SafeBreach

THE ADVENTURES OF AV AND THE LEAKY SANDBOX

A SafeBreach Labs research paper by

Amit Klein, VP Security Research, SafeBreach and

Itzik Kotler, co-founder and CTO, SafeBreach

July 2017

ABSTRACT

We describe and demonstrate a novel technique for exfiltrating data from highly secure enterprises which employ strict egress filtering - that is, endpoints have no direct Internet connection, or the endpoints' connection to the Internet is restricted to hosts required by their legitimately installed software. Assuming the endpoint has a cloud-enhanced anti-virus product installed, we show that if the anti-virus (AV) product employs an Internet-connected sandbox as part of its cloud service, it actually facilitates such exfiltration. We release the tool we developed to implement the exfiltration technique, and we provide real-world results from several prominent AV products (by Avira, ESET, Kaspersky and Comodo).

Our technique revolves around exfiltrating the data inside an executable file which is created on the endpoint (by the main malware process), detected by the AV agent, uploaded to the cloud for further inspection, and executed in an Internet connected sandbox. We also provide data and insights on those AV in-the-cloud sandboxes. We generalize our findings to cover on-premise sandboxes, use of cloud-based/online scanning and malware categorization services, and sample sharing at large. Lastly, we address the issues of how to further enhance the attack, and how cloud-based AV vendors can mitigate it.

INTRODUCTION

Exfiltration of sensitive data from a well-protected enterprise is a major goal for cyber attackers. Network-wise, our reference scenario is an enterprise whose endpoints are not allowed direct communication with the Internet, or an enterprise whose endpoints are only allowed Internet connections to a closed set of external hosts (such as Microsoft update servers, AV update servers, etc.). And since the organization is "well protected", it's quite likely to have mandatory anti-virus (AV) security software installed on every endpoint. Interestingly, many AV vendors advertise "cloud AV" offerings, in which the endpoint AV software consults a cloud service about its local findings. Note that when employing a "cloud AV", even for endpoints that are completely restricted from accessing the Internet, the endpoints are still connected to an internal network, and are allowed to send data over the internal network to an AV management server in the organization (which is allowed to connect to the AV cloud). This architecture, therefore, is not truly air-gapped.

There are various exfiltration techniques already disclosed, and in fact, our 2016 paper **"In Plain Sight: The Perfect Exfiltration"** provides both references to comprehensive lists of such techniques, and a new set of such techniques. However, all those techniques assume the endpoint can connect to arbitrary Internet hosts, or assume lax security at the enterprise, or some requirements on the hosts that are used for exfiltration. In the current paper, we look at a case wherein these assumptions do not hold, and particularly wherein the direct Internet connection, if allowed at all, is limited to a small set of hosts (those that are required for proper functioning of legitimately installed software on the endpoint). We show that even under these restrictions, it is still possible under some likely conditions, to exfiltrate data to the outside world.

Our contribution:

1. We describe an innovative technique to exfiltrate data using cloud-based AV sandboxing, that can work even if the endpoint is prevented from directly connecting to the Internet, or when it can only connect to the hosts necessary for the functioning of the legitimately installed software on the endpoint.
2. We provide a free, open source tool which implements our technique.
3. We provide real world results of testing the technique against leading AV products.
4. We provide some insights regarding the nature of cloud-based AV sandboxes from leading AV vendors.
5. We describe extensions to our original technique (not yet implemented) that can improve the success rate of the technique.

RELATED WORK

Exfiltration in general

As mentioned above, there is quite a large corpus of research around exfiltration techniques in general, such as:

- "Covert Channels in TCP/IP Protocol Stack" by Aleksandra Mileva and Boris Panajotov
- "A survey of covert channels and countermeasures in computer network protocols" by Sebastian Zander, Grenville Armitage and Philip Branch
- "Covert timing channels using HTTP cache headers" by Denis Kolegov, Oleg Broslavsky and Nikita Oleksov

Our research differs from the methods described by these papers by addressing the challenge of exfiltrating from endpoints which are not directly connected to the Internet, or have very limited connection to the Internet.

Exfiltration from disconnected endpoints

There is a lot of research published recently about exfiltrating data from endpoints over non-network media, e.g.:

- "LED-it-GO Leaking (a lot of) Data from Air-Gapped Computers via the (small) Hard Drive LED" by Mordechai Guri, Boris Zadov, Eran Atias and Yuval Elovici,
- "DiskFiltration: Data Exfiltration from Speakerless Air-Gapped Computers via Covert Hard Drive Noise" by Mordechai Guri, Yosef Solewicz, Andrey Daidakulov and Yuval Elovici
- "BitWhisper: Covert Signaling Channel between Air-Gapped Computers using Thermal Manipulations" by Mordechai Guri, Matan Monitz, Yisroel Mirski and Yuval Elovici, etc.

These methods tackle the challenge of a truly isolated machine ("air gapped") on one hand, but typically require an attacker presence physically near the endpoint on the other hand. Our research addresses a slightly different question – the endpoint may be indirectly connected to the Internet (through other machines which are allowed to access the Internet), or may be allowed to connect to a closed set of hosts, but on the other hand, we are not restricted to physical proximity, and can carry out exfiltration to practically any other Internet-connected host.

Exfiltration via 3rd party sites

An IPID-based exfiltration method such as "Covert Communications Despite Traffic Data Retention" by George Danezis can use one of the allowed Internet hosts (e.g. Microsoft's update server) to exfiltrate data. However, this method is nowadays practically outdated since modern operating systems do not implement a globally incrementing IPID generator. Moreover, the whitelisted sites are likely to be very busy, therefore the IPID-leaking method will encounter a lot of noise and will likely be impractical.

