vb 2023 LONDON

4 - 6 October, 2023 / London, United Kingdom

WHEN A BOTNET CRIES: DETECTING BOTNET INFECTION CHAINS

Guillaume Couchard & Erwan Chevalier

Sekoia.io, France

guillaume.couchard@sekoia.io
erwan.chevalier@sekoia.io

ABSTRACT

Infection chains used by commodity malware are constantly evolving and use various tricks to bypass security measures and/or user awareness. BumbleBee, QNAPWorm, IcedID and Qakbot are all often used as first-stage malicious code, allowing other more specific payloads to be dropped.

This paper will be in three parts: an overview of the infection chains and common detection methods used against them, an outline of how generic detection rules on these infection chains can help in the fight against botnets, and finally a look at how threat intelligence at scale, combined with the rest, creates a solid defence.

First, we will show our analysis of the evolution in the infection chains of a few of the most common botnets seen in 2022 and early 2023. This study shows how quickly their techniques evolve. It will also cover some detection use cases for these techniques to show how pointless it can be to build overly specific detection rules for these types of threats.

Secondly, we will dig into the creation of more generic rules against known infection chains to detect future threats. Moreover, we will show how these rules can be relevant and more effective than classic detection rules, which are focused on one technique inside an infection chain. These generic rules are based on Sigma correlation, which allows the use of multiple Sigma rules, which will be triggered depending on different criteria, such as time range.

Finally, and as an opening to further discussions, we will present our own threat intelligence and detection pipeline which, thanks to command-and-control server tracking, samples configuration extraction and detonation, allows testing detection rules for non regression, all in a common workflow.

CONTEXT OVER INFECTION CHAINS

For our study we specifically focused on two well-known botnets: Qakbot and IcedID. These two botnets are constantly monitored by security researchers, providing useful data in terms of tactics, techniques and procedures (TTPs), threat actors and related malware.

Qakbot

Qakbot, also known as Qbot, was initially a banking trojan which evolved to a modular malware that has been around since 2008.

The malware is very popular among threat actors, particularly among initial access brokers (IABs). *Prodaft* observed over one million infected victims of Qakbot between February 2022 and February 2023.

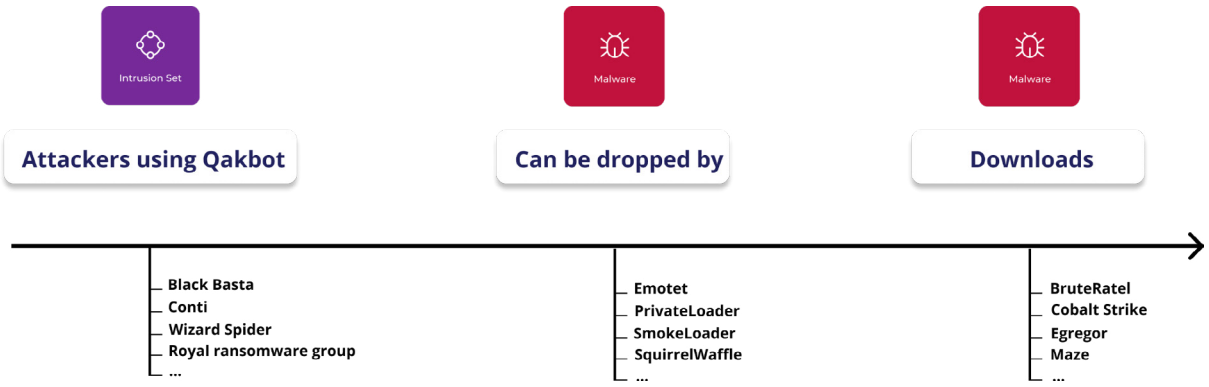

Figure 1: Some known Qakbot relationships.

Qakbot's customers (some of which are listed on the left in Figure 1) are well known in the cybercrime environment, either in the data theft or the ransomware business.

The malware may be dropped by other malware (some Qakbot loaders are listed in the centre of Figure 1) and can be used as a loader for the next payload required in the attack (some examples with very common RAT and ransomware are also listed).

IcedID

IcedID is also a banking trojan; it has been observed in the wild since at least 2017.

Like Qakbot, IcedID is now used by many advanced actors to gain initial access and then deploy another payload, such as ransomware, Cobalt Strike or any other remote access trojan (RAT). Figure 2 shows some known IcedID relationships.

It is a little less widespread than Qakbot, and used by several attack groups, where some also use Qakbot. *Prodaft* observed over 20,000 infected victims of IcedID between August 2022 and December 2022.

Figure 2: Some known IcedID relationships.

Both Qakbot and IcedID have significant impact in terms of victims as observed in related threat groups and malware. Furthermore, besides their feature similarities and their popularity among threat actors, the two botnets share similar infection chains.

The infection chains jungle

The infection chains of Qakbot and IcedID are very similar.

For a better understanding of the paper, we define an infection chain as:

> *The chain of actions that can occur starting from the phishing email with a 'benign' file and ending with the execution of the malicious payload.*

We made a list of the infection chains used by IcedID and Qakbot since early 2022, all of which are presented in the graph trees shown in Figure 3.

