

Zero Wine: Kötü Amaçlı Yazılım Davranış Analizi

İçindekiler

[giriş](#)

[Nasıl çalışır?](#)

[Sanal makineyi QEMU ile çalıştırma](#)

[Raporları yorumlama](#)

[Eksiksiz Rapor](#)

[Teller](#)

[Dosya Başlıkları](#)

[İmza](#)

[Yaygın Sorunlar](#)

[paketleyiciler](#)

[Tespit etme](#)

[tehlikeler](#)

[İndirmek](#)

[Referanslar](#)

[İletişim bilgileri](#)

giriş

Zero wine, kötü amaçlı yazılımın davranışını dinamik olarak analiz etmeye yönelik açık kaynaklı (GPL v2) bir araştırma projesidir. Zero wine, program tarafından çağrılan API'ler hakkında bilgi toplayan güvenli bir sanal sanal alanda (yalıtılmış bir ortamda) WINE kullanarak kötü amaçlı yazılımı çalıştırır.

Şarap tarafından üretilen çıktı (debug ortam değişkeni WINEDEBUG kullanılarak), kötü amaçlı yazılım tarafından kullanılan API çağrılarıdır (ve elbette onun tarafından kullanılan değerler). Bu bilgilerle kötü amaçlı yazılımın davranışını analiz etmek çok kolay hale geliyor.

Nasıl çalışır?

Sıfır şarap, Debian işletim sistemi kurulu bir QEMU sanal makine görüntüsü olarak dağıtılır. Görüntü, kötü amaçlı yazılım yüklemek ve analiz etmek ve toplanan bilgilere dayalı raporlar oluşturmak için yazılım içerir (bu yazılım /home/malware/zerowine

konumunda saklanır).

Dağıtılmış sanal makineyi doğru komut satırı seçenekleriyle çalıştırmak (sanal makineyi çalıştırmak için verilen başlangıç kabuğu komut dosyasını kullanın), analiz edilecek kötü amaçlı yazılım yüklemek için web tabanlı (web sunucusu python ile yazılmıştır) grafik arabirim sağlar (ayrıca yazılı bir CGI). , python'da).

Yeni bir kötü amaçlı yazılım yüklendiğinde, /tmp/vir/MD5_OF_THE_FILE dizinine kopyalanır, bunlar, daha önce oluşturulan WINE ortamı (tercih ederseniz WINEPREFIX) kaldırılır ve bir yedekleme sistemi taranır (yedekleme sistemi /home/malware'dir) /yedekleme/yedekleme.tar.gz). Bu işlemten sonra kötü amaçlı yazılım, malware_launcher.sh kabuk betiği kullanılarak yürütülür (dosya /home/malware/bin klasöründe saklanır).

NOT: Mevcut sistem, aynı anda birden fazla kötü amaçlı yazılımın analizine izin vermediğinden değişikliğe tabidir. Gelecekte, her yeni kötü amaçlı yazılım dosyası yüklediğinizde, daha sonra analiz edilmek üzere bir kuyruğa eklenecek ve bu kötü amaçlı yazılımı çalıştırmak için özel yeni bir WINEPREFIX oluşturulacaktır.

Sanal makineyi QEMU ile çalıştırma

Çeşitli argümanlar vererek QEMU kullanarak sanal makineyi çalıştırmalısınız. En önemlisi şudur: -redir tcp:8000::8000. Bu parametre, yerel 8000 bağlantı noktasını sanal makinenin 8000 bağlantı noktasına yönlendirir. Elbette, bağlantı noktasını değiştirebilirsiniz, ancak Unix/Linux tabanlı ortamlarda 80 numaralı bağlantı noktasında **çalıştırmamanız gerektiğini unutmayın** , çünkü kök ayrıcalıklarına ihtiyaç duyar, büyük bir güvenlik açığı açar (kötü amaçlı yazılımın sanal makineden kaçtığını ve sahip olduğunu hayal edin. gerçek sistem).

Sanal makine önyükleme işlemini bitirdiğinde (Debian tabanlı işletim sisteminin önyükleme yapması için yaklaşık 2 dakika kadar bekleyin) tercih ettiğiniz tarayıcıyla <http://localhost:8000> adresine gidebilirsiniz . Aşağıdaki çok basit web sayfasıyla karşılaşacaksınız:

Zero Wine: A Malware Analysis Tool

Select the malware file to upload and the options to test it:



Malware file

Seleccionar archivo ningún archivo cargado

Timeout

5

Restaurar

Enviar

Copyright (c) 2008 Joxean Koret

Test (PE) dosyanızı web arayüzü üzerinden sanal makineye yükleyin, zaman aşımını belirtin ve programın davranışını sıfır şarabın analiz etmesine izin verin. Bir süre sonra aşağıdaki gibi bir raporun özeti oluşacaktır:

Malware analysis

Analyzing file: **document.exe.**

MD5 Sum: **bc3dedd6c1b968d295a484229d504a15**



Warning: Folder already exists! File was previously analyzed?

