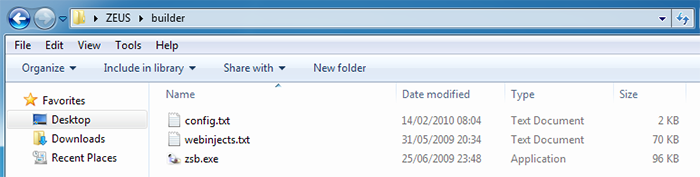
*Figure 8 - ZeuS Server - bot commands*

The C2 server is simply designed, easy to use, and an extremely effective means of operating a malicious network of bots.

ZeuS Bot Builder

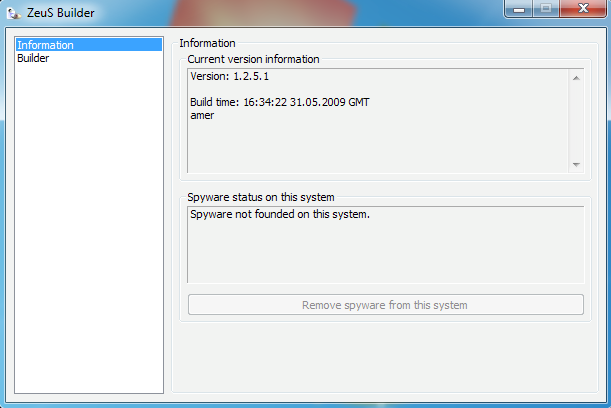
ZeuS bots are built using three primary components:

* The builder application used to build the encrypted config binary and the loader executable.
* The configuration file which defines various bot functionalities and characteristics including:   
  **◦**Botnet name   
  **◦**C2 server, loader, and config URLs   
  **◦**Encryption key   
  **◦**List of targeted banks   
  **◦**Timing info for periodic config updates, logs, and statistics
* The web injects file (discussed in more detail later). This contains specifics on building the web injects which steal credentials by injecting fake forms into web pages visited by the victim:



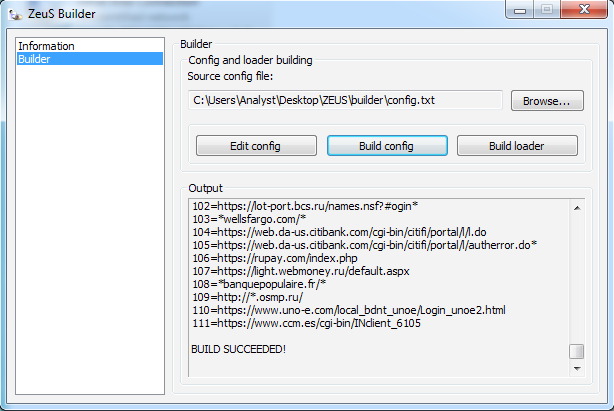
*Figure 9 - ZeuS builder components*

The ZeuS builder has an option to remove any pre-existing spyware, should the local machine be actively infected by ZeuS:



*Figure 10 - ZeuS Builder - spyware removal option*

The bot building process is simple. Users can use a custom config or edit the default config provided by the kit. Once the config file is ready the builder option *Build config* generates a compressed and encrypted binary. The binary will be downloaded and used by the loader component:

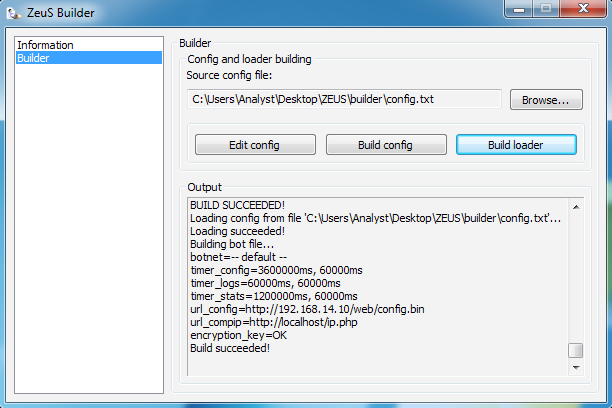


*Figure 11 – ZeuS Builder - building the config*

Building the loader is the final server-side step. Clicking the *Build loader* button will embed various essential information into the loader exe including:

* The URL of the configuration file
* Timing info
* The decryption key for the config file

The builder will attempt to obfuscate the embedded data by packing the loader with a custom packer:



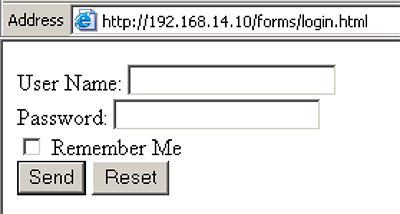
*Figure 12 - Zeus Builder - building the loader*

Keeping the config and loader separate allows threat actors to easily update each component as necessary. The next step is infecting a potential victim by having them download and execute the loader component. This will turn the victim machine into another bot in the threat actor’s botnet. Infection vectors vary, but ZeuS commonly spreads via malspam or drive-by download of the loader from malicious and/or compromised websites.