Another exfiltration using a 3rd party site would be to send a TCP SYN packet to an open port at the 3rd party site, with a source IP pointing at an attacker host (source IP spoofing). The 3rd party site will respond in a TCP SYN+ACK packet sent "back" to the source IP – i.e. to the attacker host. The payload can be embedded in the TCP ISN (initial sequence number) and/or the TCP source port, both of which are returned in the SYN+ACK packet.

A similar approach can be to send a UDP packet (request) to an open UDP port at the 3rd party site, again with a source IP pointing at an attacker host (source IP spoofing). The 3rd party site will respond with a UDP response packet sent "back" to the source IP. The payload can be embedded in the source port, or as part of the query data, if that can be deduced from the response data (for example, a DNS response contains the DNS query). However, high security enterprises may employ IP egress filtering, and thus drop the outbound endpoint packets whose IP addresses do not belong to the enterprise. In addition, NAT/PAT devices may alter the source port and source IP address and thus eliminates this channel. Moreover, TCP/UDP-aware gateways/firewalls may terminate a TCP connection and establish their own, thus again eliminating this channel.

Another approach is demonstrated in **"In Plain Sight: The Perfect Exfiltration"** by Amit Klein and Itzik Kotler. There, the authors describe exfiltration based on subtle manipulations of the application state in a 3rd party site. This approach does not work well with the hosts to which an endpoint is allowed to connect (e.g. software update hosts), since these hosts don't need a caching layer in front of them, and their application is not rich in states that can be easily manipulated.

Triggering av products

In **"Art of Anti Detection 1 – Introduction to AV & Detection Techniques"** , Ege Balci provides an extensive list of triggering behaviors for av products. Our technique uses such triggers (we experimented with 2 triggers but the technique can make use practically of each and every trigger listed) – our innovation is not in coming up with ways to trigger AV products, but rather abuses the actions some AV products take once the trigger is fired

Research on sandboxes

In **"AVLeak: Fingerprinting Antivirus Emulators Through Black-Box Testing"**, Jeremy Blackthorne, Alexei Bulazel, Andrew Fasano, Patrick Biernat and Bülent Yener describe a way to fingerprint an AV emulation sandbox as a black-box. They do not describe a way to exfiltrate data from an endpoint machine.

In Google's Project Zero entry **"Comodo: Comodo Antivirus Forwards Emulated API calls to the Real API during scans"** , Tavis Ormandy describes bugs in Comodo Antivirus which allow an attacker to run code in elevated privileges, and to exfiltrate data out of the endpoint. However, the underlying assumption there is that the endpoint is allowed to connect to arbitrary Internet hosts. Our concern is with scenarios where such connections are not allowed, usually via an organization firewall (i.e. a gateway separate from the endpoints).

In **"Your sandbox is blinded: Impact of decoy injection to public malware analysis systems"** , Katsunari Yoshioka, Yoshihiko Hosobuchi, Tatsunori Orii and Tsutomu Matsumoto describe a technique to fingerprint a public-facing sandbox. Like AVLeak, they do not concern themselves with exfiltrating data from a regular (endpoint) machine.

In **"Enter Sandbox – part 8: All those... host names... will be lost, in time, like tears... in... rain"** Hexacorn Ltd. provides a list of 800+ computer names allegedly used in sandboxes, thus creating a de-facto fingerprinting database for sandboxes.

In **"Sandbox detection: leak, abuse, test"**, Zoltan Balazs describes how to extract sandbox fingerprints (e.g. screen resolution, computer name, software installed, CPU type and count, memory size, mouse movements, etc.)

In **"Art of Anti Detection 1 – Introduction to AV & Detection Techniques"**, Ege Balci provides heuristics for sandbox detection.

Our research studies exfiltration through AV sandboxes of data obtained from an endpoint. Fingerprinting the sandboxes themselves wasn't our main object, though we did observe several properties of AV sandboxes, and we share these in the paper.

BACKGROUND

We are interested in two network architectures that can be found in highly secure organizations:

Indirect Internet Access

The corporate has cloud-based AV agents deployed on all endpoints. The corporate has a central AV management server (distributing updates and collecting samples and events) which is allowed to connect to the AV's cloud service. All other machines (endpoints) are prohibited from accessing the Internet. The access control can be enforced by a firewall/gateway between the corporate's LAN and the Internet. This firewall/gateway has a rule that prevents all machines except the AV management server from accessing the Internet. It is assumed that spoofing TCP (or IP) traffic is impossible – e.g. the AV management server is located in a different segment and the firewall/gateway employs strict ingress filtering.

In this architecture, the endpoints do not have any direct Internet access. Therefore, regular network-based exfiltration methods are ineffective here. However, software on the endpoint can influence the AV management server (for example, through the agent) and cause it to communicate with the AV cloud hosts. We will use this observation to construct an exfiltration technique.

Limited Direct Internet Access

The corporate has cloud-based AV agents deployed on all endpoints. All endpoints are prohibited from accessing the Internet, except for a closed set of hosts necessary for the functioning of the legitimately installed software on the endpoints (e.g. the AV cloud hosts, Microsoft's update servers, etc.). The access control can be enforced by a firewall/gateway between the corporate's LAN and the Internet. This firewall/gateway has a rule that prevents all machines from accessing the Internet, except to a closed set of hosts.

