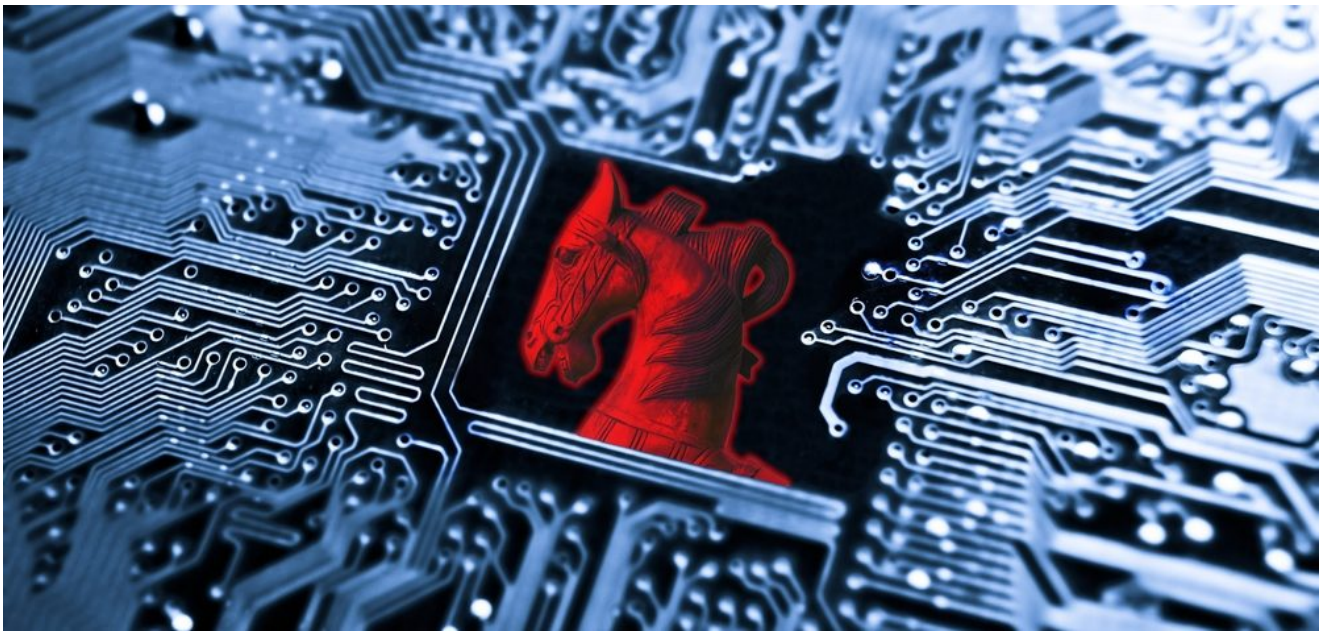
Mitigating Emotet, The Most Common Banking Trojan

intezer.com/mitigating-emotet-the-most-common-banking-trojan/

July 26, 2018



Get Free Account

Join Now

Recently, Proofpoint released a fairly surprising report, stating that Banking Trojans have surpassed Ransomware as the top malware threat found in email. This is not too surprising, due to the rising difficulty of cashing out cyber-ransom operations, and the increasing awareness of enterprises for these kinds of threats. In addition, Emotet created recent headlines with the US-CERT alert about it.

Banking Trojans usually work by logging keystrokes of unsuspected users while they log into their online bank accounts, thus stealing their credentials to their online account. Infamous Banking Trojans include Ursnif, Dridex, Zeus and more.

Most noticable, is a malware named Emotet, which according to Proofpoint's recent report, consists of 57% of all Banking Trojans. At Intezer, we have also noticed a large amount of Emotet samples uploaded to our platform daily, both by our customers and our free community users.

To help organizations and individuals cope with this threat, I decided to post an informative blog post on how we can assist with our technology, in the battle against Emotet.

