2024 DUBLIN

2 - 4 October, 2024 / Dublin, Ireland

UNVEILING SHADOWS: KEY TACTICS FOR TRACKING CYBER THREAT ACTORS, ATTRIBUTION, AND INFRASTRUCTURE ANALYSIS

Hossein Jazi

Fortinet, Canada

hhadianjazi@fortinet.com

ABSTRACT

In the complex and ever-evolving landscape of cybersecurity, advanced persistent threats (APTs) represent a significant challenge due to their sophisticated and covert nature. These threats, often state-sponsored or highly organized criminal activities, persistently target specific entities with the intent to covertly infiltrate and linger within target networks to achieve espionage, data theft or sabotage objectives.

This paper delves into the cutting-edge methodologies and technologies that cybersecurity professionals employ to track, attribute, and dismantle APTs. We will explore the intricate process of identifying and analysing the digital footprints left behind by these attackers, which involves a meticulous examination of malware signatures, attack vectors and communication patterns. We will cover various tracking techniques, especially pivotal methods for infrastructure tracking, which enable defenders to navigate through the maze of servers, domains and other resources used by adversaries, thereby uncovering the full extent of the threat landscape.

Throughout this paper, we leverage real-world examples to illustrate how practitioners can effectively track and monitor cyber threat actors. These case studies highlight successful applications of digital forensics and cyber intelligence techniques in exposing APTs, demonstrating the practical implications of theoretical strategies.

TRACKING APTS

Figure 1 illustrates the main structure for tracking cyber threat actors. The primary focus of this paper is tracking APTs, but it is important to note that tracking provides the foundational basis for attribution and offers highly valuable artifacts to attribute the activity to specific threat actors.

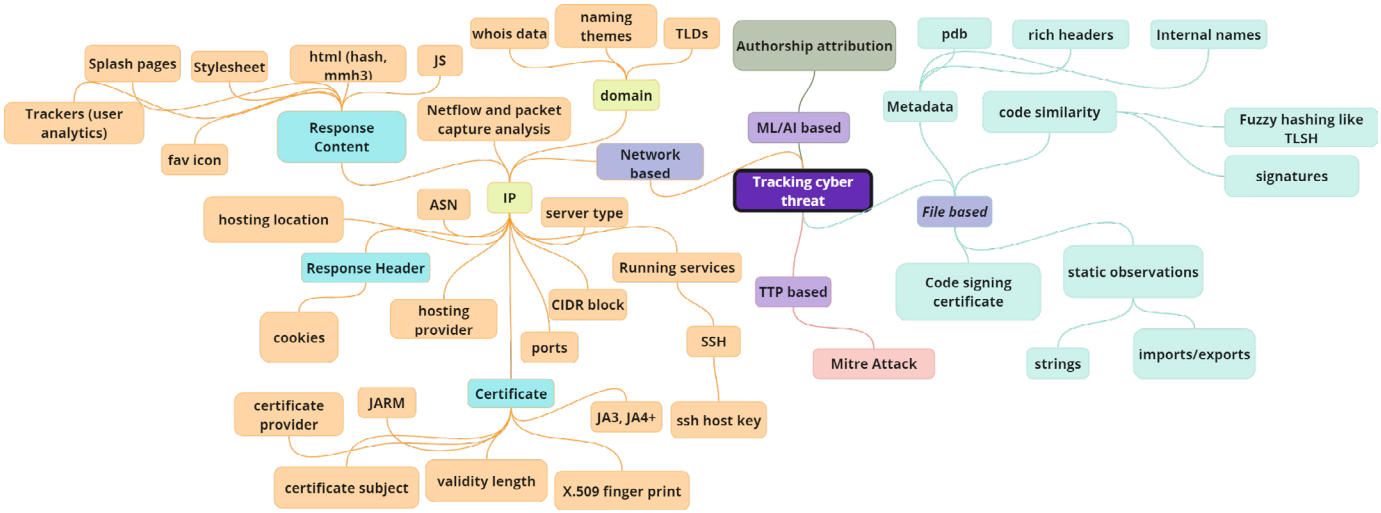

Figure 1: Tracking methods.

As Figure 1 shows, we have categorized our tracking methods into four main categories: file-based, network-based, TTP-based (behaviour-based), and ML/AI-based.