File saved as: **bc3dedd6c1b968d295a484229d504a15/document.exe**



[Report](#)



[Strings](#)



[File headers](#)



[Signature](#)

Analysis finished at Fri Dec 19 13:25:29 2008

Raporları yorumlama

Analiz bittiğinde, (bu küçük makaleyi yazarken) 4 bağlantı içeren bir raporun özet sayfası görünür. Bağlantılar "Rapor", "Dizeler", "Başlıklar" ve "İmza" dır. Seçenekler sonraki bölümlerde açıklanmaktadır.

Eksiksiz Rapor

İlk bağlantı Rapor bağlantısıdır. Bu seçenek, WINE tarafından oluşturulan tam ham izleme dosyasını gösterir. Bu dosya normalde çok büyük ve takip edilmesi zor (WINE tarafından kötü amaçlı yazılım tarafından çağrılan API'lerle karıştırılmış çok fazla API var), ancak programın ne yaptığını tam olarak anlamınıza yardımcı olabilir.

The following is a sample report for the virus MyTob (as you might see it's very long and, as so, hard to understand).

Malware analysis

Analyzing file: **document.exe**.

MD5 Sum: bc3dedd6c1b968d295a484229d504a15



Warning: Folder already exists! File was previously analyzed?

File saved as: **bc3dedd6c1b968d295a484229d504a15/document.exe**



Report



Strings



File headers



Signature

```
0009:Starting process L"C:\bc3dedd6c1b968d295a484229d504a15\document.exe" (entryproc=0x4309ec)
0009:Call KERNEL32.LoadLibraryA(00423765 "WSOCK32.dll") ret=004001e1
trace:file:ReadFile 0x2c 0x7ec4395c 16 0x7ec4391c (nil)
trace:ntdll:NtReadFile (0x2c,(nil),(nil),0x7ec43838,0x7ec4395c,0x00000010,(nil),(nil)),partial stub!
trace:file:RtlDosPathNameToNtPathName_U (L"C:\bc3dedd6c1b968d295a484229d504a15\WSOCK32.dll",0x32f910,(nil),(nil))
trace:file:RtlGetFullPathName_U (L"C:\bc3dedd6c1b968d295a484229d504a15\WSOCK32.dll" 520 0x32f694 (nil))
trace:file:RtlDosPathNameToNtPathName_U (L".\WSOCK32.dll",0x32f910,(nil),(nil))
trace:file:RtlGetFullPathName_U (L".\WSOCK32.dll" 520 0x32f694 (nil))
trace:file:RtlDosPathNameToNtPathName_U (L"C:\windows\system32\WSOCK32.dll",0x32f910,(nil),(nil))
trace:file:RtlGetFullPathName_U (L"C:\windows\system32\WSOCK32.dll" 520 0x32f694 (nil))
trace:file:wine_nt_to_unix_file_name L"\\?\C:\windows\system32\WSOCK32.dll" ->
"/home/malware/.wine/dosdevices/c:/windows/system32/wsock32.dll"
trace:file:RtlGetFullPathName_U (L"C:\windows\system32\WSOCK32.dll" 64 0x32fac0 0x32fa28)
trace:file:RtlDosPathNameToNtPathName_U (L"C:\windows\system32\WSOCK32.dll",0x32fa14,(nil),(nil))
trace:file:RtlGetFullPathName_U (L"C:\windows\system32\WSOCK32.dll" 520 0x32f754 (nil))
trace:ntdll:NtCreateFile handle=0x32fb14 access=80000000 name=L"\\?\C:\windows\system32\WSOCK32.dll"
objattr=00000040 root=(nil) sec=(nil) io=0x32fa1c alloc_size=(nil)
attr=00000000 sharing=00000005 disp=1 options=00000000 ea=(nil).0x00000000
trace:file:wine_nt_to_unix_file_name L"\\?\C:\windows\system32\WSOCK32.dll" ->
"/home/malware/.wine/dosdevices/c:/windows/system32/wsock32.dll"
trace:ntdll:NtOpenThreadToken (0xffffffff,0x00020008,0x00000001,0x32f8f4)
trace:ntdll:NtOpenProcessToken (0xffffffff,0x00020008,0x32f8f4)
trace:ntdll:NtQueryInformationToken (0x24,1,0x32f8f8,80,0x32f8f0)
trace:ntdll:NtReadFile (0x1c,(nil),(nil),0x32fb0c,0x32fb1f,0x00000055,0x32fb00,(nil)),partial stub!
trace:ntdll:NtWriteFile = SUCCESS (72)
trace:file:ReadFile 0x2c 0x7ec154bc 16 0x7ec1547c (nil)
trace:ntdll:NtReadFile (0x2c,(nil),(nil),0x7ec15398,0x7ec154bc,0x00000010,(nil),(nil)),partial stub!
trace:ntdll:NtReadFile = SUCCESS (16)
trace:file:ReadFile 0x2c 0x11e388 8 0x7ec1547c (nil)
trace:ntdll:NtReadFile (0x2c,(nil),(nil),0x7ec15398,0x11e388,0x00000008,(nil),(nil)),partial stub!
```