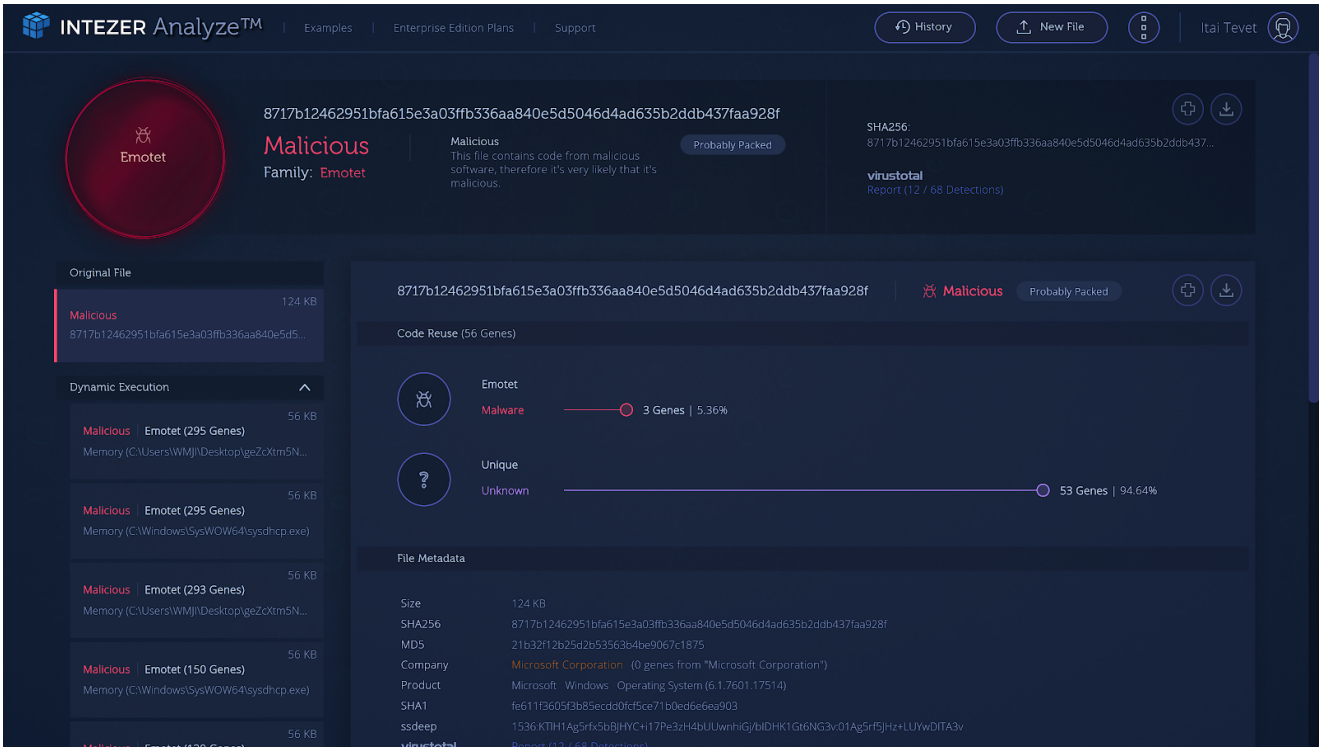
Emotet's Components and Similarities

As you might know, at Intezer, we like to approach the malware problem by identifying code reuse and similarities. In the case of Emotet (and other Banking Trojans), this approach proves itself to be very effective.

Emotet is usually composed from several different components: The initial dropper/loader, and the actual payload (which in many cases are several different ones). The initial dropper is designed to be very stealthy by constantly mutating. It's rare to see two different Emotet loaders that are exactly the same, due to obfuscation or other code mutation techniques. This is extremely effective in order to deal with signature-based defense.

