Technical Report of MRG Effitas and CrySyS Lab

An independent test of APT attack detection appliances

Gábor Ács-Kurucz[1], Zoltán Balázs[2], Boldizsár Bencsáth[1], Levente Buttyán[1], Roland Kamarás[1], Gábor Molnár[1], Gábor Vaspöri[1]

November 26, 2014.

[1] CrySyS Lab, Budapest (www.crysys.hu)
[2] MRG Effitas (www.mrg-effitas.com)

An independent test of APT attack detection appliances

Introduction

The term Advanced Persistent Threat (APT) refers to a potential attacker that has the capability and the intent to carry out advanced attacks against specific high profile targets in order to compromise their systems and maintain permanent control over them in a stealthy manner. APT attacks often rely on new malware, which is not yet known to and recognized by traditional anti-virus products. APT attackers typically use spear phishing or watering hole techniques to deliver the malware to victim computers where it is installed by enticing the user to open the file containing the malware or the link pointing to it. Installation of the malware may also involve exploiting some known or publicly unknown vulnerability in the victim system, or social engineering. Once the malware is installed, it may connect to a remote Command & Control server, from which it can download updates and additional modules to extend its functionality. In addition, the malware may use rootkit techniques in order to remain hidden and to provide permanent remote access to the victim system for the attackers.

As traditional anti-virus products seem to be rather ineffective in detecting new malware[3], and hence, mitigating APT attacks, a range of new solutions, specifically designed to detect APT attacks, have appeared on the market in the recent past. These anti-APT tools typically identify suspicious files on hosts and/or in the network traffic, open those files in a sandbox environment on virtual machines under various configuration settings, analyze the behavior produced by the virtual machines, and try to identify anomalies that may indicate the presence of a malware or an exploitation attempt. Well-known examples for such APT attack detection tools include Cisco's SourceFire, Checkpoint, Damballa, Fidelis XPS, FireEye, Fortinet, LastLine, Palo Alto's WildFire, Trend Micro's Deep Discovery and Websense.

There is no doubt that these new tools are useful, which is also underpinned by the fact that they have helped to identify several zero-day exploits recently.[4,5] However, determining the real

[3] See the 2012 Imperva report on Assessing the Effectiveness of Antivirus Solutions, available at http://www.imperva.com/docs/HII_Assessing_the_Effectiveness_of_Antivirus_Solutions.pdf (availability verified on 10/28/2014)
[4] http://www.fireeye.com/blog/technical/cyber-exploits/2013/02/in-turn-its-pdf-time.html (availability verified on 10/28/2014)

effectiveness of these tools is challenging, because measuring their detection rate would require testing them with new, previously unseen malware samples with characteristics similar to those of advanced malware used by APT attackers. Developing such test samples require special expertise and experience obtained either through the development of advanced targeted malware or at least through extensive analysis of known samples.

The difficulty of testing APT attack detection tools properly has recently manifested itself in a lively dispute over a comparative test performed by NSS Labs in 2013 and 2014. NSS Labs is an independent testing firm specialized in testing security products, such as Intrusion Prevention Systems. In 2013, they started to test APT attack detection solutions, which they call Breach Detection Systems. When they made their first results available to their clients in July 2013, FireEye, one of the key players in the anti-APT market, heavily criticized the testing methodology used by NSS Labs and they withdrew from further testing. FireEye claimed in a blog post[6] that NSS Labs poorly selected the samples which they used in the test, as "the NSS sample set doesn't include Unknowns, Complex Malware (Encoded/Encrypted Exploit Code & Payload), and APTs." While this claim was refused by NSS Labs[7], it seems to be true that most of the NSS Labs samples were not custom developed for the purpose of the test, but they were known samples or slightly modified versions of known samples.

For this reason, we at MRG Effitas and CrySyS Lab decided to join our forces and perform a test of leading APT attack detection tools using custom developed samples[8]. MRG Effitas has a lot of experience in testing anti-virus products, while the CrySyS Lab has a very good understanding of APT attacks gained through the analysis of many targeted malware campaigns (including Duqu, Flame, MiniDuke and TeamSpy). Therefore, collaborating and bringing together our complementary sets of expertise looked like a promising idea.