Too many API calls (generally, internal WINE calls) that are just junk calls, uninteresting in general for us. Anyway, remember that they might help you understanding the malware you're analyzing even when various calls appears to be uninteresting.

Strings

The output of the Linux command “strings”. Sometimes (in ~~stupid~~ easy to understand malware) you might discover interesting strings, URLs, and so on... The following is an example (MyTob):

Malware analysis

Analyzing file: **document.exe**.

MD5 Sum: bc3dedd6c1b968d295a484229d504a15



Warning: Folder already exists! File was previously analyzed?

File saved as: **bc3dedd6c1b968d295a484229d504a15/document.exe**



Report



Strings



File headers



Signature

```
wwwwww
kernel32.dll
LoadLibraryA
GetProcAddress
;33;33;0
wwwwww
kernel32.dll
LoadLibraryA
GetProcAddress
;33;33;0
wwwwww
kernel32.dll
LoadLibraryA
GetProcAddress
;33;33;0
wwwwww
kernel32.dll
LoadLibraryA
GetProcAddress
;33;33;0
wwwwww
kernel32.dll
LoadLibraryA
GetProcAddress
;33;33;0
wwwwww
kernel32.dll
LoadLibraryA
GetProcAddress
;33;33;0
```


File Headers

This report shows the tool used to pack the program if any (using the signatures from PEIdSignatures) and the full output of analyzing the given PE program with the open source library PEFile. The report generated will be similar to the following image:

Malware analysis


Analyzing file: **document.exe**.


MD5 Sum: **bc3dedd6c1b968d295a484229d504a15**


 **Warning:** Folder already exists! File was previously analyzed?

File saved as: **bc3dedd6c1b968d295a484229d504a15/document.exe**

 **Report**

 **Strings**

 **File headers**

 **Signature**

```
-----Signature-----  
Mew 11 SE v1.2 (Eng) -> Northfox  
MEW 11 SE v1.2 -> Northfox[HCC]  
  
-----Parsing Warnings-----  
Corrupt header "IMAGE_IMPORT_DESCRIPTOR" at file offset 52229. Exception: 'Data length less than expected header length.'  
  
-----DOS_HEADER-----  
[IMAGE_DOS_HEADER]  
e_magic:      0x5A4D  
e_cblp:        0x0  
e_cp:          0x0  
e_crlc:        0x0  
e_cparhdr:     0x0  
e_minalloc:    0x0  
e_maxalloc:    0x4550  
e_ss:          0x0  
e_sp:          0x14C  
e_csum:        0x2  
e_ip:          0x0  
e_cs:          0x0  
e_lfarlc:      0x0  
e_ovno:        0x0  
e_res:         à  
e_oemid:       0x10B  
e_oeminfo:     0xC05
```

In this example the header seems to be corrupted, a signal of malware trying to make disassembling harder.

Signature

This is the most interesting report generated by zero wine (when the malware runs OK). This report shows the most interesting API calls and the values used. Easier to see in the following example:

Malware analysis

Analyzing file: **document.exe**.

MD5 Sum: bc3dedd6c1b968d295a484229d504a15



Warning: Folder already exists! File was previously analyzed?