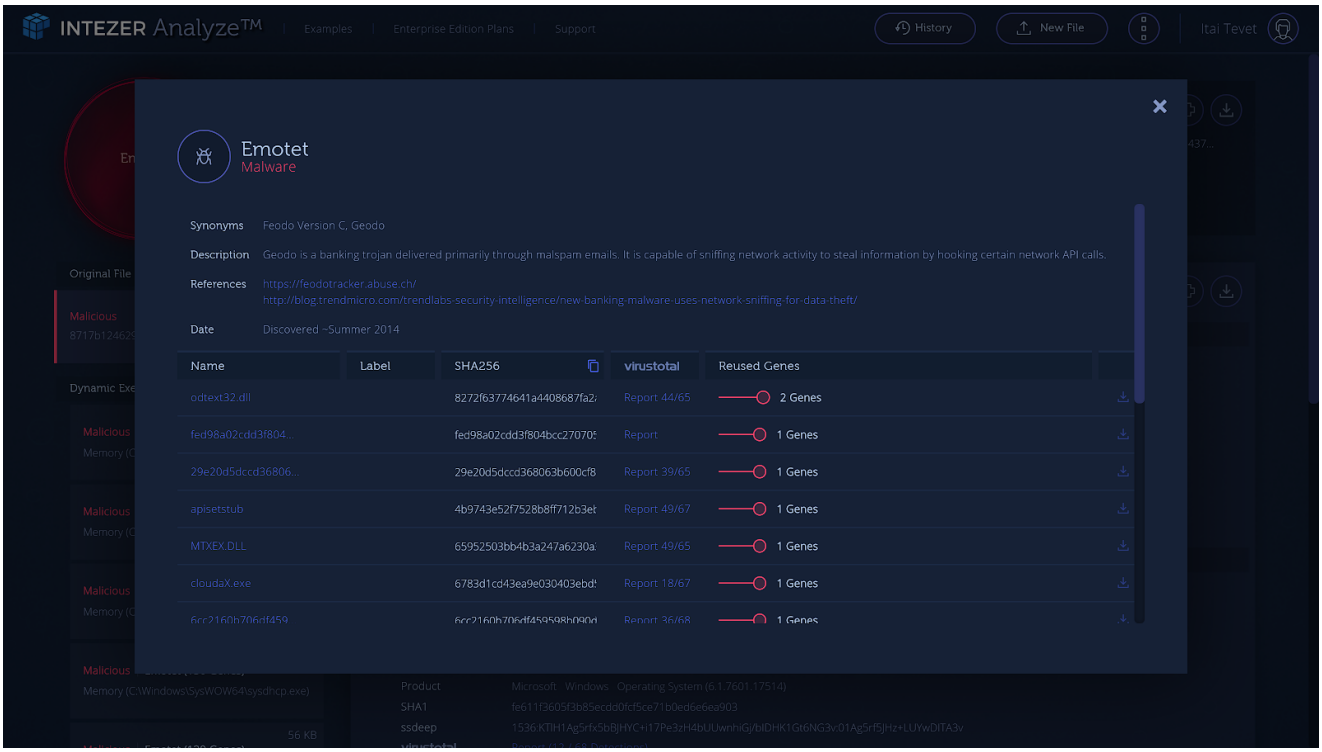
The below report shows a code similarity analysis on the Emotet dropper. It is clear to see that almost 95% of the code is unique, which illustrates the mutation and evasiveness of this module. However, we can also see 3 "genes", or 5% of the file's code that was already seen in other Emotet variants. We can also observe that the malware was detected only by 12 anti-malware engines at the time of the first analysis in our system, which demonstrates the effectiveness of such evasion techniques.

Also, notice the file's metadata: it tries to mimic the Microsoft Windows product details, so that the file would seem legitimate to the untrained user — although there's absolutely no Microsoft code in this binary.