In this architecture, the endpoints have very limited access to the Internet. Theoretically, one can exfiltrate data by sending packets to one of the allowed Internet hosts (e.g. to a Microsoft update server), but this requires the attacker to also be able to eavesdrop on the communication between the organization and the (Microsoft update) server. This is not feasible for a non-state sponsored attacker, and thus it is out of scope of this paper. Alternatively, an endpoint can send IP packets to one of the whitelisted hosts, thereby theoretically enabling the various "exfiltration via 3rd party sites" techniques discussed above, but as we wrote there, none of these methods is effective nowadays for highly secure enterprises (with egress filtering) and with the whitelisted Internet hosts.

Triggering an AV agent

AV products typically employ numerous behavioral rules that attempt to detect suspicious/malicious processes. Examples for behavioral rules are:

- Writing to disk an image (file) which is known to be malicious (known malware/virus)
- Setting an auto-start entry (persistence)
- Injecting code into other processes

- Unpacking code in memory
- Connecting to known malicious hosts (known C&C)
- Making system level modifications (e.g. replacing system files/libraries, adding/changing entries in the hosts file, modifying the DNS settings)
- Making changes to the browsers, e.g. installing plugins, modifying the proxy settings, etc.

This is, of course, just a small subset of all possible triggers.
Once a process activity triggers the AV product, the image file of the process is marked as suspicious/malicious (and, depending on the AV product, may be sent to the cloud for further inspection.)

Cloud AV sandboxing

We are going to abuse the cloud AV sandboxing feature that many AV vendors use. The rationale for this feature is that it enables the AV vendor to offer lightweight agent software, and carry out the heavy-lifting security analysis work in the cloud. Specifically, in such an architecture, the AV agent needs to conduct only basic security checks against other processes and files, allowing for a grey area where a binary "malicious/non-malicious" decision cannot be determined locally. A process/file falling into this grey area is sent to the cloud for further analysis, and a security decision is obtained from the cloud (sometimes in near real time).
A sample that arrives at an AV cloud sandbox is typically executed there and its behavior is observed. The sandbox is not a sensitive environment, therefore a malicious executable can be run there with no harm to real users or resources. Based on the observed behavior, the AV vendor can arrive at a more educated decision about the nature of the sample (i.e. whether it is malicious or not).

An AV cloud sandbox may be isolated from the Internet, or connected to it. There's a good argument in favor of a connected sandbox, as it allows the sample to run in a more "natural" environment. For example, an unknown file (which is in fact a new malware dropper) can connect to its C&C server, download a secondary malware (such as the well-known Zeus malware) and run it. The AV logic can then inspect the Zeus file and classify it is a known malware file, and thus determine that the original sample is actually a malware dropper, hence it is malicious in itself. All this is impossible when the AV cloud sandbox is isolated from the Internet.

Exfiltration methods from an Internet-connected machine

There are numerous ways to exfiltrate data from an Internet-connected machine, as mentioned in the "related work" section. If stealth requirements are alleviated (e.g. when a one-time attempt is made), then exfiltration can be as simple as connecting to an Internet host (owned/accessible by the attacker), e.g.:

- Sending HTTP/HTTPS request to the attacker's host
- Forcing DNS resolution (for hostnames in a domain owned by the attacker, and where the attacker also controls the authoritative name server for the domain)
- Sending email to an attacker mailbox (SMTP)
- Sending an IRC message
- Pinging (ICMP Echo) an attacker host
- Submitting (via HTTP/HTTPS) a comment in a popular web forum

This is, of course, merely a very small subset of all the possible ways to transmit data out of the machine.

EXFILTRATION THROUGH AV CLOUD SANDBOXES

We present a novel method of exfiltrating data from endpoints which are not directly connected to the Internet, or whose connection to the Internet is limited to only those hosts needed for the functionality of its legitimately installed software. Our technique requires the following:

- An AV product (agent) installed on endpoints, which submits to the AV cloud unknown/suspicious executable files:
  - Directly, via an Internet connection from the endpoint to the AV cloud; or
  - Indirectly, by sending the sample to an enterprise AV management server and having the latter submit the sample via a direct Internet connection to the AV cloud (in real time or in offline/batch mode), or manually via copying files (though a physical medium e.g. a USB stick) from the enterprise AV management server to an Internet-facing machine and submitting the files from there.
- The AV cloud service employs a sandbox which can directly connect to the Internet.
- The attacker's process (malware) is running on the endpoint.

Our technique can exfiltrate data in this scenario even when the endpoint has a very limited Internet connection (e.g. one which is not used by the user – such as limiting the connection to software update servers), and even when no direct connection is allowed at all.

Details of the technique

The attacker process (called "Rocket") contains a secondary executable (called "Satellite") as part of its data. The Satellite can be encrypted/compressed to hide the fact that it is another executable, thus the Satellite can be no more than a piece of data in the Rocket memory space (and file) that does not jeopardize the Rocket.
The Satellite contains a placeholder for arbitrary data ("payload") to be exfiltrated. The location of the placeholder should be known to the Rocket, e.g. as an offset+length, or as magic-start/end markers, within the data of the embedded Satellite.

The attack proceeds as follows:

1. The Rocket collects the data (payload) it needs to exfiltrate.
2. The Rocket decrypts/decompresses the Satellite into its useful image in memory, and embeds the payload in a designated area in the Satellite image. The payload can be compressed/encrypted to further ensure that it is not suspicious by itself. The Rocket then writes the Satellite image to disk, as a file (henceforth, the "Satellite file").
3. The Rocket spawns the Satellite (from its file) as a child process.
4. The Satellite performs an intentionally suspicious action, such as persisting itself, or writing a known malware file to disk.
5. The endpoint AV product detects the suspicious action of the Satellite, possibly quarantines it, and sends its image file to the AV cloud. Note that the Satellite file contains the payload.
6. The cloud AV executes the Satellite file in an Internet-connected sandbox.
7. The Satellite process can now try any of the Internet-based exfiltration methods to exfiltrate the (encrypted/compressed) payload to the attacker, e.g. by sending it in an HTTP/HTTPS POST request to an attacker host, or by breaking it into small pieces, hex-encoding it and using it as a DNS subdomain for DNS resolution requests whose domain is owned by the attacker.