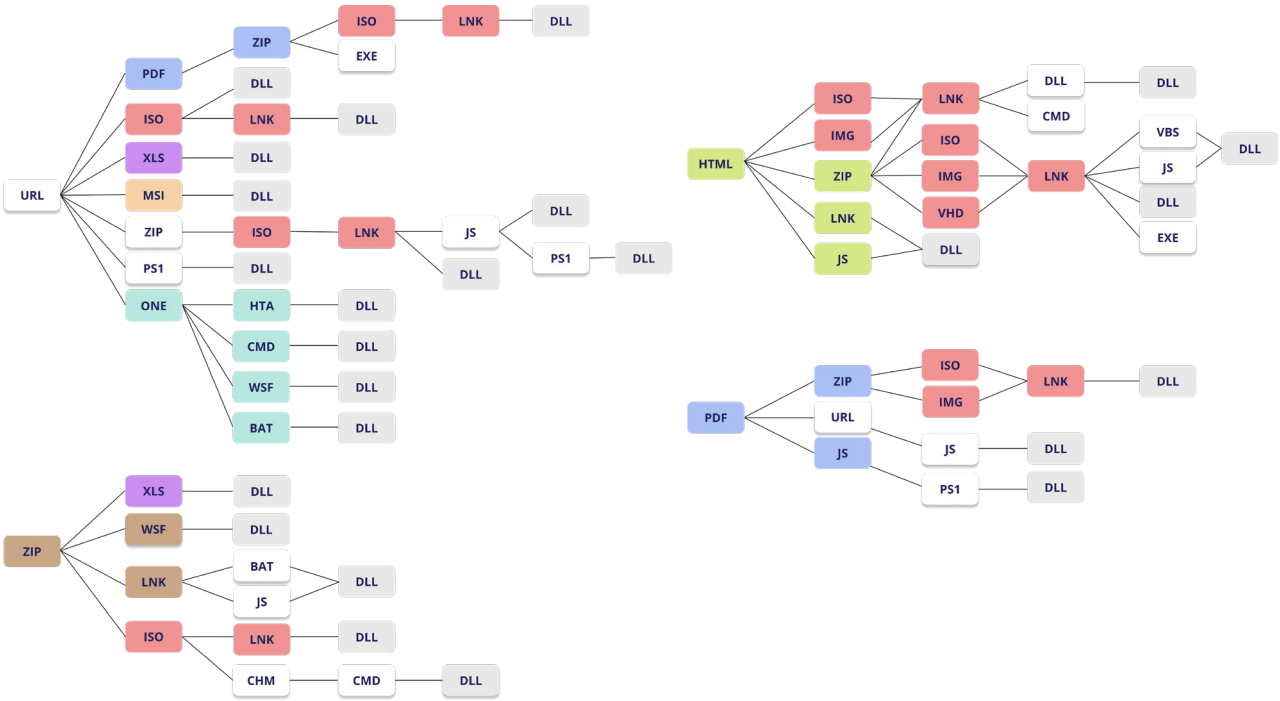

Figure 3: Graph trees of Qakbot and IcedID infection chains, since 2022.

There are many different chains and therefore many different techniques, making it hard for defenders to detect each infection chain.

When looking at this graph, an obvious thought would be to detect the DLL execution, as it occurs at the end of every infection chain. However this is not an easy task – there are many techniques to perform DLL execution, and some are hard to catch with classic logs or security solutions, especially without a high false positive rate.

While looking at other ways to detect most of the infection chains we realized that 'root' infection chains, coloured on these graph trees, might be a good way to detect the different chains.

Are 'root' infection chains worth spending time on?

To answer this question, the timeline below shows the 'coloured blocks'/'root infection chains' (simplified) usage over time since January 2022.

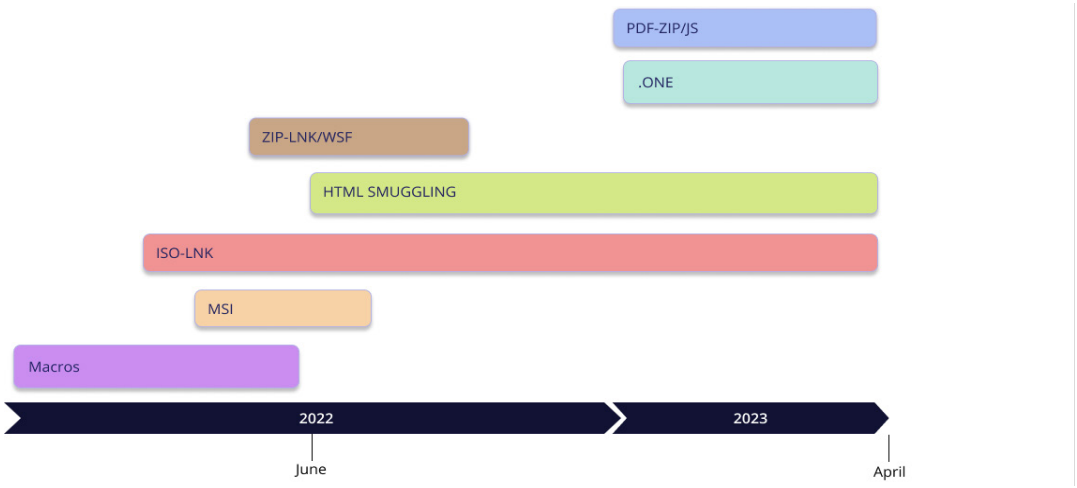

Figure 4: Timeline of Qakbot & IcedID 'root' infection chains since January 2022.

The timeline shows that there are not many different 'root' infection chains and they tend to last over time. Furthermore, even when new ones are used, the old ones often remain in use.

In mid-2022, *Microsoft* announced some new security measures [1], changing the default behaviour of *Office* applications to block macros in files from the internet. This corresponds with the end of the 'macros' infection chain in the timeline.

Prior to that, the ISO-LNK infection chain had started to be used, and is still one of the most commonly used infection chains today, often with HTML smuggling (embedding a file inside an HTML file). Note that ISO-LNK includes the usage of .img and .vhd in the timeline.

In early 2023, *Microsoft* announced some new security measures [2] for *OneNote* (.one) files, blocking users from directly opening an embedded file with a dangerous extension. This change counters most of this infection chain.

Now that we have a bit of context on the botnets and their infection chains, we will focus on the main topic of this paper: detection engineering.

Telemetry and research environment

Before going into the detection rules we present a brief introduction to our workflow for events, because it is important, both in order to build effective detection rules and for the rest of the paper, to understand our detection choices.

The challenge and context is that our customers are quite heterogeneous when it comes to security products. They may use cloud, network or host-related products.

Furthermore, some of them deploy endpoint detection and response agents (we support different ones), others use *Microsoft Windows* collectors, and various network filtering equipment.