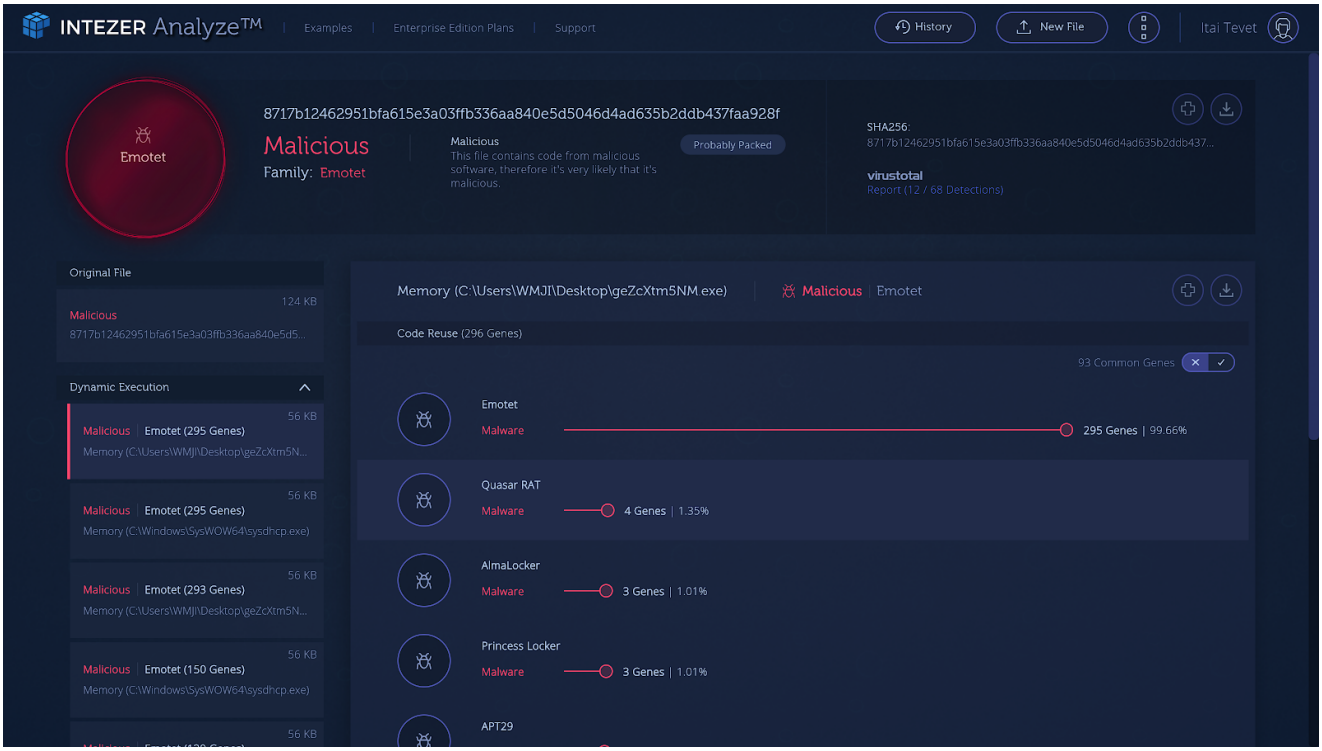
https://analyze.intezer.com/#/analyses/a2e33ff7-6439-453d-885c-25305d433cc9

Digging into the 3 gene connection, we can see exactly which other Emotet variants share some of these genes:



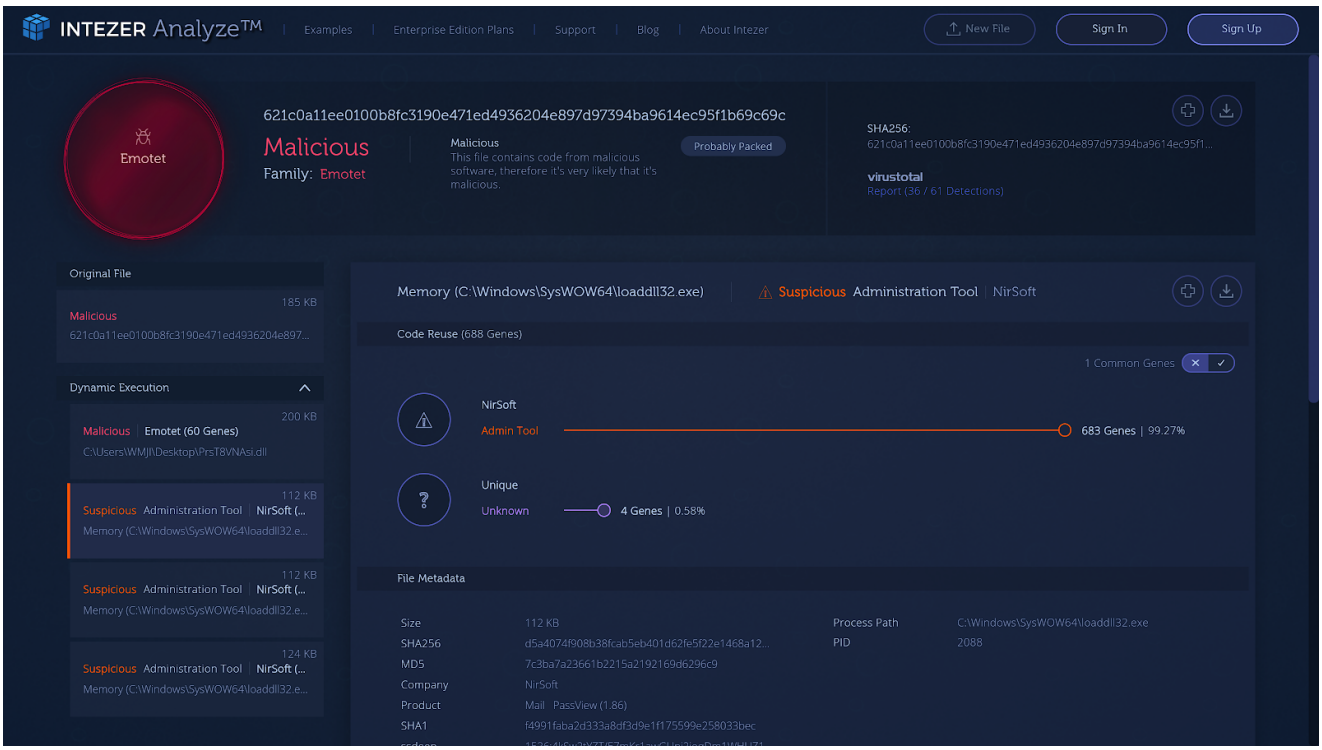
In almost all Emotet cases, we noticed that although the dropper tries to mutate and evade signature-based defenses, we find a small amount of similarity (usually 2-5 “Genes”). It’s a very small connection, but it’s always there.