Note that this attack is “noisy” in the sense that the AV product will flag the Satellite file as suspicious and as such this may have visible impact on the user (see Figures 1-2 below), as well as visibility in logs and records. However, for a one time exfiltration attack this will already be too late, as the payload will already be traveling to the cloud by the time this incident is investigated by flesh-and-blood analysts.

RESULTS WITH LEADING AV PRODUCTS

In February-March 2017, we ran tests against multiple AV products. We provide details of the tests and results in this section.

Test setup

We built Rocket-Satellite combinations as explained above. The code was written in C/C++ and compiled in Microsoft VC 2015 – Rocket was compiled for an x86 target, and Satellite – mostly for an x64 target, but sometimes for an x86 target (this was a random choice, we don’t place too much weight on the target architecture). For simplicity, the Satellite and the payload were not encrypted or compressed. The payload area was marked with “magic-start” and “magic-end” markers. For best results, The Rocket executable file should be placed in in one of the regular folders, such as the “Desktop” folder (but not in the “Downloads” folder which is probably subject to different AV policies).

Rocket collects data from the machine (in out proof of concept – the machine name, and additionally some hard-wired short string that we used to identify the exact test we ran), prepares the Satellite per above, writes the Satellite to the current working directory and runs it. It should be noted that the due to the way they are created (with experiment-specific strings), each Satellite file was unique, i.e. had a unique MD5/SHA1/SHA2 hash signature.

For triggers (Satellite actions), we chose two very simple triggers (each was tested on its own):

1. Writing the EICAR file to disk. The EICAR file is a de-facto standard file that deliberately triggers AV products. This is done by opening a file named “daemon.com” in the current working directory, and writing the 68 bytes of the EICAR data to this file using the C runtime library function fwrite().
2. Moving the Satellite image file to the user’s startup folder (%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup) as “foo.exe”, overriding any existing “foo.exe” file. This is done using the DeleteFile() and MoveFile() WinAPI calls.

For exfiltration, we chose two methods, both are attempted in the same Satellite run:

1. DNS-based exfiltration. The Satellite encodes the payload (using hex-encoding), then breaks it into chunks that fit into DNS labels, and exfiltrates each chunk as a subdomain in a host-resolution query for a domain owned by the attacker (so eventually the query arrives at the authoritative DNS server run by the attacker). The resolution request is carried out via invoking the Windows Socket library function getaddrinfo() or gethostbyname().
2. HTTP-based exfiltration. The Satellite sends the payload as-is to the attacker's Internet host in an HTTP POST request body. This is done using WinINet’s HttpOpenRequest() and HttpSendRequest().

Free open source code for the above implementation is available in the GitHub repository at https://github.com/safebreach-labs/spacebin

Test results

We checked 10 products out of the top 11 anti-virus products from the vendor list in OPSWAT's "Antivirus and Compromised Device Report: January 2015". This is their last report on pure Anti-Virus vendors. We could not test SpyBot due to research time constraints.

The following table summarizes our results. Note that for products that are marked "-", it only means that we could not exfiltrate the payload via the combination of the two simple triggers we used, and the two naïve exfiltration techniques we implemented. It does not mean that other combinations cannot potentially succeed for these products.

Product	Version	Trigger	Exfiltration Method	Success
Avast Free Antivirus	12.3.2280			-
Microsoft Windows Defender	Client v. 4.10.14393.0 Engine v. 1.1.13407.0			-
AVG	Build 1.151.2.59606 Framework v. 1.162.2.59876			-
Avira Antivirus Pro	15.0.25.172	Persistence	DNS, HTTP	Yes
Symantec Norton Security	22.8.1.14			-
McAfee Cloud AV	0.5.235.1			-
ESET NOD32 Antivirus	10.0.390.0	EICAR	DNS	Yes
Kaspesrky Total Security 2017	17.0.0.611(c)	Persistence	DNS, HTTP	Yes
Comodo Client Security	8.3.0.5216	Persistence	DNS, HTTP	Yes
BitDefender Total Security 2017	Build 21.0.23.1101 Engine v. 7.69800			-

As can be seen, 4 major Anti-Virus products (Avira, ESET, Kaspersky and Comodo) out of 10 were positively shown to facilitate exfiltration of data from “isolated” endpoints.