Sekoia.io ingests over two billion events per day from European companies with facilities all over the world.

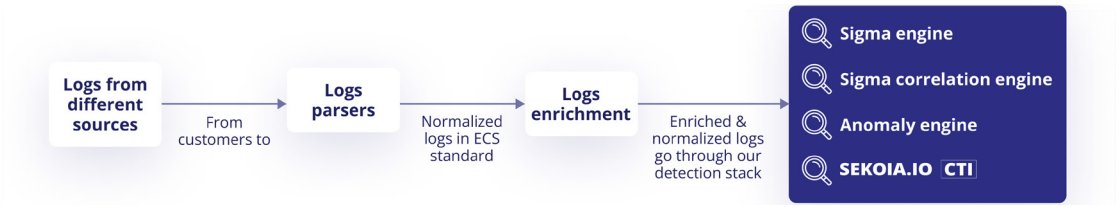

Figure 5: Sekoia.io detection environment.

Using Sigma language on top of ECS (Elastic Common Schema) allows us to use the same detection rule for every customer.

Considering our detection stack, this paper will detail Sigma rules and Sigma correlation rules. Both are slightly different from the original Sigma specification since we use the ECS standard, however the logic remains the same and other languages can apply this logic as well.

Our approach is quite common, we try to have the best possible normalization: ideally, have the same field name and sometimes values across different log formats. When performing correlation, event field name will be used to pivot; as for

other correlation detection languages, the field name or value will be used with aggregation operators (group by, count by) to correlate with other events.

EXISTING SIGMA RULES BENCHMARK

Suspicious Scheduled Task Name as GUID

The first rule we will study as an example is the 'Suspicious Scheduled Task Name as GUID' Sigma rule. While this rule is not specific to Qakbot, according to its creation date and references, the rule was made following *The DFIR Report*'s blog post [3] on Qakbot using Zerologon. In that attack they used scheduled tasks with a GUID as a task name, as in the following example:

```
schtasks.exe /Create /F /TN "{97F2F70B-10D1-4447-A2F3-9B070C86E261}" /TR "cmd /c start /min \"\"
powershell.exe -Command [...]
```

Figure 6: Scheduled task creation command line.

The rule looks like this:

```
detection:
    selection_img:
        Image|endswith: '\schtasks.exe'
        CommandLine|contains: '/Create '
    selection_tn:
        CommandLine|contains:
            # Can start with single or double quote
            - '/TN "{'
            - "/TN '{"
            - "/TN {"
    selection_end:
        CommandLine|contains:
            # Ending of the name to avoid possible FP in the rest of the commandline
            - '}"'
            - "}'"
            - '} '
    condition: all of selection_*
```

Figure 7: 'Suspicious Scheduled Task Name as GUID' Sigma rule.

Therefore it precisely detects the command line observed in Figure 6. The pros and cons of the rule are displayed on a constraint-based radar as shown in Figure 8.

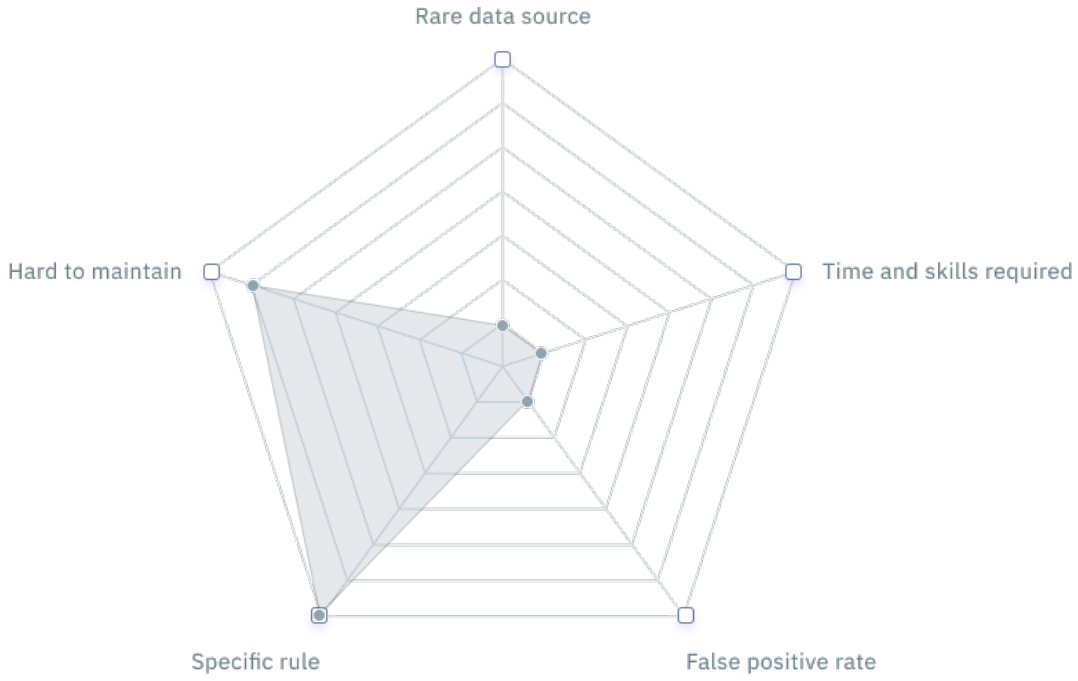

Figure 8: Constraint-based radar for the 'Suspicious Scheduled Task Name as GUID' rule.

First, the rule has many advantages:

- Required data sources are very common: process names and process command lines.
- It is quite easy to make the rule: no specific skill is required and it's easy to test the rule in a lab environment.
- Very low false positive rate (in our environment, at least).

However, there is one main caveat: the rule is very specific to the command line used and therefore can easily be bypassed by other techniques even if they use scheduled tasks as well.

Finally, the rule is very hard to maintain over time, since if another technique uses scheduled tasks but without GUID then a new rule must be created, or a significant update of this rule will be needed. In a large detection engineering team this could cause many issues such as rule regressions or multiple rule updates by different analysts for the same technique, etc.