Unlike in the NSS Labs test, our goal was not to determine the detection rates of different APT attack detection products, because that would have required testing with a large set of custom developed malware samples, which was not feasible to obtain within the limited time frame and with the limited resources we had for the

[5] http://www.fireeye.com/blog/technical/targeted-attack/2014/10/two-targeted-attacks-two-new-zero-days.html (availability verified on 10/28/2014)

[6] http://www.fireeye.com/blog/corporate/2014/04/real-world-vs-lab-testing-the-fireeye-response-to-nss-labs-breach-detection-systems-report.html (availability verified on 10/27/2014)

[7] https://www.nsslabs.com/blog/dont-shoot-messenger (availability verified on 10/27/2014)

[8] We do not call them *malware* samples, as neither our intent was malicious, nor the samples have any malicious functionalilty, but they serve only testing purposes.

test. Instead, our goal was simply to implement some ideas we had for bypassing cutting-edge APT attack detection tools without actually being detected, and to test if our ideas really work in practice.

We developed 4 custom samples in 2 weeks and without access to any APT attack detection tools during the development, and then later tested with these samples 5 APT attack detection solutions in Q3 2014. All 5 tested products are well-established in the market; however, we cannot mention vendor names in this public report. The result of the test was alarming:

- one of our 4 custom samples bypassed all 5 products,
- another sample of the remaining 3 samples bypassed 3 products,
- only the two simplest samples have been detected by the tested products, and even those triggered alarms with low severity in some cases.

In this report, we describe our test methodology, including a brief description of each sample we developed for the purpose of the test, and we present the test results in more details. We decided to publish this report for multiple reasons:

- First of all, we believe that our test was more appropriate for evaluating the detection capabilities of APT attack detection tools than the earlier NSS Labs test was, because we used custom developed samples that resemble better the malware used in APT attacks than the samples used in the NSS Labs test. At least, we cannot be blamed that our test did not "include Unknowns, Complex Malware (Encoded/Encrypted Exploit Code & Payload), and APTs."

- Second, some of the products that we tested seem to be overestimated by the users who believe that those products are silver bullets. This misconception may stem from the products' marketing strategies and outrageously high prices that suggest that they are truly exceptional tools which will catch every attack. The danger is that then users may believe they do not anymore need to spend effort for internal monitoring of their networks, log analysis, host based intrusion detection, etc. We, on the other hand, have already emphasized at multiple occasions that these products can and will be bypassed by determined attackers. So users should periodically ask the questions what if despite all the expensive tools deployed, the attackers managed to successfully compromise the system, and how to check whether a system has already been compromised or not? Our test is a clear proof that mainstream APT attack detection tools can be bypassed (even with moderate effort), and if we could do that, then APT attackers will also be able to do that, if they have not done so yet.

- Third, we observed that some vendors of APT attack detection tools are often reluctant to participate in tests that try to evaluate the effectiveness of their products. On the one hand, we understand their caution, as a test may lead to results that, if published, may ruin their business. On the other hand, we all know that the approach of security by obscurity has its own pitfalls, and a false sense of security is actually worse than not having any protection but being aware of that. So, by publishing this report, we would like to encourage anti-APT tool vendors to participate in independent tests more readily and cooperatively, in order to have sufficient amount of convincing results about their products, based on which well-informed decisions can be made by the users.
- And last but not least, we believe that there are significant differences in the APT detection capabilities of the tested products, and users should be aware that not all vendors provide the same detection rate. This is a well-known fact in the traditional antivirus industry, and the same applies to the novel APT detection tools. Thus, clients who plan to buy these products should run proper tests either in-house or outsourced before spending their money.

Test methodology and setup

Test setup

The network diagram in Figure 1 explains the logical topology of the test setup. Before running the tests, vendors (or their representative integrator companies) approved the test setup as fully functional.

Goal

All test samples had the objective of implementing traditional RAT functionality, including remote interactive code execution, as well as file download and upload. Remote communication was implemented via back-connect C&C communication, where the maximum interval between the polling requests was no more than 1 minute. Attackers sometimes have higher limits.

The test

All tests were executed only once during the tests. In case of 2 tested products, the C&C traffic was alive for 30 minutes, and in case of the remaining 3 tested products, it was alive for 24 hours. This is because we only had a limited timeframe for the test for the first 2 products. During the test, we downloaded large files from the victim client, and executed local commands, simulating real attackers. We have not initiated any lateral movement.