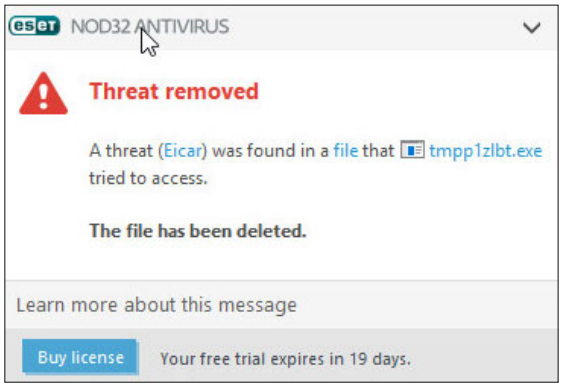

Figure 1 - ESET NOD32 Antivirus notification (writing EICAR file to disk)

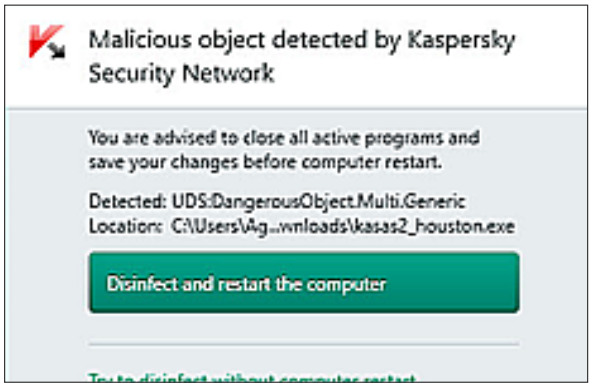

Figure 2 - Kaspersky Total Security 2017 notification (persistence in the Startup folder)

Insights on AV sandboxes

We encountered 44 different sandbox instances from 22 templates. In general, they belong to 10 classes, based on their computer name:

***name*PC** (1 template, 13 instances: REYNAPC, MALVAPC, ELEANOREPC, WRIGHTPC, BRIAPC, JORIPC, GABBIPC, HELSAPC, MAMEPC, SHARAIPC, ARACHONPC, FLORIANPC, EDITHPC) – In this class/template, the computer name comprises of a person's first name followed by “PC”. This class is operated by ESET (its DNS queries come from an IP owned by ESET). All machines encountered share some basic properties such as disk volume (e0b2a963), performance counter frequency (100MHz) and MAC address prefix (00:1B:21:13:37:xx – the last byte varies). HTTP traffic to the Internet is blocked, but DNS is enabled.

***something*-PC** (11 templates, 15 instances: WIN7-PC, ROGER-PC, DAVID-PC, ADMIN-PC, APIARY7-PC, ANTONY PC, LUSER-PC, PERRY-PC, KLONE_X64-PC, 0M9P60J5W-PC, MIKI-PC) – each template in this class is different (different CPU, different clock speed, different MAC address, different disk volume number).
The template 0M9P60J5W-PC had 5 instances (different computer/user/domain names but all exhibit the same properties). All templates use IPs in consumer ISPs to connect to the Internet. Usually the username is identical to the computer name, but for ROGER-PC we also encountered cases wherein the username was a seemingly random string of alphanumeric characters. This class is operated by Comodo and/or Kaspersky and/or Avira.

C-TT (2 templates: C02TT with 4 instances: C02TT22, C02TT26, C02TT36 and C02TT18; C06TT – one instance, C06TT43) – this class is used by Avira.

ZGXTIQTG8837952 (1 template, 4 instances) – this class is used by Comodo. In one case, we encountered it behind TOR.

spurtive (2 templates – "spurtive" and "circumstellar", one instance each) – used by Comodo.

5 one-template, one instance classes (ABC-WIN7, PC, WIN-U9ELNVPPAD0, PC-4A095E27CB, WIN-LRG949QLD21
Not much information is available since each was encountered only once or twice.

It appears that not a lot of effort is made to hide the identity of the sandboxes from the processes running on them. Most of the time, the names are fixed, and even if not, the disk volume serial number is not changed. In some cases, the MAC address clearly indicates a virtualized environment (VMware). The Performance Counter is also an indicator for virtualization in some cases (see http://www.securitygalore.com/site3/wpc-frequency-vm-os-matrix for details of this technique).

It is also noteworthy that several sandbox templates (e.g. "ROGER-PC") are shared (or OEMed) between several players. This is clear in the case of "ROGER-PC" (Comodo, Kaspersky and Avira), but can be deduced in the case of "BRIA-PC" and "JORI-PC" – both were only observed being used by Comodo, but their DNS traffic is coming from an IP owned by ESET. This may indicate for example, that Comodo uses third party AV engines/services to help evaluate unknown files. The full details of the sandboxes we encountered are provided in the appendix.

Additional observations:

- Reaction time (time between sample upload and exfiltration) of sandbox execution varies from minutes to hours.
- In some cases (ROGER-PC, ANTONY-PC) we noticed that sometimes the sandbox prevented the investigated process from accessing the environment variables.
- It appears that additional sandbox executions sometime took place days and even weeks after the sample was submitted. We suspect these were related to either batch processing/re-evaluation, or due to analysts manually investigating the samples.
- In some cases, we noticed that the sandbox ran the Satellite before the data was "burned" into it by the Rocket. Our theory is that the AV product statically scanned the Rocket file/process, discovered the embedded executable (Satellite), extracted it as-is and executed it.

ON-PREMISE AV SANDBOXES

Some AV vendors offer an alternative to cloud AV sandbox in the form of an on-premise sandbox, in which the suspicious executable can be run safely. In this configuration, the attack may still be carried out, provided that:

- The sandbox allows outbound (Internet) traffic of some sort; and
- The enterprise network configuration allows outbound (Internet) traffic from the sandbox of at least one of the protocols/traffic types allowed by the sandbox

We did not experiment with on-premise AV sandboxes. Even if we could find AV on-premise sandboxes that allow outbound traffic, the impact would be determined on a customer-by-customer basis as it depends on the enterprise network configuration as it applies to the sandbox.

Nevertheless, enterprises should be aware of this attack variant.