Suspicious Microsoft OneNote Child Process

The second rule is 'Suspicious Microsoft OneNote Child Process' [4], which is supposed to detect a suspicious *OneNote* child process. We saw Qakbot using a similar infection chain on 7 February 2023.

```
ONENOTE.EXE C:\Users\Admin\AppData\Local\Temp\cancellation.one

cmd.exe /c C:\Users\Admin\AppData\Local\Temp\Open.cmd

powershell Invoke-WebRequest -URI https://nerulgymkhana.com/CCoN/01.gif -OutFile C:\programdata\putty.jpg

rundll32 C:\programdata\putty.jpg,Wind
```

Figure 9: Qakbot OneNote infection chain commands.

This kind of infection chain was highly popular in early 2023 and the Sigma rule existed and was made publicly available before that.

The rule does detect the infection chain shown in Figure 9 since it is quite classic. The attacker embeds a file in the *OneNote* file that will be executed when the user opens it (in this case, the Open.cmd file). The constraint radar for this rule is shown in Figure 10.

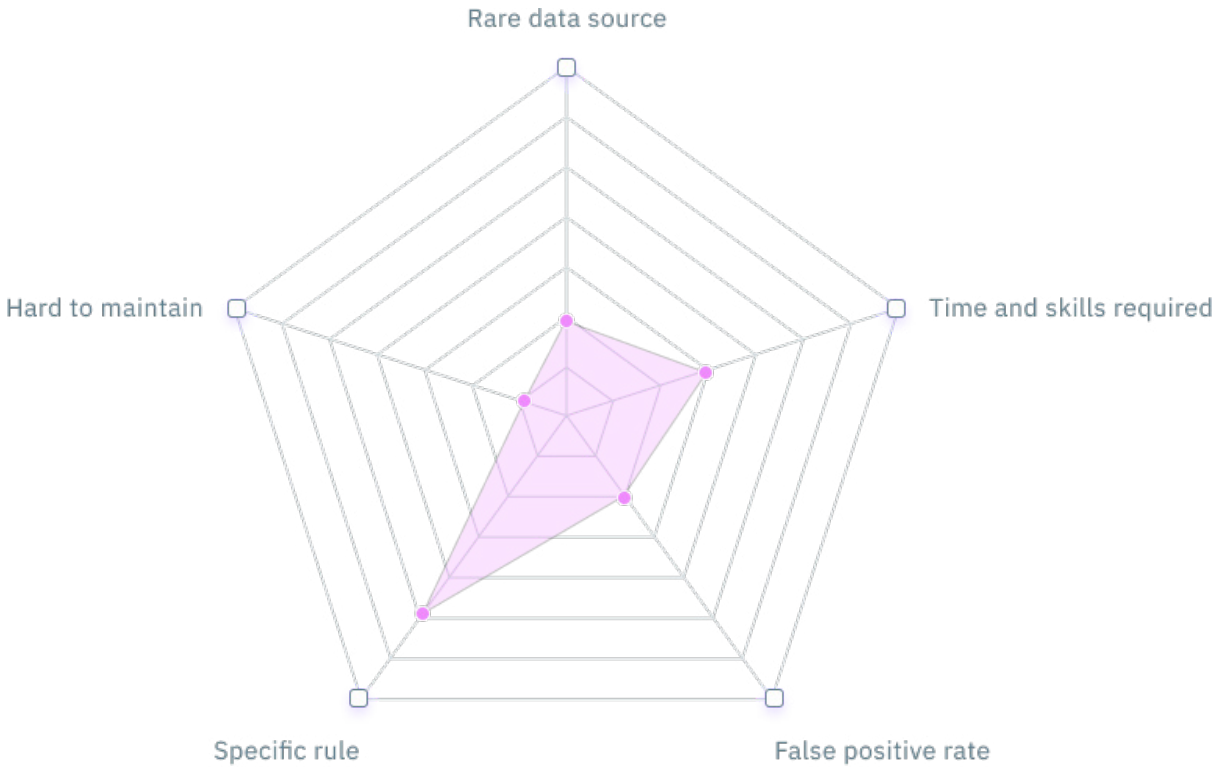

Figure 10: Constraint-based radar for the 'Suspicious Microsoft OneNote Child Process' rule.

This time the radar is a little more balanced around the centre, which is what we aim for. The rule has many advantages:

- Required data sources are common: parent-child process relationships and process command lines.
- The rule is very easy to test.
- Only two false positives were observed in our environment in 30 days.
- The rule is easy to update because it covers the overall *OneNote* infection chain. Therefore, if a change is observed in a *OneNote* infection chain, analysts will know that this is the rule that needs to be changed. This makes the rule regression fairly easy to test.

There are two main drawbacks for this rule: it is specific to *OneNote* infection chains and it is based on a list of processes, file extensions and file paths. Therefore it might be easily bypassed – although we haven't seen a bypass from the *OneNote* infection chains used by Qakbot and IcedID.

Overall the rule is great, but such a rule on an infection chain is mainly possible because of how *OneNote* operates as it embeds the 'next-stage' payload, allowing OneNote.exe to be the parent process and helpfully listing suspicious children processes.

Unfortunately it won't be that 'easy' with other infection chains, which is why Sigma correlation is helpful.

Enhancing detection using Sigma correlation

Sigma correlation [5] is fairly new, but useful in detection engineering. It allows correlation between 'classic' Sigma rules, which creates Sigma correlation rules. Furthermore, it also allows chaining of those rules, giving multiple possibilities.

ISO-LNK infection chain detection using Sigma correlation rule

In our public *GitHub* repository [6] we share a Sigma correlation rule to detect the ISO-LNK infection chain. The rule is summed up in the schema shown in Figure 11.