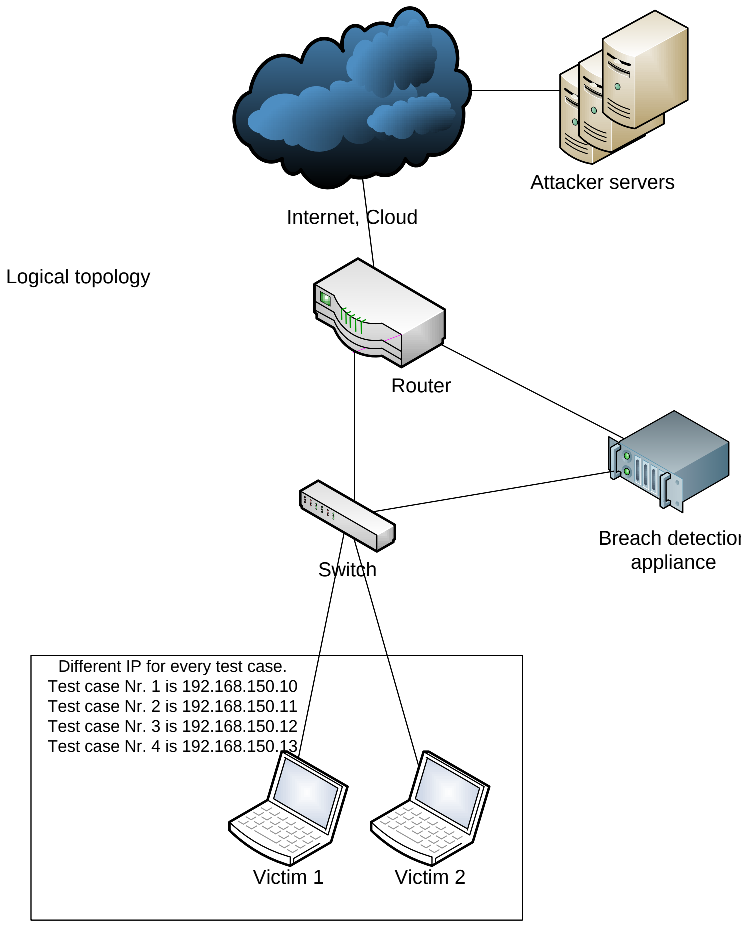
Attacker servers
Internet, Cloud
Logical topology
Router
Breach detection appliance
Switch
Different IP for every test case.
Test case Nr. 1 is 192.168.150.10
Test case Nr. 2 is 192.168.150.11
Test case Nr. 3 is 192.168.150.12
Test case Nr. 4 is 192.168.150.13
Victim 1
Victim 2

Figure 1: Logical topology of the test setup

Test sample 1

This test case simulates attackers with limited knowledge and resources. The malware delivery is a plain known Java Runtime Environment exploit, with poor obfuscation. After successful exploitation, the shell-code downloads and executes a publicly available RAT (Remote Admin/Access Tool/Trojan). This RAT installer is obfuscated with encryption, where the decryption routine brute-forces the key. This decryption phase is resource intensive, and can thwart sandbox analysis of the malware. After successful decryption, the malware shows an error dialog box, which has to be accepted by the user in order to run the real payload. The RAT payload connects back through an encrypted TCP channel to the RAT server (RAT client in RAT terminology). This test sample should have been detected by all of the tools which provide zero-day malware detection. The use of known exploits and publicly available RAT tools are common in APT attacks, although this initial phase of the attack cannot be considered as advanced. We used new domains and IP addresses with unknown reputation in this attack.

Test sample 2

Test sample 2 simulates attackers with moderate knowledge and resources. In this case, instead of Java exploits, the Java self-signed applet attack has been used. The applet that delivers the malware has been generated with publicly available tools, without obfuscation. Once the user accepts the execution of the signed applet, the malware is dropped to the user's temp folder and started. The malware executes Metasploit's reverse_http Meterpreter shellcode, after initial anti-debug and anti-sandbox techniques. The anti-sandbox technique used in this malware is in-house developed by MRG Effitas and CrySys Lab, thus, it is not flagged by sandboxes as a sandbox-detection activity. If the malware detects debugging or it detects that it is running in a sandboxed environment, then it immediately quits. If it detects that it is running in a user workstation, it executes the shell-code, which connects to the Metasploit server, and downloads the obfuscated stage2 metsrv.dll. In this way, network appliances won't be able to see the clear DLL on the network. The DLL is loaded on the client side using reflective DLL injection method. The HTTP based C&C protocol has been rewritten, and it is encrypted between the client and server. We used new domains and IP addresses with unknown reputation in this attack. The self-signed applet may be suspicious for the detection tools, but the automated dynamic analysis of the sample itself won't reveal its real functions.