CLOUD-BASED ANALYSIS SERVICES

There are several online services that offer analysis of binaries – either by subjecting them to multiple AV products, or by running proprietary malware analysis logic. Assuming the attacker knows that such services are used (manually, or as a backend to other security products/services) in the attacked enterprise, the attacker can use the same technique described above to spawn a binary with embedded enterprise data to be run in the cloud by the online service and exfiltrate the enterprise data. In this case, while the online service facilitates the attack, it is not a vulnerability in the online service since there's no expectation from the service to prevent exfiltration.

Google VirusTotal (https://www.virustotal.com/) was tested with a vanilla (non-triggering) Satellite and with the two triggering Satellites, just to observe if any of its sandboxes are connected to the Internet. Indeed, we got back sandbox fingerprints from sandbox templates ROGER-PC (DNS and HTTP), ANTONY-PC (DNS and HTTP), VVHMQ-PC (DNS, HTTP) ,ABC-WIN7 (DNS, HTTP), PC (DNS, HTTP), ADMIN-PC (DNS, HTTP), PERRY-PC (DNS, HTTP), PC 4A095E27CB (DNS, HTTP), LUSER-PC (DNS, HTTP), APIARY7-PC (DNS, HTTP), MIKI-PC (DNS, HTTP) and MAMEPC (DNS). This is possibly due to VirusTotal's scanning with Avira and/or ESET and/or Comodo and/or Kaspersky, and/or by having triggered another cloud based product which we either didn't test, or failed to trigger.

Joe Security's Joe Sandbox Cloud - File Analyzer (https://www.file-analyzer.net/) was tested with a vanilla Satellite and was observed to leak via a single sandbox (DNS only, limited to 10 DNS queries). The File Analyzer sandbox we encountered has the environment variable "COMPUTERNAME" set to "226533".

Payload Security's Hybrid Analysis (https://www.reverse.it/) was tested with a vanilla Satellite and was observed to leak via a single sandbox (HTTP and DNS). The Hybrid Analysis sandbox name we encountered is 7YTRfV35L8.

We conclude that when online scanning services VirusTotal, Hybrid Analysis and Joe Sandbox Cloud (and perhaps others) are used to evaluate a binary (either manually or via a 3rd party product using these services as part of its backend), exfiltration becomes possible.

VENDOR STATUS

Cloud AV sandboxes

- Avira – fixed on May 2nd (10h30m!).
- ESET – fixed on May 15th or before.
- Comodo – fixed on May 26th .
- Kaspersky - provided to us on July 14th with the following statement: "If customers are concerned about this scenario, they may configure their device and security settings accordingly. Those using our consumer products can disable files sent to the cloud sandbox without affecting detection effectiveness. Those using corporate products can install Kaspersky Private Security Network, the on-premises version of the Kaspersky Security Network cloud service, or a private fix to disable files sent to the cloud sandbox."

Cloud-based analysis services

- VirusTotal – informed us on June 16th that they WONTFIX, and provided the following statement: "We have our sandboxes on the internet on purpose to allow them communicate with C2 machines so we record the traffic".
- Joe Security – fixed on June 19th (3d11h!) and provided the following credit acknowledgement: "Improved ISIM max UDP request config (@credits to Amit & Itzik from Safebreach)".
- Payload Security - informed us on June 23rd that "Payload Security chooses not to comment".

IMPROVING AV CLOUD SANDBOXES

Clearly, one way to prevent this attack is to block AV sandboxes from accessing the Internet. Indeed, some cloud AVs probably do exactly that.
Of course, this may be too strict in many cases, because it would be interesting to observe the Internet traffic of a sample. Therefore, we can tune the defense technique a bit. We can only apply Internet blocking for samples not coming from the Internet (i.e. binaries which are created on the machine, or which are obtained from another machine in the enterprise). The reasoning is this – a sample which is obtained from the Internet (e.g. downloaded from a website) does not carry any enterprise endpoint-specific payload, and as such, won't be able to exfiltrate anything useful from the endpoint when run in an Internet-connected sandbox.

The same applies for on-premise AV sandboxes.

FUTURE RESEARCH

The technique we developed has a lot of untapped potential. We speculate that there are many additional "supportive" AV products which we have not marked as such due to the simplicity of our tests. Therefore, there are several options for future research, such as:

- Test additional triggers – e.g. packing the Satellite code with a known packer, connecting to malware C&C, etc.
- Test additional exfiltration techniques – e.g. ping (ICMP), HTTPS, SMTP, IRC, connection to popular sites, etc.
- Improve the stealth of the Rocket-Satellite combination by compressing/encrypting the Satellite image inside the Rocket.
- A more robust alternative to AV triggering is to reverse engineer the protocol between the AV agent on the endpoint and the AV cloud or the AV management server, and have the Rocket submit the sample (Satellite) directly, without triggering any AV behavioral rule (and without visual cues to the user, and log artifacts).

CONCLUSIONS

We have demonstrated that cloud-based AV sandbox execution can be used to exfiltrate data from endpoint machines by "burning" the data into the binary to be scanned in the cloud. In this way, even high security enterprises which prevent their endpoints from directly accessing the Internet, or severely restricting Internet access to endpoints, can still have their data exfiltrated. We demonstrated this technique successfully with the latest versions of Avira Antivirus Pro, ESET NOD32 Antivirus, Kaspersky Total Security 2017 and Comodo Client Security. As part of this research we also made some observations on the sandboxes used by these vendors, and commented on the ease of their detection.

We can generalize our findings and state that sharing an executable (suspicious/malicious sample) from the organization, with the outside world in some manner (e.g. submitting the sample to a cloud analysis service or allowing such file submission) can result in data exfiltration, unless there is confidence that the sample has arrived from outside the organization and the file has not changed since its arrival.