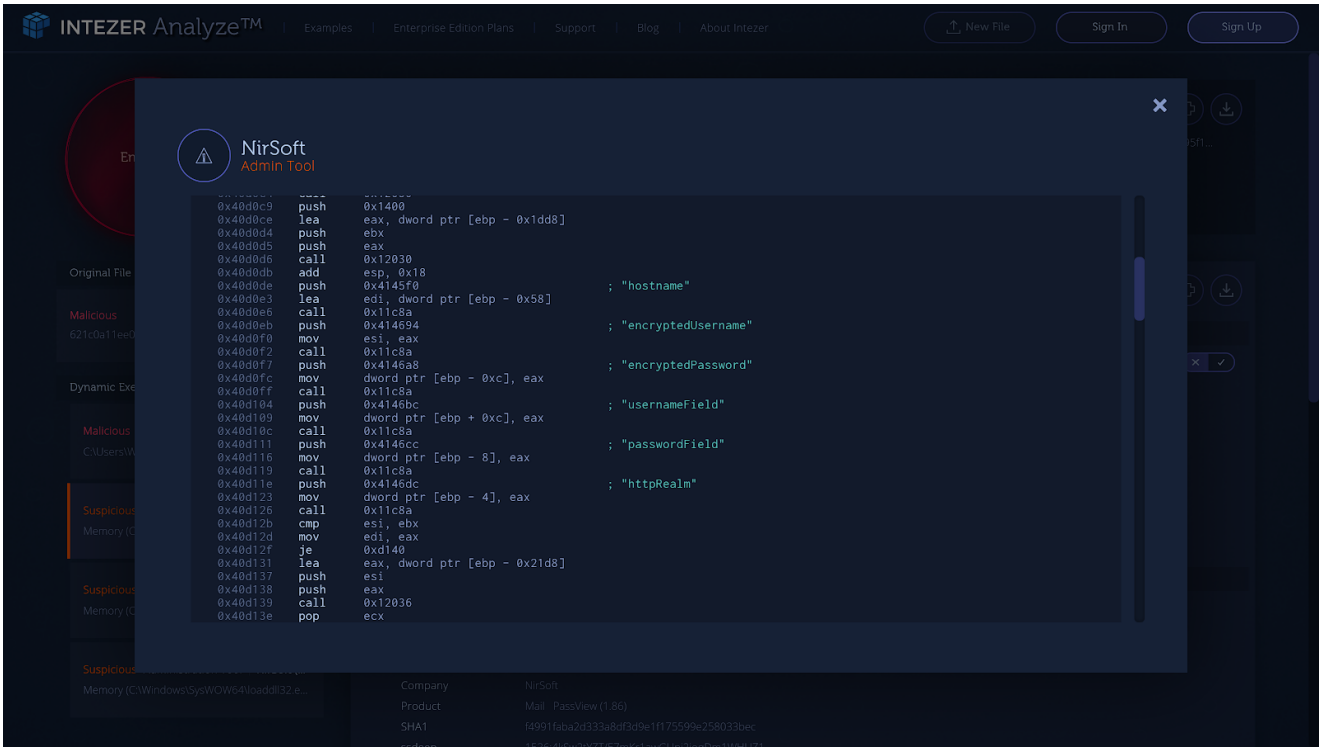
The dropper’s goal is to execute the final payload in memory, where the attacker has much more “freedom” to stay hidden. Indeed, after running the dropper, we see modules in memory that contain much more code (see 56 genes of the dropper vs 296 genes in one of the payloads), and also have a much stronger code similarity to other Emotet variants.