Test sample 3

This test case simulates attackers with moderate knowledge and resources. In this case, Microsoft Office Visual Basic macro code execution has been used as a delivery method. After the macro code execution is allowed by the user, the Visual Basic code executes the shellcode directly in the Office process space. This shellcode is similar to the test 2 sample, as it is based on a Metasploit's reverse_http, and downloads the obfuscated stage2 metsrv.dll, thus network appliance won't be able to see the clear DLL on the network. The DLL is loaded on the client side using reflective DLL injection method. The HTTP based C&C protocol has been rewritten, and it is encrypted between the client and server. We used new domains and IP addresses with unknown reputation in this attack. This attack can be detected by the Visual Basic macro code easily, but from and end host point of view, detection is hard, as there is no malware written to the disk.

Test sample 4 - BAB0[9]

BAB0 is a custom designed sample written in C++ with a server side written in PHP. It was designed to be as stealthy as possible, and utilizes multiple methods to avoid detection. Actually, this test case simulates attackers with moderate resources and some understanding of the state-of-the-art

[9] Babo means hobbit in Hungarian. We called this sample Babo, as its objective was to stealthily bypass all state-of-the-art defenses, while actually being very simple, and this situation shows a parallel to the story of the Lord of the Rings, where Frodo, the small hobbit managed to bypass all defenses of the fearsome Sauron, the Lord of Mordor, and reached Amon Amarth, where the One Ring was finally destroyed.

detection tools and how advanced malware work. For example, this can simulate organized criminals when attacking high value targets. On the other hand, nation state attackers surely have more resources and knowledge to develop even stealthier malware.

The executable of BAB0 is downloaded by the victim as part of an HTML page, where it is actually hidden in an image with steganography. Thus, the executable never appears in clear in the network traffic. The downloaded page also contains scripts that extract the executable from the image when the user clicks on it. To avoid extracting the executable in a sandbox environment on the detection tools, the website's underlying HTML and JavaScript code is misleading for an automated analysis environment, but it has nothing special from a user's perspective. On the other hand, the page does not use CAPTCHA or other Turing test methods that would be unfair from a testing perspective. The user has to simply click on something that appears to be a download button.

Once the sample is running, it presents a decoy program to the user to appear as an ordinary program. It does not try to modify the registry or any configuration on the machine by itself. Persistence can be achieved later by sending commands that add the executable to the appropriate registry entries or making it start with the system in some other ways.

To hide the C&C network traffic, the client simulates a user clicking on links in a web forum, and downloads full HTML pages with CSS style sheets and images. The real C&C traffic is hidden inside these HTTP requests using data hiding methods. In the tests, we hosted the C&C server on domains with some positive reputation. It helped to simulate a fairly common scenario when the malware author compromised domains without negative reputation to host (part of) the C&C infrastructure. The command types that can be sent to the client include: directory traversal, file download and upload, and command execution.

Test results

The following table contains the test results. In favor of the tested products, we marked as "detected" even those cases where the given product detected the given sample with low confidence (or as a low severity event). Detecting the sample means that at least one of the following stages has been detected: exploit, malware download, command and control channel.

Sample\Product	Product 1	Product 2	Product 3	Product 4	Product 5
Test sample 1	detected	detected	detected	detected	detected
Test sample 2	detected	detected	detected	detected	detected
Test sample 3	detected	bypassed	bypassed	detected	bypassed
Test 4 - BAB0	bypassed	bypassed	bypassed	bypassed	bypassed

Conclusion

The main message of this work is that novel anti-APT tools can be bypassed with moderate effort. If we were able to develop samples that were not detected by these tools without actually having access to any of the tested products during the development phase, then resourceful attackers who may be able to buy these products will also be able to develop similar samples, or even better ones. In addition, the test results also show that there are differences between the different products in terms of their detection capabilities, as some of the products detected our test sample 3, while others did not. We cannot reveal in this report which products performed better, but we can help organizations to test the products integrated in their environment.

Next steps

We plan to develop more custom samples for using them in our future tests. We are also thinking about creating a test environment where zero-day browser exploits can be used efficiently. We have already developed an in-house test environment, where known good and known bad URLs, IP addresses, domains can be simulated, and we may use this in our future tests.

Finally, we have a strong intention to publish BAB0 in the near future. This may seem to be controversial, as making the details of BAB0 publicly available can help attackers. We have a different opinion: Powerful attackers have probably been using already similar tricks, but apparently detection tools are not yet prepared to cope with them. By publishing BAB0, we push anti-APT vendors to strengthen their products, which will ultimately make the attackers' job harder.

Contacts

For further information, please contact either Zoltan Balázs (Zoltan.Balazs@mrg-effitas.com) or Levente Buttyán (buttyan@crysys.hu). Please note that we cannot provide any vendor specific information about the tests, but we can help organizations to test the products integrated in their environment.