ACKNOWLEDGEMENTS

We would like to thank Yoni Fridburg for his help in setting up the AV research lab.

APPENDIX – FINGERPRINTS OF ENCOUNTERED SANDBOXES

Computer name	C02TT36/C02TT22/C02TT26/C02TT18
Class	C-TT (template C02TT)
Used by	Avira
IP	79.207.224.213 (Deutsche Telekom, Germany) / 87.162.248.42 (Deutsche Telekom, Germany)
Performance Counter Frequency	2825742
CPU	Intel64 Family 6 Model 45 Stepping 7, GenuineIntel
Domain	C02TT36/C02TT22/C02TT26/C02TT18
Username	Administrator
MAC	Random
Disk Volume Number	a0e1740e
Additional Software	Python TCL/TK

Computer name	C06TT43
Class	C-TT (template C06TT)
Used by	Avira
IP	87.162.248.42 (Deutsche Telekom, Germany)
Performance Counter Frequency	10000000 (probably virtualized)
CPU	Intel64 Family 6 Model 26 Stepping 5, GenuineIntel
Domain	C06TT43
Username	Administrator
MAC	Random
Disk Volume Number	08266a95
Additional Software	Python TCL/TK

Computer name	WIN7-PC
Class	Something-PC
Used by	Comodo
IP	192.241.99.0/24 (B2 Net, USA)
Performance Counter Frequency	1955556
CPU	Intel64 Family 6 Model 86 Stepping 2, GenuineIntel
Domain	win7-PC
Username	win7
MAC	08:00:27:60:0B:FB (Cadmus) / 08:00:27:63:DC:9F (Cadmus), 08:00:27:2D:9B:64 (Cadmus)
Disk Volume Number	c8e74c10
Additional Software	Python

Computer name	ROGER-PC
Class	Something-PC
Used by	Comodo, Kaspersky, Avira
IP	95.25.4.0/24 (Beeline, Russia)
Performance Counter Frequency	2734726
CPU	Intel64 Family 6 Model 15 Stepping 11, GenuineIntel
Domain	Roger-PC
Username	Roger
MAC	00:07:E9:E4:CE:4D (Intel)
Disk Volume Number	88fdb972
Additional Software	Java

Computer name	0M9P60J5W-PC (and computer names derived from “nDfFAdDKCg4D”, “wdqqwmpPw3rg”, “vvhMq”, “DfCMD”) (differs from ROGER-PC only by perf. freq. counter and username, domain and computer name)
Class	Something-PC
Used by	?
IP	95.25.4.0/24 (Beeline, Russia)
Performance Counter Frequency	2543427
CPU	Intel64 Family 6 Model 15 Stepping 11, GenuineIntel
Domain	0M9P60J5W-PC (same as computer name)
Username	0M9P60J5W (computer name without “-PC”)
MAC	00:07:E9:E4:CE:4D (Intel)
Disk Volume Number	88fdb972
Additional Software	Java

Computer name	PERRY-PC
Class	Something-PC
Used by	?
IP	23.253.228.196 (Rackspace, USA)
Performance Counter Frequency	10000000 (probably virtualized)
CPU	Intel64 Family 6 Model 63 Stepping 2, GenuineIntel
Domain	Perry-PC
Username	Perry
MAC	08:00:27:76:47:D9 (Cadmus) / 08:00:27:43:D3:7A (Cadmus)
Disk Volume Number	44cb6efe
Additional Software	None?

Computer name	DAVID-PC
Class	Something-PC
Used by	Comodo, Avira
IP	66.129.102.52 (Cythereon, USA)
Performance Counter Frequency	14318180
CPU	Intel64 Family 6 Model 45 Stepping 2, GenuineIntel
Domain	David-PC
Username	David
MAC	00:50:56:88:00:BF (VMware) / 00:50:56:88:6C:C2 (VMware)
Disk Volume Number	24b8ccbd
Additional Software	QuickTime

Computer name	ADMIN-PC
Class	Something-PC
Used by	Comodo
IP	193.69.196.112 (Moss Ventelo, Norway), others
Performance Counter Frequency	2724345 / 2683330 / 2719648
CPU	Intel64 Family 6 Model 62 Stepping 4, GenuineIntel
Domain	Admin-PC
Username	Admin
MAC	08:00:27:E5:DF:53 (Cadmus)/ 08:00:27:9D:E7:3A (Cadmus)
Disk Volume Number	0ce74e66
Additional Software	None?

Computer name	APIARY7-PC
Class	Something-PC
Used by	?
IP	130.207.203.2 (GAtech, USA), 165.254.103.0/24 (NTT America, USA)
Performance Counter Frequency	100000000
CPU	Intel64 Family 6 Model 15 Stepping 11, GenuineIntel
Domain	Apiary7-PC
Username	Apiary7
MAC	00:4F:49:22:33:* (unknown vendor)
Disk Volume Number	001fea32
Additional Software	None?