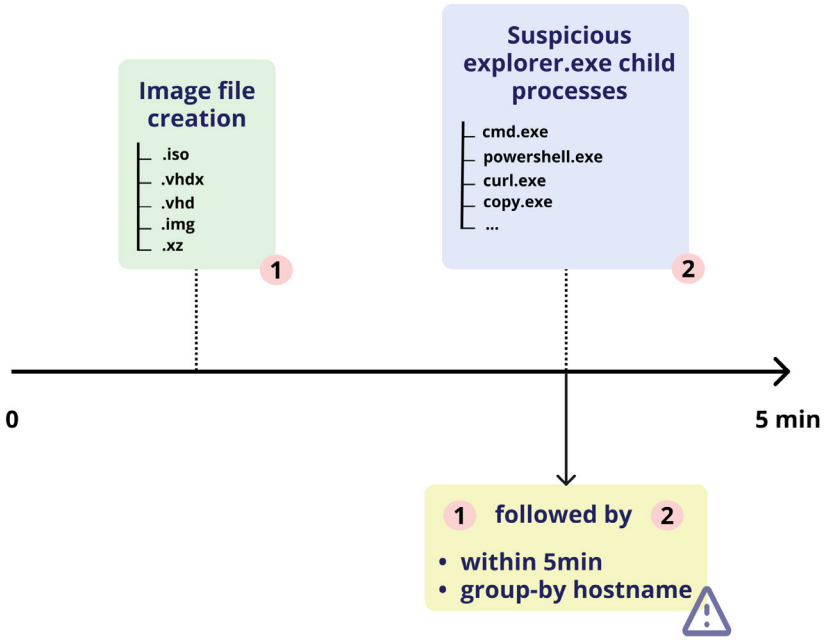

Figure 11: ISO-LNK Sigma correlation rule schema.

It correlates two classic subrules which are classic Sigma rules:

- The first subrule detects the creation of files with suspicious extensions: archive format files from ISO to XZ.
- The second subrule detects a suspicious process name (a truncated list of over 60 common *Windows* processes) with the parent process 'explorer.exe'.
- If the first subrule is followed by the second subrule within five minutes on the same host, it will raise an alert.

Why focus on explorer.exe child processes?

The user simply opens image files with explorer.exe and will click on the LNK in *Explorer* as well, triggering the malicious processes with explorer.exe as the parent.

Still, a process name 'list' can always be bypassed. Too many false positives were observed without the list, which is why we had to find the usual balance of quality versus false positive rate.

The rule does detect the infamous ISO-LNK infection chain.

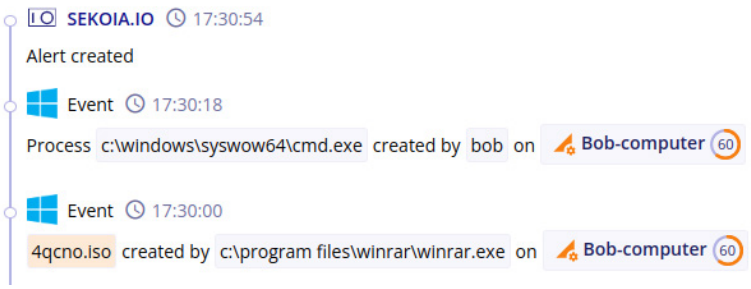

Figure 12: Events timeline from an ISO-LNK alert.

The 'cmd.exe' process in Figure 12 is executed from explorer.exe (by clicking on a created LNK file) to launch a malicious CMD file, as shown in Figure 13.

process.parent.name	explorer.exe
process.name	cmd.exe
process.command_line	c:\windows\syswow64\cmd.exe /q /c vibrations\\polaroid.cmd

Figure 13: ISO-LNK process tree extract.

We merged the constraint radar of this rule and the following one since they were quite similar.

ISO-LNK was one of the two most commonly used infection chains in the past year and it was worth focusing detection efforts on. However, there was another that was commonly used: HTML smuggling.

HTML smuggling infection chain

We also shared this rule in our public *GitHub* repository [7]. This rule is more complex and is explained in the schema in Figure 14.

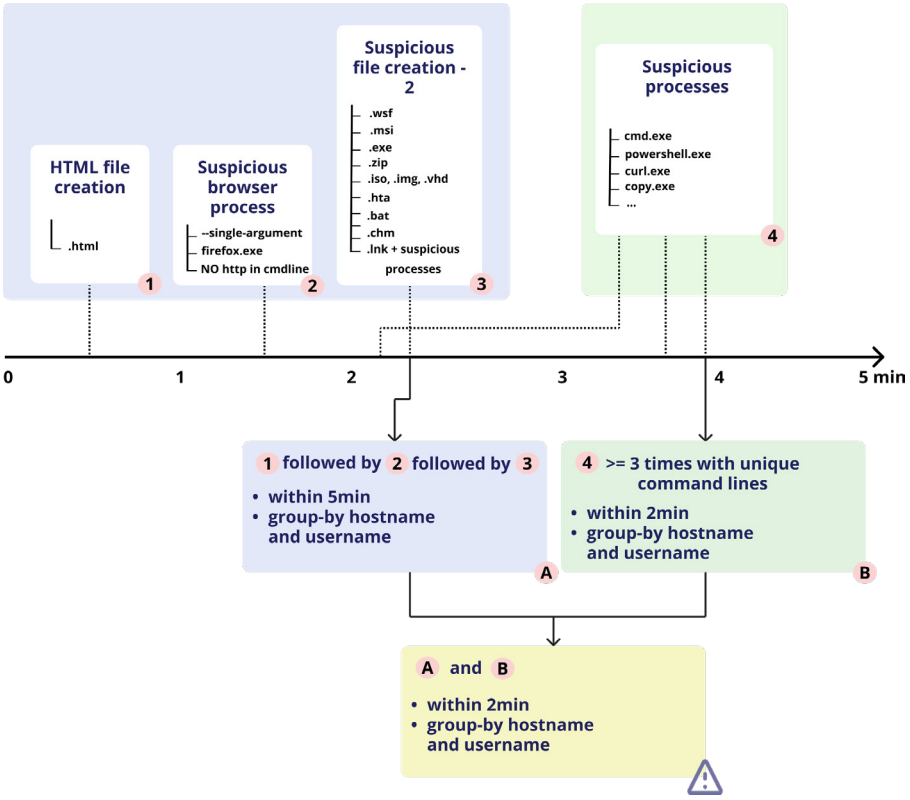

Figure 14: HTML smuggling Sigma correlation rule schema.