Web Injects

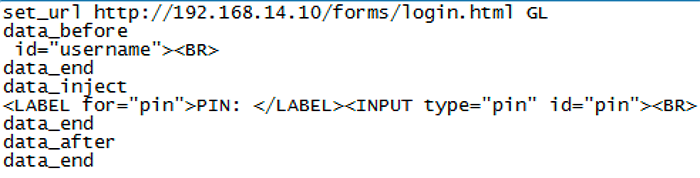
Banking and information stealing Trojans use web injects to trick end users into disclosing more information than required when completing online forms. This is typically done by injecting HTML or JavaScript into the original web page before it is rendered by the user's browser. Web injects allows the threat actor to add content like PIN or credit card entry fields, or remove content like security alerts from the webpage. Unsuspecting users are often tricked by the web injects into providing sensitive information.

The web injects used by ZeuS are contained in the *webinjects.txt* file, provided as part of the bot builder. The injects are easily modified. The simple login form below provides an example:



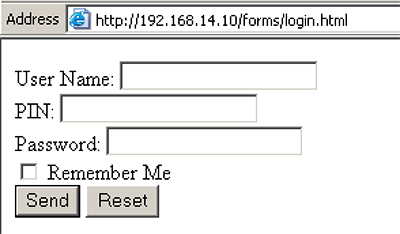
*Figure 13 - ZeuS - original form provided by web server*

The Web inject required to create a field named “PIN” on this form is shown in Figure 14. The URL of the form is provided and the location to inject the new field is specified:



*Figure 14 - ZeuS - basic web inject example*

When the victim browses to the listed URL ZeuS hooks the relevant HTTP Windows APIs and injects the new field into the form. The injected form is then rendered by the user’s browser:



*Figure 15 - ZeuS - injected webpage with new "PIN" field*

Web injects allows ZeuS threat actors to evade many client-side security techniques and steal information from websites of their choosing. Attackers can easily add or modify the web-injects to account for a change in a banking institution’s URL or webpage layout by updating the config. This adaptability ensures attackers maintain maximum threat coverage with minimum downtime.