Computer name	ANTONY-PC
Class	Something-PC
Used by	Kaspersky
IP	95.25.4.0/24 (Beeline, Russia)
Performance Counter Frequency	100000000
CPU	Intel64 Family 6 Model 15 Stepping 11, GenuineIntel
Domain	Antony-PC
Username	Antony
MAC	00:FF:F2:F8:30:* (unknown vendor)
Disk Volume Number	c0982f3b
Additional Software	Perl, Python

Computer name	LUSER-PC
Class	Something-PC
Used by	Avira
IP	188.99.236.30 (Vodafone, Germany), 88.71.118.98 (Vodafone, Germany), 88.73.120.59 (Arcor, Germany)
Performance Counter Frequency	1757871
CPU	Intel64 Family 6 Model 28 Stepping 10, GenuineIntel
Domain	luser-PC
Username	luser
MAC	00:25:90:36:66:* (Super Micro), 00:25:90:65:3B:* (Super Micro)
Disk Volume Number	f210a4e5
Additional Software	Strawberry Perl

Computer name	MIKI-PC
Class	Something-PC
Used by	?
IP	180.43.21.173 (NTT, Japan)
Performance Counter Frequency	14318180 (probably virtualized)
CPU	Intel64 Family 6 Model 60 Stepping 3, GenuineIntel
Domain	miki-PC
Username	miki
MAC	00:AA:00:BB:A9:12 (Intel)
Disk Volume Number	6232faa4
Additional Software	Python

Computer name	KLONE_X64-PC
Class	Something-PC
Used by	Comodo
IP	103.208.85.0/24 (OneProvider, Singapore)
Performance Counter Frequency	14318180
CPU	Intel64 Family 6 Model 45 Stepping 7, GenuineIntel
Domain	Klone_x64-PC
Username	admin
MAC	00:50:56:A1:*:* (VMware)
Disk Volume Number	a8727cfb
Additional Software	(note: defines environment variables MD5, SAMPLEALTPATH, SAMPLEPATH)

Computer name	REYNAPC/MALVAPC/ELEANOREPC/WRIGHTPC/BRIAPC/ JORIPC/GABBIPC/HELSAPC/SHARAIPC/MAMEPC/ ARCHONPC/FLORIANPC/EDITHEPC
Class	namePC
Used by	ESET, Comodo
IP	38.90.226.226, 91.228.165.165, 91.228.167.167 (ESET, Slovakia)
Performance Counter Frequency	100000000
CPU	Intel64 Family 6 Model 2 Stepping 3, GenuineIntel
Domain	namePC
Username	Administrator
MAC	00:1B:21:13:37:* (Intel)
Disk Volume Number	e0b2a963
Additional Software	None?

Computer name	Spurtive/circumstellar
Class	spurtive
Used by	Comodo
IP	71.138.0.0/16 (AT&T, USA)
Performance Counter Frequency	3579545
CPU	Intel64 Family 6 Model 47 Stepping 2, GenuineIntel
Domain	w7sb64-01
Username	me
MAC	00:50:56:AA:5F:8A (VMware), 00:0C:29:B1:BC:6B (VMware)
Disk Volume Number	3c2ef4f4/43a26e26
Additional Software	None?

Computer name	ABC-WIN7
Class	ABC-WIN7
Used by	?
IP	207.102.138.40 (Telus, Canada)
Performance Counter Frequency	2127275/2148691
CPU	Intel64 Family 6 Model 45 Stepping 7, GenuineIntel
Domain	ABC-WIN7
Username	abc
MAC	08:00:27:*:*:* (Cadmus)
Disk Volume Number	18ea1f18
Additional Software	Python (also has environment variable PWD)

Computer name	PC
Class	PC
Used by	?
IP	8.40.242.11 (Horizon, USA)
Performance Counter Frequency	100000000
CPU	AMD64 Family 15 Model 6 Stepping 1, AuthenticAMD
Domain	PC
Username	Administrator
MAC	00:50:89:6F:40:BC (Safety Management Systems)
Disk Volume Number	98b68e3c
Additional Software	None?

Computer name	WIN-LRG949QLD21
Class	WIN-LRG949QLD21
Used by	?
IP	95.25.4.233 (Beeline, RU)
Performance Counter Frequency	3320312
CPU	Intel64 Family 6 Model 58 Stepping 9, GenuineIntel
Domain	WIN-LRG949QLD21
Username	[REDACTED]
MAC	00:0C:29:84:A9:47 (VMware)
Disk Volume Number	16074753
Additional Software	None?

Computer name	WIN-U9ELNVPPAD0
Class	WIN-U9ELNVPPAD0
Used by	Avira
IP	87.162.248.42 (Deutsche Telekom, Germany)
Performance Counter Frequency	14318180
CPU	x86 Family 6 Model 26 Stepping 5, GenuineIntel
Domain	WIN-U9ELNVPPAD0
Username	User
MAC	00:0C:29:D1:79:F5 (VMware)
Disk Volume Number	aeb34453
Additional Software	None?

Computer name	PC-4A095E27CB
Class	PC-4A095E27CB
Used by	Comodo
IP	72.12.209.146 (WinTek, USA)
Performance Counter Frequency	2272519/2243720
CPU	Intel64 Family 6 Model 15 Stepping 11, GenuineIntel
Domain	PC-4A095E27CB
Username	STRAZNJICA.GRUBUTT
MAC	00:11:2F:8F:A0:* (ASUSTek)
Disk Volume Number	fc05c743
Additional Software	None?

Computer name	PC-4A095E27CB
Class	ZGXTIQTG8837952
Used by	Comodo
IP	62.102.148.67 (TOR exit node), 167.114.230.104 (RunAbove, France), 108.175.11.230 (1&1, USA)
Performance Counter Frequency	1957382
CPU	Intel64 Family 6 Model 60 Stepping 3, GenuineIntel
Domain	FS02
Username	BLtJc5wjxpj53/8oymz9F1vH9bS/LPRbZ32WM6jf3
MAC	02:00:4C:4F:4F:50 (unknown vendor), 60:02:92:*:*:* (Pegatron)
Disk Volume Number	f0ecfa4a
Additional Software	None? Environment variables: tHH, tMM, tSS, tmpH, tmpM, tmpS, tnow, tnext