The rule is divided into two correlation rules chained together.

The first correlation rule (shown with blue background in Figure 14) detects .html, .pdf or .zip file creation followed by a browser opening a file locally (no http://) and followed again by suspicious file creation based on the file extension (.iso, .hta, etc.). LNK is the exception where it needs to be combined with a list of processes to avoid having too many events.

This is a time-based correlation rule that needs all of the above to happen within five minutes. This time frame was tricky to set – on the one hand, some amount of time is needed since the user has to do everything manually. For instance, in a common HTML smuggling infection chain the user needs to double click on the HTML file to open it, retrieve the ZIP archive with the password contained in the HTML file, decompress the archive and then double click on the malicious file inside. Therefore the process can clearly last longer than five minutes, but we found that too long a time frame leads to many false positives.

The second correlation rule (shown with green background in Figure 14**)** detects multiple (at least three) suspicious processes being spawned within two minutes.

The two correlation rules are then chained together, and if we spot the two of them in a timespan of two minutes, then an alert is raised, as shown in Figure 15.

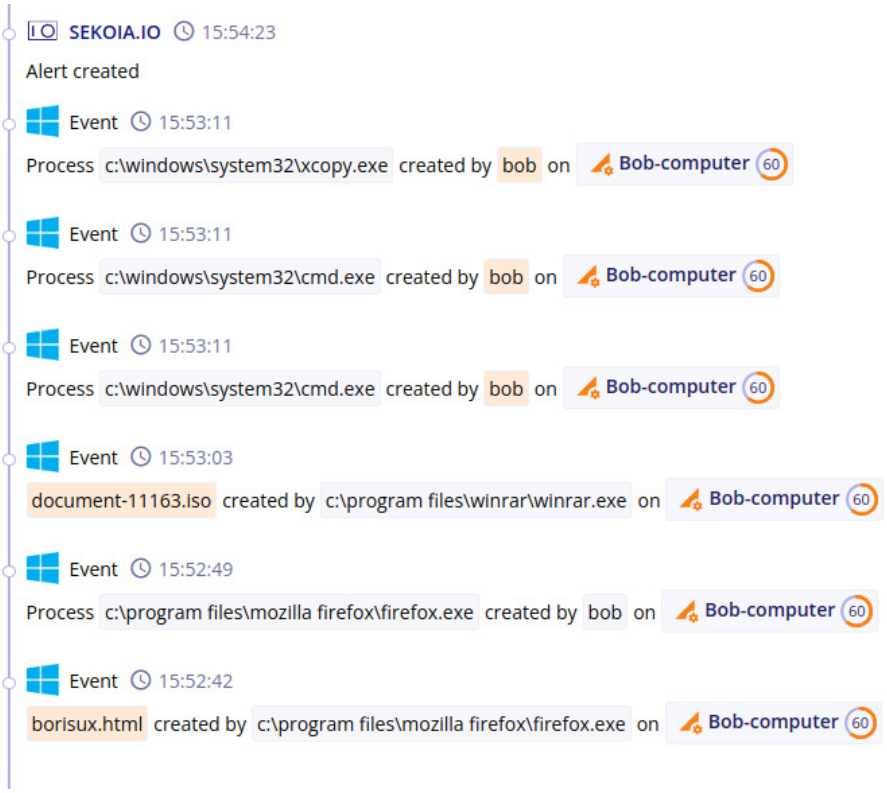

Figure 15: Events timeline from an HTML smuggling alert.

Our constraint radar, shown in Figure 16, is the same for both correlation rules since they are quite similar.

The radar is nicely balanced around the centre, mainly for the following reasons:

- The rules use common data sources with the exception of the files created that are not available by default on *Windows* and often require Sysmon or a specific agent. Hence why the radar is not so close to the centre there.
- The rules are not hard to build, however the time required is quite significant since it requires several tests to spot false positives and filter them out.
- The false positive rate observed in our environment was low but highly dependent on the environment and activity of the company, especially for the HTML smuggling rule. More specifically, we observed 0 false positives between 9 May 2023 and 9 June 2023 for the ISO-LNK correlation rule. However, we observed around 100 false positives for the HTML smuggling rule – most of these were concentrated on a few customers with specific environments.
- The rules are specific to ISO-LNK and HTML smuggling, but they cover both techniques and therefore cover a large scope of infection chains. Nevertheless, lists of processes and file extensions are relied upon, which can be bypassed.
- The rules are easy to maintain, since once a bypass is found in an infection chain it is easy to find which rule to update. It is also easy for non-regression tests as there are not many different rules for the same infection chain.