- **File-based tracking**: This category focuses on tracking cyber threat actors based on the toolsets and malware families they use. It specifically involves extracting static indicators from the threat actor toolsets to track and identify new campaigns associated with the same threat actor.
- **TTP-based (behaviour-based) tracking**: This category focuses on the tactics, techniques and procedures (TTPs) employed by a threat actor and uses them for tracking and attribution. It emphasizes the behaviour of the attack for tracking purposes.
- **Network-based tracking**: This category focuses on tracking threat actors based on the infrastructure they use. Cyber threat actors often configure their infrastructures in specific ways, leaving valuable information that analysts can use to track them.
- **ML/AI-based tracking**: This category uses machine learning and artificial intelligence methods for tracking and attribution. It mainly involves using machine-learning models for authorship attribution. We will not focus on this category in this paper.

File-based tracking

Threat actors tend to reuse specific parts of code or use particular libraries, which can be useful for tracking them. We categorize these tracking methods into three main categories: static indicators, code similarity and metadata.

Static indicators

Static indicators are indicators extracted from the static analysis of a toolset or malware used by the threat actor. For example, specific strings can be used to track and identify new variants of malware associated with a threat actor.

An example of this category comes from our research on the Dacls RAT malware used by the Lazarus threat actor. This malware had both *Windows* and *Linux* variants. In a research report, the researchers identified two unique strings, 'c_2910.cls' and 'k_3872.cls', which are the names of certificate and private key files previously observed.

By pivoting from these unique strings, we were able to identify and track the activities of this malware family, uncovering that the threat actor had started to use a *MacOS* variant [1] of the Dacls RAT malware in their campaigns.

The same pattern can be applied to the imports and exports used in malware. By examining specific export names or imports, you can uncover new malware families used by the same threat actor.

For imports, imphashing can be used to track malware families. Tracking malware using imphashing involves generating a hash value from the import address table (IAT) of an executable file, which lists functions that the executable imports from dynamic link libraries (DLLs). By extracting the IAT from different malware samples and creating their imphashes, analysts can compare these hash values. Identical or similar imphashes suggest that the samples are related or variants of the same malware family. This method is particularly useful for identifying and tracking new variants of known malware, as threat actors often reuse code and import functions in their malicious software.

As an example, you can use imphash to find different variants of the LocalPotato NTLM exploitation tool used by FIN11 [2] for privilege escalation. This payload exploits CVE-2023-21746 to escalate its privileges on the victim's machine.

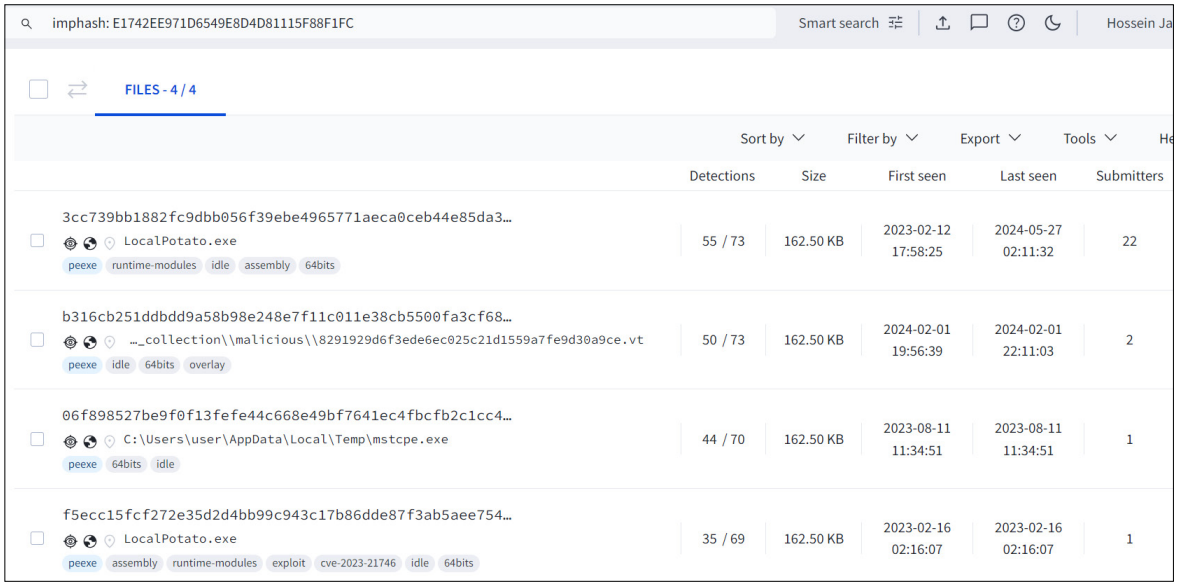

Figure 2: Using imphash to find variants of LocalPotato.

