

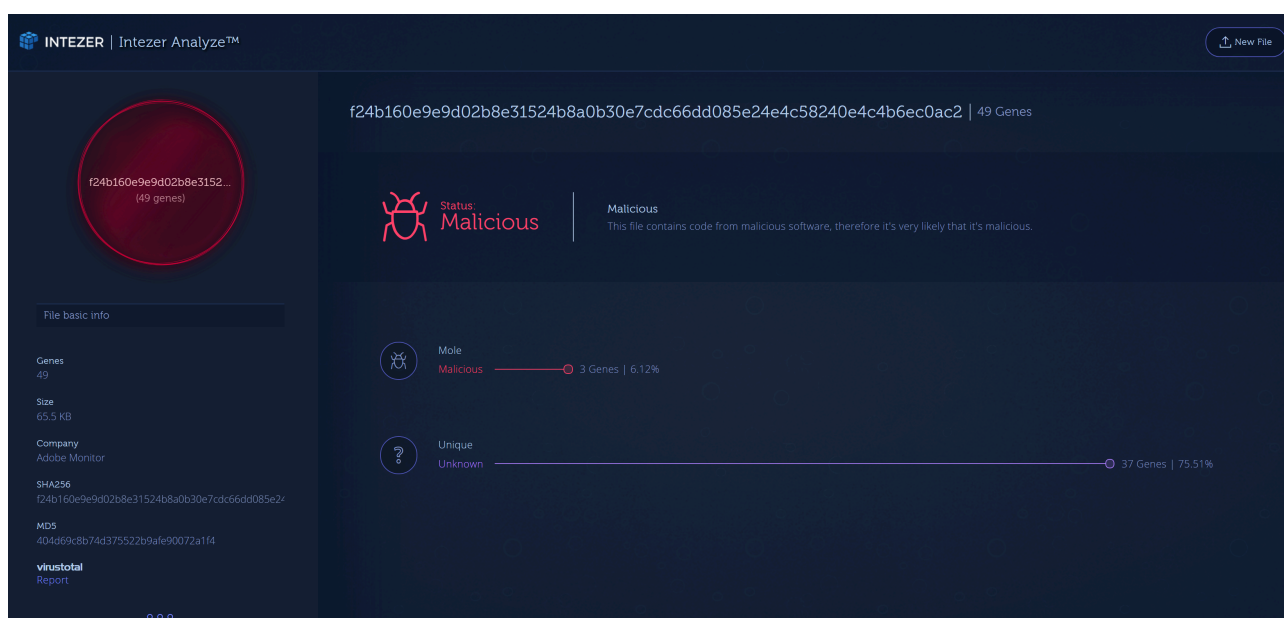
Silence of the Moles

By Jay Rosenberg

Published: 2017-11-01 · Archived: 2026-04-05 14:41:17 UTC

Kaspersky Labs published a technical analysis of a new malware, [Silence](#) that is aimed at attacking financial institutions. After uploading the loader of this malware to [Intezer Analyze™](#), we have found a possible connection through code reuse to the loader of another campaign of malware, [Mole](#) previously discovered by Unit 42 of Palo Alto Networks.

This connection might be an indicator that these two attacks are originated from the same threat actor, but currently it is too early to tell.



([Intezer Analyze™](#) public report available [here.](#))

Silence Loader: f24b160e9e9d02b8e31524b8a0b30e7cdc66dd085e24e4c58240e4c4b6ec0ac2

Mole Loader: 50117ce3fe5dba572cf23584dc7541a7cfd4026d4316e69d29cdf536873fdf20

If we look at the code of the two loaders used by both campaigns side by side, we can see that the code is very similar and according to our system is unique to these families of malware.

```
call ds:GetProcAddress
mov [ebp+var_30], eax
mov [ebp+var_2C], 1000h
```