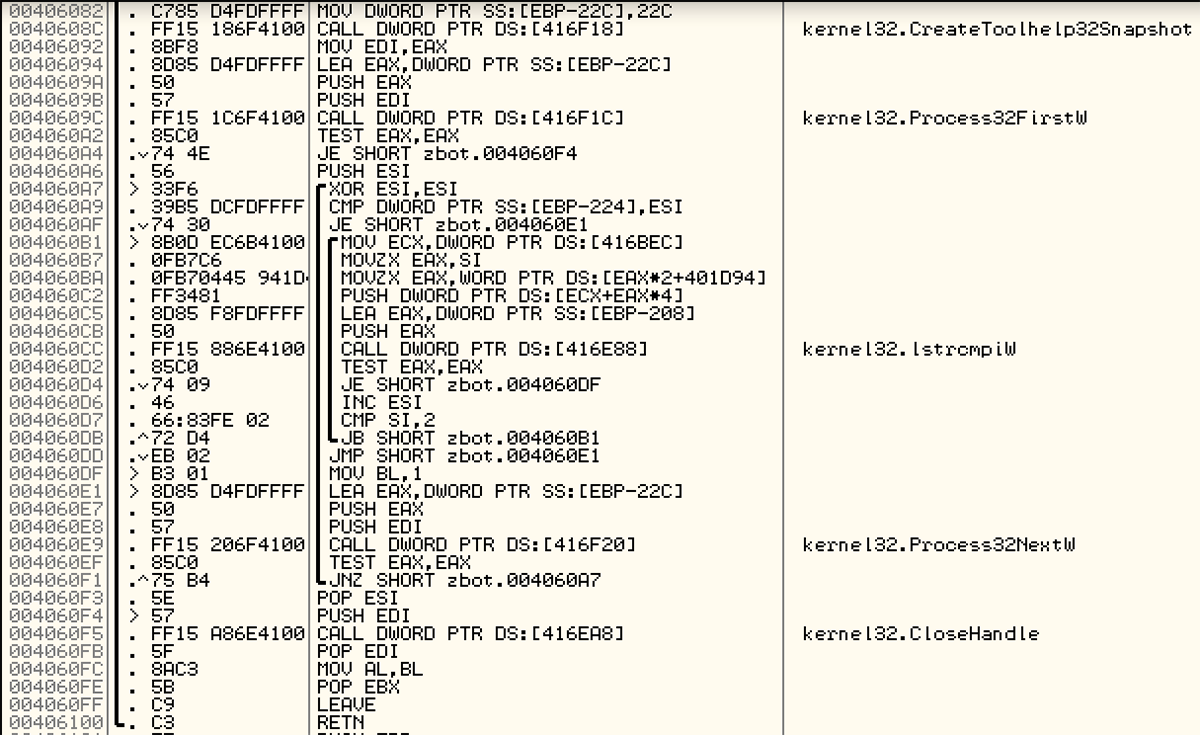
ZeuS Bot

The Zeus bot binary employs a custom packer that obfuscates the samples code and hinders analysis and reverse engineering efforts. The de-obfuscation process goes through a number of stages before Zeus begins its installation. Loading the sample into OllyDbg verifies the custom packing with this warning:



*Figure 16 - ZeuS bot - OllyDbg compressed code warning*

Once de-obfuscated, ZeuS proceeds to dynamically load *LoadLibrary* and *GetProcAddress* from *Kernel32.dll*. It then decrypts several strings which are the Windows APIs it loads with *LoadLibrary* and *GetProcAddress*. Once this process is completed ZeuS is ready to engage in malicious activities. Its first task is to enumerate the processes running on the system:



*Figure 17 - ZeuS bot - process enumeration*

Process enumeration is performed to check for two Firewall applications running on the system, Agnitum Outpost Firewall (*outpost.exe*) and ZoneLabs (now ZoneAlarm) Firewall (*zlclient.exe*). Should either of these products be detected, the bot installation will abort:

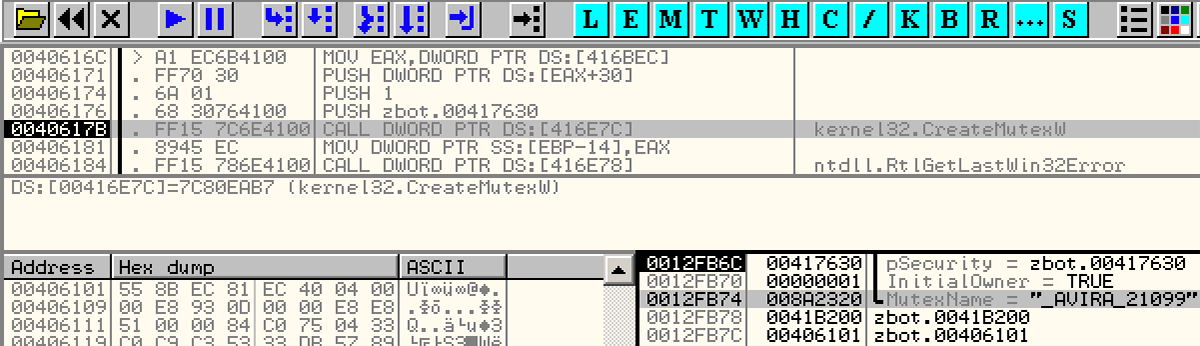


*Figure 18 - ZeuS bot - outpost.exe process check*



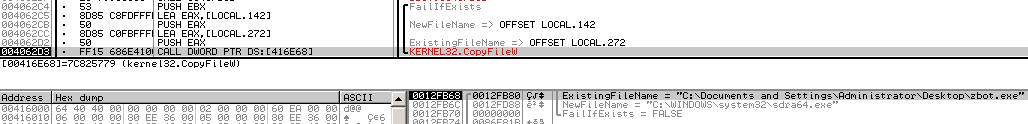
*Figure 19 - Zeus bot - zlclient.exe process check*

Next, a mutex named *\_AVIRA\_2109* is created to ensure only one copy of ZeuS is running on the system:



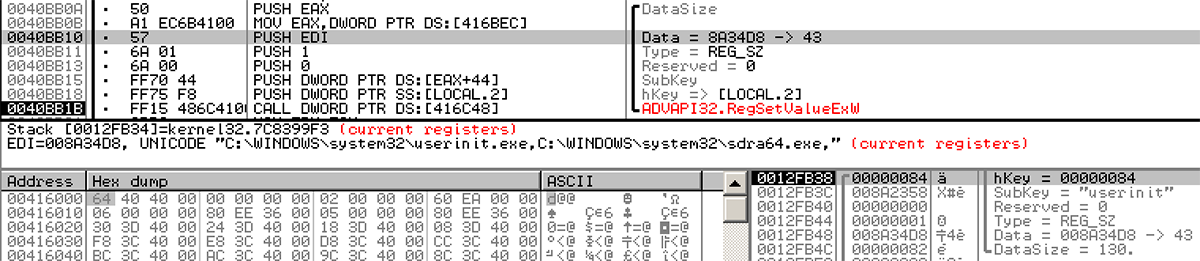
*Figure 20 - ZeuS bot - "\_AVIRA\_2109" mutex creation*

Once the mutex has been created the bot searches for *C:\windows\system32\sdra64.exe*. If found, the file is deleted, as this is the same location and filename it uses on an infected PC. ZeuS then copies itself to *C:\windows\system32\sdra64.exe*. This replacement process provides a means of updating the bot binary:



*Figure 21 - ZeuS - copies itself to the System32 directory as sdra64.exe*

ZeuS modifies the *userinit* subkey of the *HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon* registry entry to maintain persistence. It adds the path to *sdra64.exe* to the existing value of *C:\WINDOW\system32\userinit.exe*:



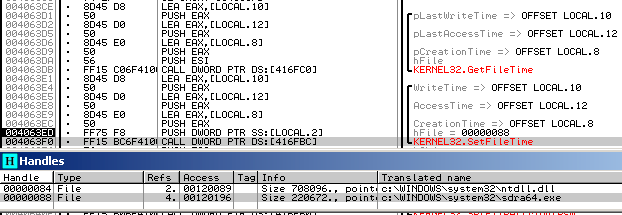
*Figure 22 - ZeuS bot - registry modification for persistence*

ZeuS takes numerous steps to hide itself from both antivirus (AV) and the end user. The first of these steps is to set the attributes of  *sdra64.exe* to **system** and **hidden**. This hides the file from view unless users change their folder viewing options to include **hidden** and **system** files:



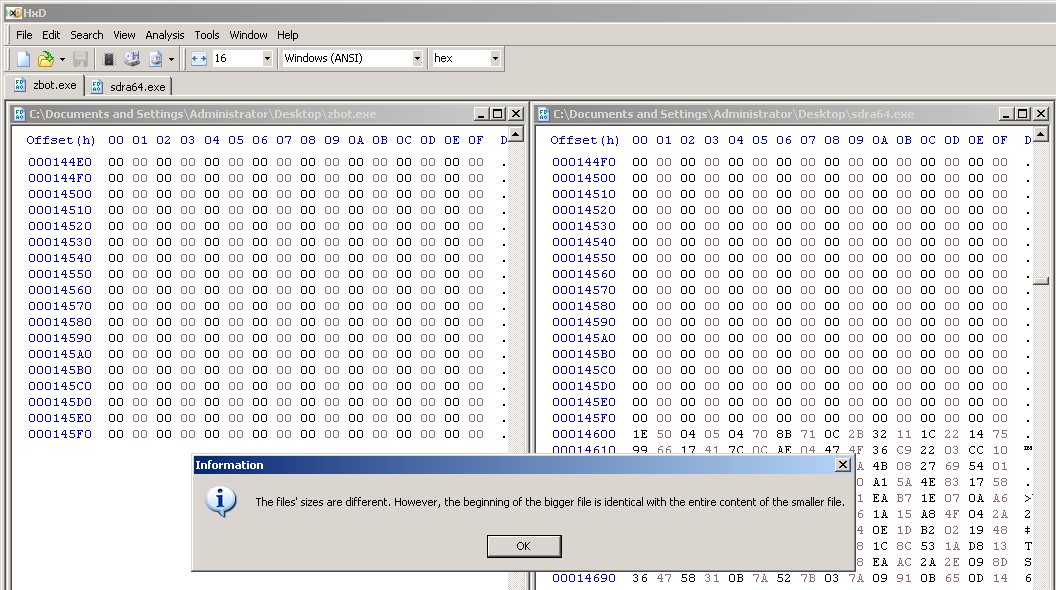
Figure 23 - ZeuS bot - set file attributes of sdra64.exe to **system** and **hidden**