Code similarity

Code similarity analysis is a critical technique used in malware and APT tracking. This approach involves comparing the code of different malware samples to identify similarities and commonalities, which can help in understanding the relationships between different pieces of malware and attributing them to specific threat actors or groups. Key concepts in code similarity analysis include code reuse, where malware authors frequently reuse code snippets or entire modules, which can link different malware samples to the same author or group. Analysing the code structure, such as control flow graphs and function calls, reveals similarities between different malware samples that may not be apparent from a superficial examination. Additionally, observing behavioural patterns, such as the use of specific techniques for persistence, evasion or exploitation, can help identify a common origin or shared development practices among different malware variants.

In some cases, we observed that APTs tend to reuse encryption keys, encoding and encryption algorithms, API hashing methods, and command-and-control communications. These elements are crucial for tracking and attributing malware families. Typically, advanced reverse engineering skills are required to analyse the malware, identify unique patterns, and then use those patterns in YARA rules to track the APT group.

For instance, we analysed a case involving the Lazarus APT [3], where the attackers used a custom encryption algorithm. We translated this algorithm into a YARA rule, allowing us to identify the same encryption algorithm used in the BISTROMATH RAT associated with the same group. Using this rule, we were also able to uncover all the payloads used in that campaign. This approach demonstrates how identifying and leveraging unique code patterns can enhance malware tracking and attribution efforts, providing valuable insights into APT operations.

```
signed __int64 __fastcall String_decoder(__int64 a1, __int64 a2, __int64 a3)
{
  __int64 v3; // r10
  char v4; // r11
  signed __int64 result; // rax
  unsigned int v6; // er9
  __int64 v7; // rbx
  char v8; // cl

  a3 = (signed int)a3;
  v3 = a2;
  v4 = -124;
  result = 1461817411164;
  v6 = 162112194;
  if ( (signed int)a3 > 0164 )
  {
    v7 = a1 - a2;
    do
    {
      v8 = *(_BYTE *)(v7 + v3++);
      *(_BYTE *)(v3 - 1) = v4 ^ result ^ v6 ^ v8;
      v4 = v4 & result ^ v6 & (v4 ^ result);
      v6 = (v6 >> 8) | ((((unsigned __int16)v6 ^ (unsigned __int16)(8 * v6)) & 0x7F8) << 20);
      result = ((unsigned int)result >> 8) | ((((_DWORD)result << 7) ^ ((unsigned int)result ^ 16
                                                                        * ((unsigned int)result ^ 2 * (_DWORD)result)) & 0xFFFFFF80) << 17);
      --a3;
    }
    while ( a3 );
  }
  return result;
}
```

Figure 3: Custom encryption algorithm used by Lazarus APT.

Fuzzy hashing

Fuzzy hashing is a technique used in digital forensics and cybersecurity to identify similarities between different pieces of data, such as files or strings, even when they are not identical. Unlike traditional cryptographic hashing, which produces a completely different hash for even the slightest change in input, fuzzy hashing generates a hash that reflects the degree of similarity between the inputs. This makes it particularly useful for detecting variations of malicious files and tracking the evolution of malware. Various fuzzy hashing algorithms, including ssdeep, sdhash, and TLSH (Trend Micro Locality Sensitive Hash), have been developed to address different needs within cybersecurity.

Ssdeep is one of the earliest and most widely used fuzzy hashing algorithms, known for its ability to detect similarities in spam emails and malware. Sdhash creates a similarity digest based on the statistical features of data blocks, making it useful in digital forensics for comparing files and identifying fragments within larger datasets. TLSH, on the other hand, is designed to be robust against minor changes and calculates a hash value indicating the similarity between files, making it effective for clustering similar malware samples and identifying new variants in large-scale malware databases. For APT tracking, fuzzy hashing helps identify common tools used by APTs across different incidents, linking activities to specific threat actors. By comparing hashes of files and scripts used in different attacks, analysts can track the infrastructure used by APTs, identifying common servers and resources.