File saved as: **bc3dedd6c1b968d295a484229d504a15/document.exe**



Report



Strings



File headers



Signature

```
0009:Call KERNEL32.CreateMutexA(00000000,00000000,004141f0 "H-E-L-L-B-O-T") ret=00408cac
0009:Call KERNEL32.CopyFileA(0032f150 "C:\\bc3dedd6c1b968d295a484229d504a15\\document.exe",0032f250
"C:\\windows\\system32\\msmgrxp.exe",00000000) ret=00408dc5
trace:file:CopyFileW L"C:\\bc3dedd6c1b968d295a484229d504a15\\document.exe" ->
L"C:\\windows\\system32\\msmgrxp.exe"
trace:file:CreateFileW L"C:\\bc3dedd6c1b968d295a484229d504a15\\document.exe" GENERIC_READ FILE_SHARE_READ
FILE_SHARE_WRITE creation 3 attributes 0x0
trace:file:CreateFileW L"C:\\windows\\system32\\msmgrxp.exe" GENERIC_WRITE FILE_SHARE_READ FILE_SHARE_WRITE creation 2
attributes 0x20
0009:Call KERNEL32.CreateProcessA(00000000,0032f250
"C:\\windows\\system32\\msmgrxp.exe",00000000,00000000,00000001,00000028,00000000,00000000,0032f0cc,0032f0bc)
ret=00408e77
0018:Call KERNEL32.CreateMutexA(00000000,00000000,004141f0 "H-E-L-L-B-O-T") ret=00408cac
0018:Call KERNEL32.CopyFileA(0033f150 "C:\\windows\\system32\\msmgrxp.exe",004141d8 "C:\\funny_pic.scr",00000000)
ret=00408e9a
trace:file:CopyFileW L"C:\\windows\\system32\\msmgrxp.exe" -> L"C:\\funny_pic.scr"
trace:file:CreateFileW L"C:\\windows\\system32\\msmgrxp.exe" GENERIC_READ FILE_SHARE_READ FILE_SHARE_WRITE creation 3
attributes 0x0
trace:file:CreateFileW L"C:\\funny_pic.scr" GENERIC_WRITE FILE_SHARE_READ FILE_SHARE_WRITE creation 2 attributes 0x20
```

The "Signature" report tells us that the malware created the mutex "H-E-L-L-B-O-T" (process id 0009) and copied itself to c:\windows\system32\msmgrxp.exe. Next, the copied file is executed (process 0018). This copy of the malware checks for the mutex "H-E-L-L-B-O-T" and, if it already exists, copies the binary to c:\funny_pic.scr and to other various places.

Enough information to write a simple behavior report.

Common Problems

Packers

Zero wine runs malware quite well overall, however, it has problems with various packers (in example, wine fails almost always with PE programs packed with Armadillo) and sometimes you will get no data for both "Report" and "Signature" sections. Anyway, the "Headers" and "Strings" report's sections will appear giving you interesting information about the binary (although not the behavior of the malware).

Detection

Detection of the WINE environment demonstrated to be extremely easy. In example, the registry key HKLM\Software\Wine or HKCU\Software\Wine can be opened to detect it. Another example: Check the file size of any Windows critical system file. When running under WINE, the files will be ridiculously small, while in a real Windows system it will have a (always) bigger size.

Another "advanced" detection technique: Open any critical Windows file and decompile the entry point. When running under WINE the function will decompile to the following 2 simple instructions:

```
.text:10001000                public start
.text:10001000 start          proc near
.text:10001000                mov     eax, 1
```

```
.text:10001005          retn     4  
.text:10001005 start     endp
```

For the lazy people: Just search for the following binary string B8 01 00 00 00 C2 04 00 at .text:10001000.

Dangers

First of all, remember, **RUNNING MALWARE IN YOUR COMPUTER IS ALWAYS A BAD IDEA**. You must isolate the virtual machine (or the real hardware box) from the real world when possible. Sometimes it's mandatory to allow the malware to connect to the real world, however, you might be attacking other people.

And, what is more important, remember, a malware written to escape the WINE sandbox will escape (even when the WINE's sandbox is configured with only the C: drive) and it might affect your network, your real machines.

Download

Download zerowine (as a prebuilt virtual machine for QEmu or the python source code) from [here](#).

References

Project's web page: <http://sourceforge.net/projects/zerowine>

ŞARAP: <http://www.winehq.org>

QEMU: <http://bellard.org/qemu/>

Ero Carrera'dan PEFile: <http://code.google.com/p/pefile/>

PEIdSignatures: <http://code.google.com/p/pefile/wiki/PEIdSignatures>

PEID: <http://www.PEiD.info/>

Python: <http://www.python.org>

İletişim bilgileri

Yazar: Joxean Koret

E-Posta: <admin@joxeankoret.com>

<joxeankoret@yahoo.es>

Profesyonel Web Sitesi: <http://www.joxeankoret.com>