ZeuS mimics the created, modified, and accessed times of *ntdll.dll.*This gives the user the impression that *sdra64.exe* has been present on the system for a long time:



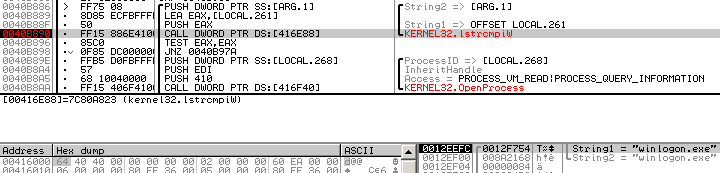
*Figure 24 - ZeuS bot - mimic ntdll.dll MAC times*

ZeuS appends random data to the end of *sdra64.exe* in an attempt to evade signature-based AV:



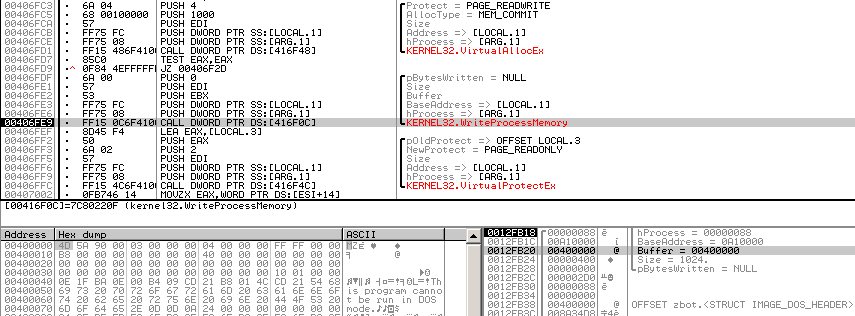
*Figure 25 - Zeus Bot - Appended Data for Signature Based AV Evasion*

The bot enumerates the running processes again to obtain a handle for *winlogon.exe* for process injection:



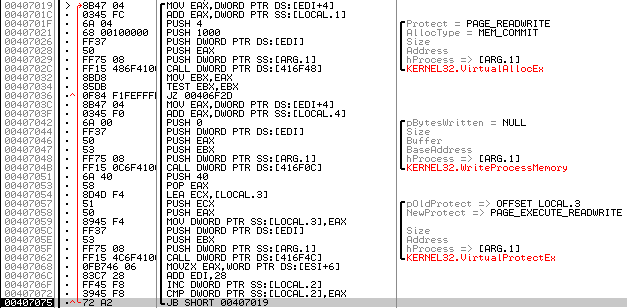
*Figure 26 – ZeuS bot - search for winlogon.exe for process injection*

ZeuS runs some initial checks on the token of *winlogon.exe* to ensure it is running as **system**. It then makes a trio of calls to *VirtualAllocEx*, *WriteProcessMemory*, and *VirtualProtectEx*. These calls  allocate memory in the remote *winlogon.exe* process, write data from the bot’s process address space, and changes the memory protections:



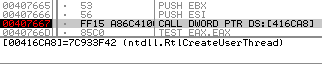
*Figure 27 - ZeuS bot - allocating memory in the remote winlogon.exe process*

It will repeat this process to write all its malicious code into *winlogon.exe* and change virtual protections to *read*, *write*, and *execute*:



*Figure 28 - Zeus bot - process injection loop*

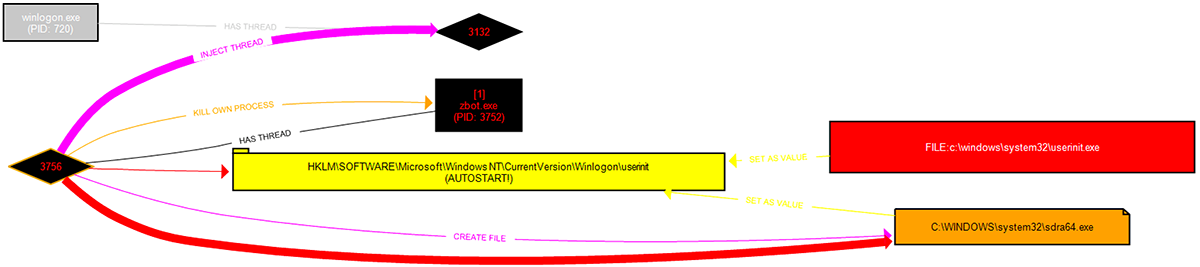
Zeus does not adopt the common method of calling *CreateRemoteThread* followed by *ResumeThread* for injection. Instead, it calls *RtlCreateUserThread* to complete injection into the *winlogon.exe* process:



*Figure 29 - Zeus bot - call to RtlCreateUserThread to execute the injected code*

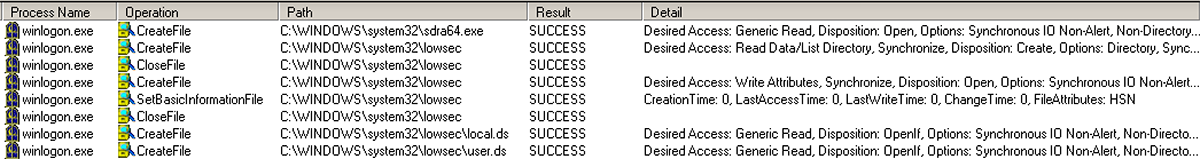
ProcDot gives us a high-level visualization of the steps Zeus has taken so far:

* Copying of the bot to sdra64.exe in the system32 directory
* Modification of the registry for persistence
* Injection into the winlogon.exe process

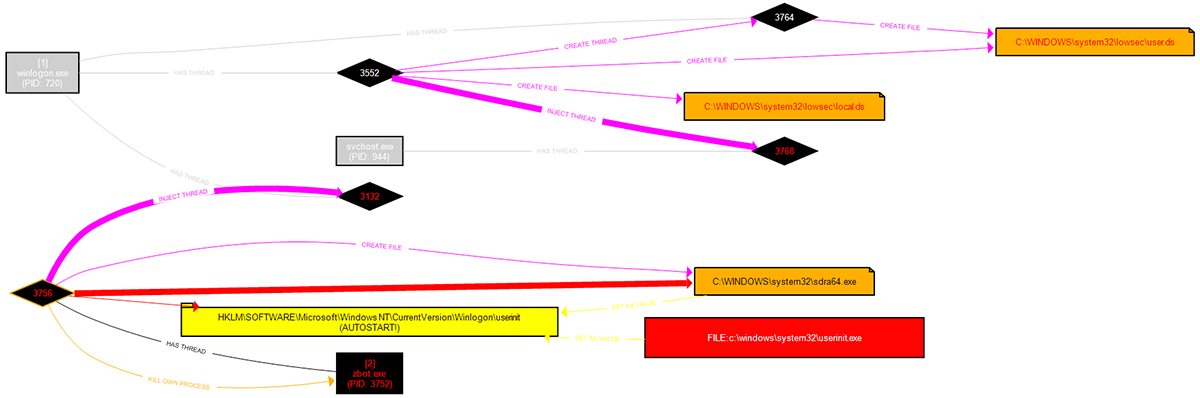


*Figure 30 - Zeus bot – ProcDot visualization of initial installation*

Shortly after ZeuS injects itself into *winlogon.exe* the original process exits leaving the malicious thread to perform the next steps. The infected *winlogon.exe* process checks to see if *sdra64.exe* exists on disk. It then creates a folder called *lowsec* in the *system32* directory. The folder is created with the same **system** and **hidden** attributes used to hide *sdra64.exe*. Two new files are created in the *lowsec* directory named *local.ds* and *user.ds*. *Local.ds* is used to store stolen information and *user.ds* is used to store the bot’s configuration data:

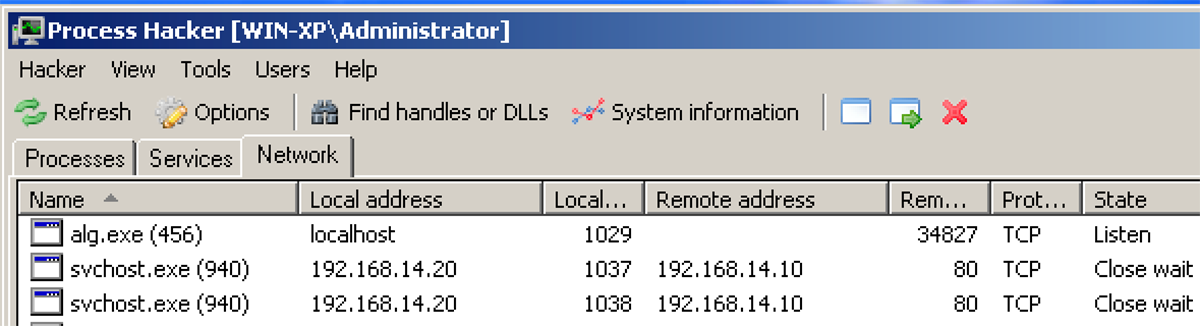


*Figure 31 - ZeuS bot - Winlogon.exe creating lowsec folder,  local.ds, and user.ds files*

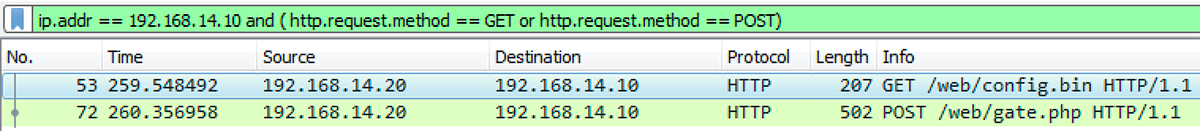
ZeuS then injects itself into *svchost.exe* which will continue the malicious activities. The chain of events described thus far can be easily visualized using ProcDOT:

*Figure 32 - ZeuS bot - ProcDot chain of events from initial infection*

The malicious *svchost.exe* process connects to the C2, downloads the configuration data, and reports a successful infection including details about the infected system:

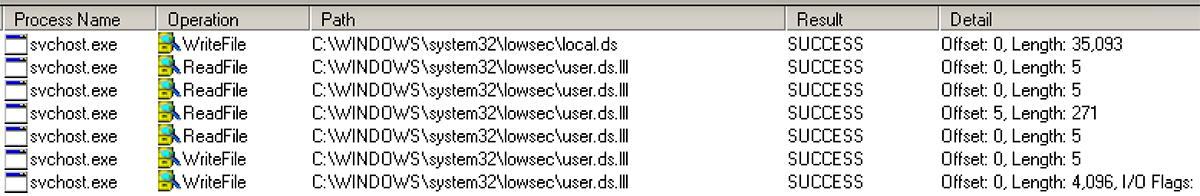


*Figure 33 – ZeuS bot - svchost connection to C2*



*Figure 34 - ZeuS bot - config download and C2 check-in*

Svchost.exe stores the downloaded configuration data in *user.ds*. It logs its reports and the stolen information it will exfiltrate to *local.ds* within the hidden *lowsec* folder:



*Figure 35 - ZeuS bot - injected svchost.exe process interacting with local.ds and user.ds files*

This completes the infection process. The target PC is now an active member of a ZeuS botnet and will execute any script commands sent by the botnets master. The infected processes will perform web injects by hooking the Windows API functions responsible for sending and receiving HTTP(S) data, Unsuspecting users will provide confidential information which Zeus then sends to the configured C2 or drop server.

BlackBerry Stops ZeuS

Threat actors use ZeuS to perform web injects and steal all manner of information from victims. The abundance of ZeuS crimeware kits have made this threat both widely accessible and prolific in the wild. ZeuS attacks have already caused billions of dollars in losses, making it a proven cyber threat worthy of concern.

BlackBerry uses artificial intelligence-based agents trained for threat detection on millions of both safe and unsafe files. This allows BlackBerry to block Zeus based on countless file attributes instead of a specific file signature. BlackBerry, which offers a [predictive advantage](https://threatvector.cylance.com/en_us/home/cylance-vs-future-threats-the-predictive-advantage.html) over zero-day threats, is trained on and effective against both new and legacy cyberattacks.

Appendix

Indicators of Compromise (IOCs)

* **Hashes   
  ◦**04E900A2FBC88AFB6A19606DF7001F0E2D5BA67D458A07BC5B995B79F03EF6C7
* **Filenames   
  ◦** c:\windows\system32\sdra64.exe  
  ◦  c:\windows\system32\lowsec\local.ds  
  ◦  c:\windows\system32\lowsec\user.ds
* **Mutexes**  
  **◦**\_AVIRA\_2109
* **Registry Keys**  
  **◦**SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\userinit