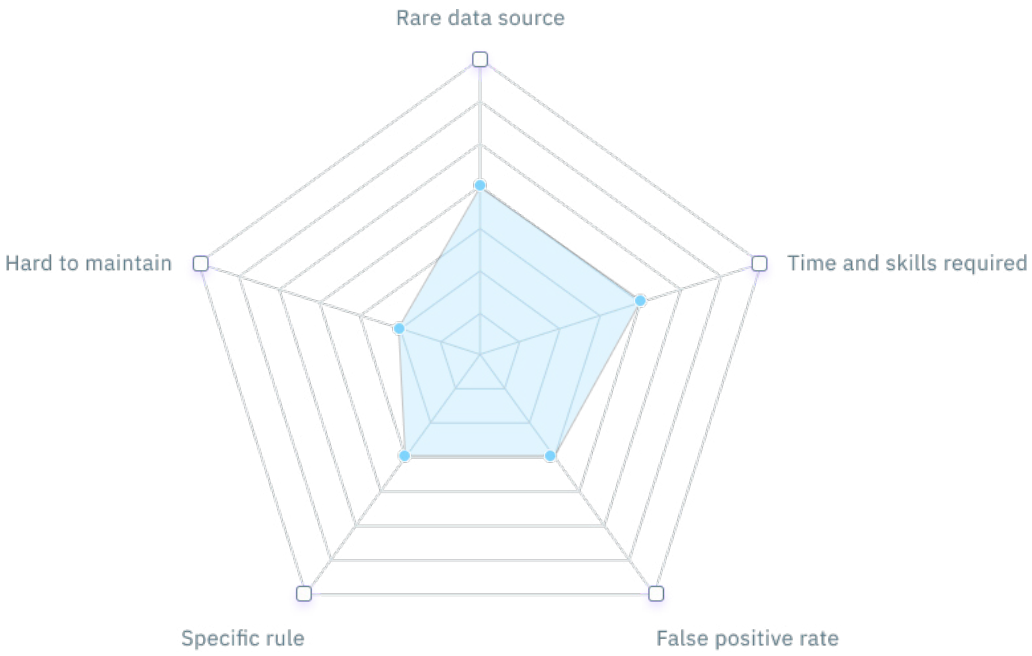

Figure 16: Constraint-based radar for both Sigma correlation rules.

Rules benchmarking using constraint-radar

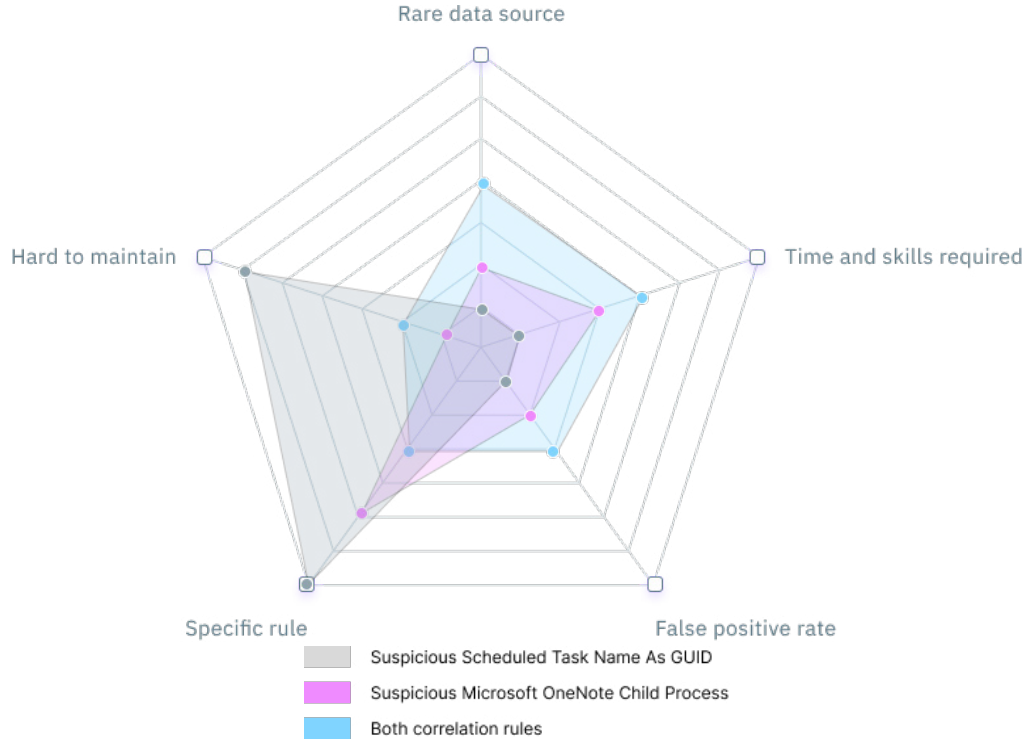

Figure 17: Constraint-based radar for studied Sigma rules.

To conclude, all four rules have pros and cons, but some are more balanced than others.

The rule titled 'Suspicious Scheduled Task Name As GUID' is a great quick win, but extremely specific and hard to maintain over time. The OneNote rule is really good, even if it is focused on *OneNote*. Finally, the correlation rules are the most balanced in terms of efficiency versus complexity and false positives.

THREAT INTELLIGENCE & DETECTION PIPELINE

In the previous part we focused on detection rules from a host perspective, however we also need to detect these threats from a network perspective.

C2 trackers

To detect the threats from a network perspective we mainly rely on indicators of compromise (IOCs), which are retrieved via an internal project called 'C2 trackers' to monitor attackers' infrastructure. We use heuristics to dig into data from various services scanning the internet. While Suricata or Snort-like rules could be a great way to detect the threats from a network perspective, we are unable to perform that for the moment.

A classic example of a Qakbot heuristic with its default TLS certificate on C2 server details is presented below:


```
"cipher_selected": "TLS_CHACHA20_POLY1305_SHA256",
"certificates": {                                          1
  "_encoding": {
    "leaf_fp_sha_256": "DISPLAY_HEX"
  },
  "leaf_fp_sha_256": "2f4055d179d0dbca38dcf7473ffbbc00558333bcb3d067d489504878bcc87972",
  "leaf_data": {
    "names": [
      "tqcs.biz"
    ],
    "subject_dn": "C=AU, OU=Ekejyrjli Ivbtdgu Ogovau, CN=tqcs.biz",
    "issuer_dn": "C=AU, ST=GI, L=Xuiraioi, O=Paevjwyc Xmo Fbkfiodak, CN=tqcs.biz",
    "pubkey_bit_size": 2048,
    "pubkey_algorithm": "RSA",
    "tbs_fingerprint": "5dba93dfd21571b4048448121b1fd2704f186026796df96182f5669c6678de3c",
    "fingerprint": "2f4055d179d0dbca38dcf7473ffbbc00558333bcb3d067d489504878bcc87972",
    "issuer": {
      "common_name": [
        "tqcs.biz"
      ],
      "locality": [
        "Xuiraioi"
      ],                                                   2
      "organization": [
        "Paevjwyc Xmo Fbkfiodak"
      ],                                                   3
      "province": [
        "GI"
      ],                                                   4
      "country": [
        "AU"
      ]                                                    5
    },
    "subject": {
      "common_name": [
        "tqcs.biz"
      ],                                                   6
      "organizational_unit": [
        "Ekejyrjli Ivbtdgu Ogovau"
      ],
      "country": [
        "AU"
```


Figure 18: Qakbot C2 default certificate heuristic.