Silence

```
loc_4079F3:
push 4
push 2000h
lea edx, [ebp+var_2C]
push edx
push 0
lea eax, [ebp+var_20]
push eax
push 0FFFFFFFh
call [ebp+var_30]
mov [ebp+var_1C], eax
cmp [ebp+var_1C], 0
jz short loc_4079F3
```

```
mov ecx, [ebp+var_1C]
shl ecx, 2
mov [ebp+var_1C], ecx
mov [ebp+ProcName], 'R'
mov [ebp+var_17], 't'
mov [ebp+var_16], 'p'
mov [ebp+var_15], 'E'
mov [ebp+var_14], 'n'
mov [ebp+var_13], 'c'
mov [ebp+var_12], 'o'
mov [ebp+var_11], 'd'
mov [ebp+var_10], 'e'
mov [ebp+var_F], 'P'
mov [ebp+var_E], 'o'
mov [ebp+var_D], 'i'
mov [ebp+var_C], 'n'
mov [ebp+var_B], 't'
mov [ebp+var_A], 'e'
mov [ebp+var_9], 'r'
mov [ebp+var_8], 0
mov edx, [ebp+var_1C]
add edx, 0Ch
mov eax, 1
shl eax, 1
mov [ebp+eax+ProcName], dl
lea ecx, [ebp+ProcName]
push ecx ; lpProcName
push offset aNtd11_0 ; "ntd11"
call ds:GetModuleHandleA
push eax ; hModule
call ds:GetProcAddress
mov [ebp+var_24], eax
push 25h
call [ebp+var_24]
mov [ebp+var_34], eax
mov edx, [ebp+var_1C]
add edx, 0Ch
mov eax, 1
shl eax, 1
mov [ebp+eax+ProcName], dl
push offset aRtlDecodepoint ; "RtlDecodePointer"
push offset aNtd11_1 ; "ntd11"
call ds:GetModuleHandleA
push eax ; hModule
call ds:GetProcAddress
mov [ebp+var_28], eax
mov ecx, [ebp+var_34]
push ecx
call [ebp+var_28]
mov [ebp+var_1C], eax
mov eax, [ebp+var_1C]
lea eax, [eax+407AAEh]
jmp eax
```

```
call ds:GetProcAddress
mov [ebp+var_48], eax
mov [ebp+var_20], 1000h
```

Mole

```
loc_4023F0:
push 4
push 2000h
lea eax, [ebp+var_20]
push eax
push 0
lea eax, [ebp+var_24]
push eax
push 0FFFFFFFh
call [ebp+var_48]
mov [ebp+var_1C], eax
cmp [ebp+var_1C], 0
jz short loc_4023F0
```

```
mov eax, [ebp+var_1C]
shl eax, 2
mov [ebp+var_1C], eax
mov [ebp+ProcName], 'R'
mov [ebp+var_17], 't'
mov [ebp+var_16], 'p'
mov [ebp+var_15], 'E'
mov [ebp+var_14], 'n'
mov [ebp+var_13], 'c'
mov [ebp+var_12], 'o'
mov [ebp+var_11], 'd'
mov [ebp+var_10], 'e'
mov [ebp+var_F], 'P'
mov [ebp+var_E], 'o'
mov [ebp+var_D], 'i'
mov [ebp+var_C], 'n'
mov [ebp+var_B], 't'
mov [ebp+var_A], 'e'
mov [ebp+var_9], 'r'
mov [ebp+var_8], 0
mov eax, [ebp+var_1C]
add eax, 0Ch
xor ecx, ecx
inc ecx
shl ecx, 1
mov [ebp+ecx+ProcName], al
lea eax, [ebp+ProcName]
push eax ; lpProcName
push offset aNtd11_0 ; "ntd11"
call ds:GetModuleHandleA
push eax ; hModule
call ds:GetProcAddress
mov [ebp+var_44], eax
push 25h
call [ebp+var_44]
mov [ebp+var_50], eax
mov eax, [ebp+var_1C]
add eax, 0Ch
xor ecx, ecx
inc ecx
shl ecx, 1
mov [ebp+ecx+ProcName], al
push offset aRtlDecodepoint ; "RtlDecodePointer"
push offset aNtd11_1 ; "ntd11"
call ds:GetModuleHandleA
push eax ; hModule
call ds:GetProcAddress
mov [ebp+var_4C], eax
push [ebp+var_50]
call [ebp+var_4C]
mov [ebp+var_1C], eax
mov eax, [ebp+var_1C]
lea eax, [eax+4024A6h]
jmp eax
```

(sub_4079A0 vs sub_4023A0)

Through the disassembly in the photo above, we can also see there is a string initialized through an array, “RtpEncodePointer,” that is later used for a call to GetProcAddress. This looks like a typo and the author of the code meant to write “RtlEncodePointer” because “RtpEncodePointer” does not exist in ntdll.dll. The evidence suggests that this code was being reused. There are no references to “RtpEncodePointer” available publicly online besides automated reports of a couple unclassified malware from [Hybrid Analysis](#).

In addition to the links within the code, there are several other similarities we have witnessed between the Mole and Silence malware, such as the attack vectors (spear phishing, packaging of the malware) and motives — which can be extra evidence for this connection.

Yet again, we see that identifying code reuse can be very valuable in detecting new malware, and in some cases for attribution purposes. We invite you to read more of the posts in our [blog](#) and to request an [invite](#) to the community edition of our product.

Follow [@jaytezer](#) for more updates.

Source: <http://www.intezer.com/silenceofthemoles/>