This is a pattern we see time and time again with Emotet: the initial payload is mutated/obfuscated and contains a small connection to Emotet, whereas the final payloads in memory are “unpacked” and contain code that correlates clearly to other variants.

Another interesting example of code similarities within Emotet, is the sample mentioned in Check Point’s blog post where Emotet was embedding the free software Nirsoft in order to steal email credentials. Indeed after execution of the initial loader, we can see a module in memory that shares code with Nirsoft Mail PassView:





https://analyze.intezer.com/#/analyses/22c91372-2663-4d5c-940f-7090f222cef0/sub/7fe6c86f-cc1e-4d36-80e2-6fca66f0e4cd

Mitigating Emotet

(A) I have a suspicious file that I suspect is Emotet

You can analyze the suspicious file with Intezer's community edition. It's free 🙂 You'll have the same insights as the above examples, and know instantly if the file is indeed malicious, and if it's really Emotet or a different kind of threat. Search for other malicious Emotet files across your network using instructions in (C).

(B) I suspect that one of our machines is infected by Emotet

1. Check for scheduled tasks using Autoruns to spot binaries that are a part of Emotet's persistence mechanism. Analyze these files using VirusTotal or Intezer.

2. Since Emotet's payload resides in memory, memory analysis is your best choice to triage and analyze a suspicious endpoint. Use winpmem to obtain a memory image of the computer, and then use Volatility to extract all loaded executables from memory. You can do that by using Volatility's procdump and dlldump plugins. Once you obtain all binaries from memory, send them to Intezer for analysis to identify Emotet payloads. Code similarity detection works just fine also with memory-extracted items.

3. Submit the detected Emotet payload to your anti virus vendor and update signatures. As initial response, you can kill the process and remove the persistence binary using Autoruns.

(C) I suspect, or I don't know, if machines across my network are infected with Emotet

As mentioned before, due to the fact that Emotet resides in memory, a good approach would be to search for a YARA signature across all the network. For that, you'll need a relevant YARA rule that detects Emotet payloads, as well as a YARA memory scanner.

I recommend checking out Loki, an open-source YARA scanner by Florian Roth (@cyb3rops) that can help you scan endpoints across your network. For YARA rules to detect Emotet, there's one that GoDaddy's security team has written: https://github.com/godaddy/yara-rules/blob/master/emotet.yara, and the one within the CAPE project: https://github.com/ctxis/CAPE/blob/master/data/yara/CAPE/Emotet.yar.

Once a file is identified using a YARA rule, we recommend to dump the suspicious process and analyze it with Intezer Analyze, to confirm the initial detection. Signature-based detection such as YARA may result sometimes in false-positives, especially in memory, so it's always a good practice to reconfirm.

Summary

Emotet and other Banking Trojans cause a huge pain for enterprise organizations and end-users alike. Usually, these types of malware are highly evasive and polymorphic, which poses a challenge for many security solutions to protect against them. However, time and time again we observed that when utilizing code similarity detection techniques, these threats can be better identified and mitigated.

We hope that the information provided in this post, together with our free malware analysis platform, will help to decrease the number of Emotet victims worldwide.

Itai Tevet
Once led a government CERT. Now, CEO at Intezer, changing the way we detect, analyze and respond to malware.