Every entry from 1 to 6 has specific values that could be searched using regexes or exact matches. We have been able to find approximately 30 C2 servers each month since October 2022 using this heuristic.

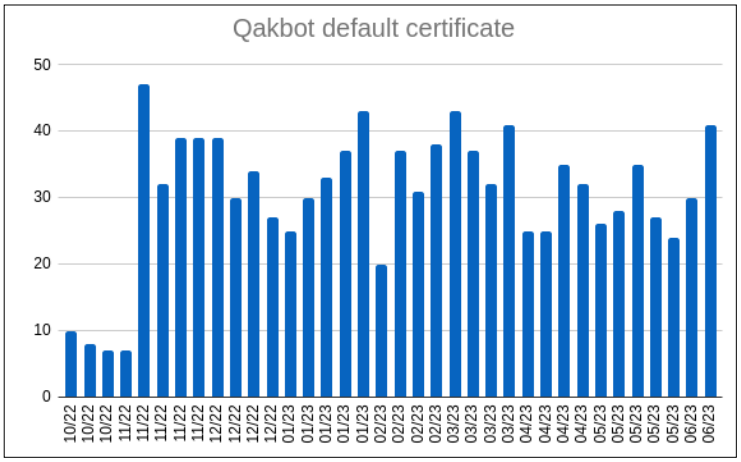

Figure 19: Qakbot C2 default certificate heuristic servers results.

Malware detection pipeline

To check if our rules still match the infection chains used, but also to test them for non-regression purposes, we built a malware detection pipeline. This pipeline also has the advantage of allowing us to retrieve malicious indicators (hashes and C2 network infrastructure), updating our CTI.

Figure 20: Malware detection pipeline.

First, we retrieve malware samples using different sets of YARA rules. These samples are then pushed into an orchestration tool (FAME at the time of writing) to execute several modules on them. These modules can be used to extract malware configurations directly if possible, but the role of one of the modules is to send the samples into different sandboxes, both internal and external ones.

Our internal sandbox sends the logs to our platform, allowing us to test if our detection rules raise the alerts they are supposed to. That way, we can avoid rule regression and even observe if a new infection chain has emerged.

The difficulty here is that some infection chains, like the ones described in this paper, are quite complex. For instance, Qakbot infection chains often need a password that's embedded in a file to open another one. Therefore, the execution of these infection chains is not fully automated (yet?).

All this process also allows us to check if the C2 indicators retrieved are still active, and potentially allows us to get new C2s, therefore improving our trackers' heuristics.

CONCLUSION

Detection engineering at scale allows us to confirm that there is no such thing as a magic rule. A lot of work is needed to build efficient detection rules and ultimately have zero regression and defend against attackers that are more and more advanced.

In order to describe what we observed in terms of detection of infection chains, we have re-used the concept of the pyramid of pain of the cyber threat intelligence field:

- From trivial to simple: it is easy for an attacker to change, but also easy for defenders to detect and often possible with classic Sigma rules.
- From annoying to tough: it is more complex for attackers and defenders. It usually requires correlation rules to have good coverage.

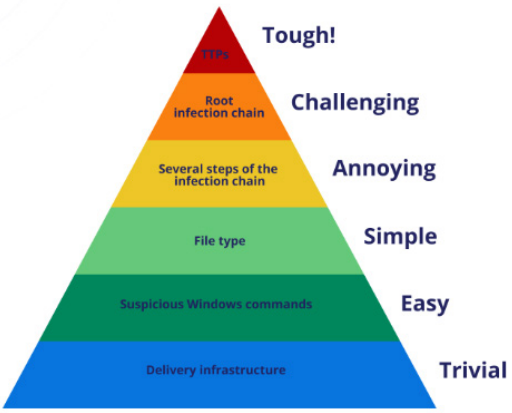

Figure 21: Infection chains Pyramid of Pain.

ACKNOWLEDGEMENTS

We thank the malware hunters pr0xylife [8] and malware-traffic-analysis [9], and acknowledge the use of the *Recorded Future Triage* sandbox [10], which made our work easier.

REFERENCES

[1] Microsoft. Macros from the internet will be blocked by default in Office. https://learn.microsoft.com/en-us/deployoffice/security/internet-macros-blocked.

[2] Microsoft. OneNote blocks embedded files that have dangerous extensions. https://learn.microsoft.com/en-us/deployoffice/security/onenote-extension-block.

[3] The DFIR Report. Qbot and Zerologon Lead To Full Domain Compromise. 21 February 2022. https://thedfirreport.com/2022/02/21/qbot-and-zerologon-lead-to-full-domain-compromise/.

[4] https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_office_onenote_susp_child_processes.yml.

[5] https://github.com/SigmaHQ/sigma-specification.

[6] https://github.com/SEKOIA-IO/Community/blob/main/sigma_rules/host/correlation_iso-lnk_infection_chain.yml.

[7] https://github.com/SEKOIA-IO/Community/blob/main/sigma_rules/host/correlation_html_smuggling.yml.

[8] pr0xylife. https://github.com/pr0xylife.

[9] malware-traffic-analysis. https://www.malware-traffic-analysis.net/.

[10] Recorded Future Triage. https://tria.ge/.

[11] Guillaume, C.; Chevalier, E. XDR detection engineering at scale: crafting detection rules for SecOps efficiency. SEKOIA.IO. 3 October 2022. https://blog.sekoia.io/xdr-detection-rules-at-scale/.

[12] https://github.com/SEKOIA-IO/Community/tree/main/sigma_rules/.