We mainly use TLSH to find similarities between malware variants. In one case, we found an HTML file (1.xhtml 406a09578b07415880b035cb8afd688465ffd28a9c7c46680987295ce50d8840). Initially, we were unsure about the actor behind it, but the infrastructure used suggested a possible connection to NetSupport RAT. Using TLSH, we identified 21 related HTML files. Upon investigating these files, we noticed that some were known to be used by the Gamaredon APT. This finding allowed us to continue our investigation and confirm that the new HTML file was also associated with Gamaredon APT. Additionally, it helped us discover HTML files that had not been reported before.

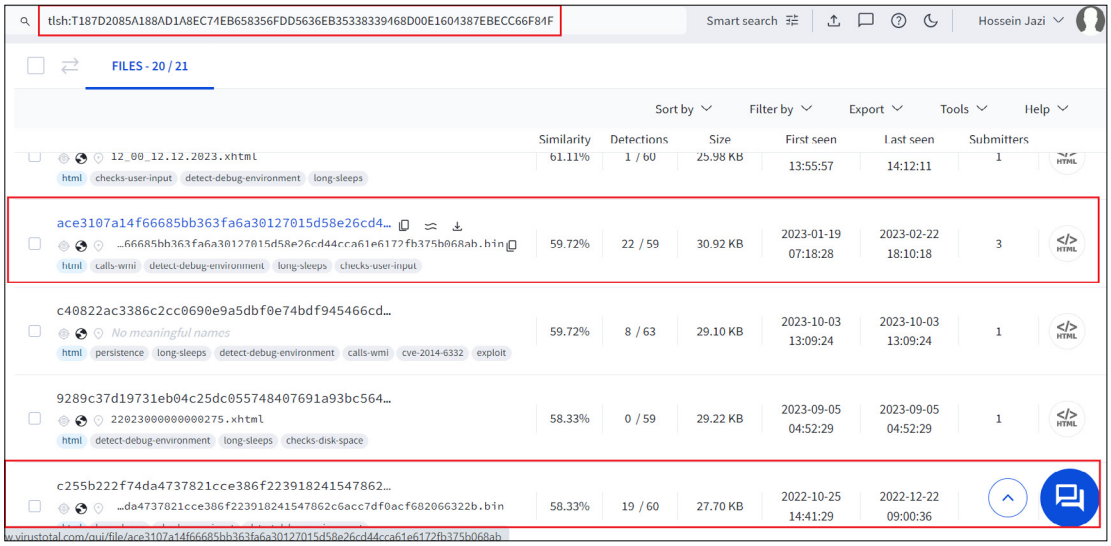

Figure 4: Use of TLSH to identify related files.

Signature

As we briefly mentioned in the code similarity section, signatures play an important role in tracking malware families associated with threat actors. The most popular signatures for tracking APT samples are YARA rules.

YARA (Yet Another Ridiculous Acronym) rules are a powerful tool in malware analysis and threat hunting, and are used to identify and classify malware samples based on textual or binary patterns. They enable analysts to define specific patterns that match strings, byte sequences, or other characteristics within files, using plain text, hexadecimal strings, regular expressions, and wildcards. YARA rules can also incorporate complex conditions and Boolean logic for precise targeting. This modular and reusable nature makes them effective for detecting known malware, tracking APT groups, threat hunting, and aiding in incident response.

For example, analysts create YARA rules to detect unique encryption algorithms or code reuse by APT groups like Lazarus. By targeting specific characteristics of malware files, such as API hashing methods or unique binary sequences, YARA rules link new malware samples to known threat actors. This method helps in identifying variants of known malware and linking them to specific threat actors, enhancing our ability to detect and respond to threats quickly and accurately.

Here is an example of a YARA rule we developed to detect a variant of malware associated with Turla APT:

```
rule SFX_TUR_AND_KOPILUWAK
{
  meta:
    author = "H2J"
    description = "Strings contained in SFX"


  strings:
        $str1 = "Setup=c:\\windows\\system32\\wscript.exe /b"
        $str2 = "Silent=1"
  condition:
      all of them

}
```

Figure 5: YARA rule used to detect a Turla malware variant.

Aside from YARA rules, antivirus (AV) signatures can also help track malware families associated with APTs. Some companies develop robust signatures specifically designed to detect malware families linked to APTs.

For example, in a recent attack campaign associated with the Lazarus APT, the BeaverTail [4] malware was used to target victims. Using AV signatures, new samples associated with this campaign can be identified effectively.

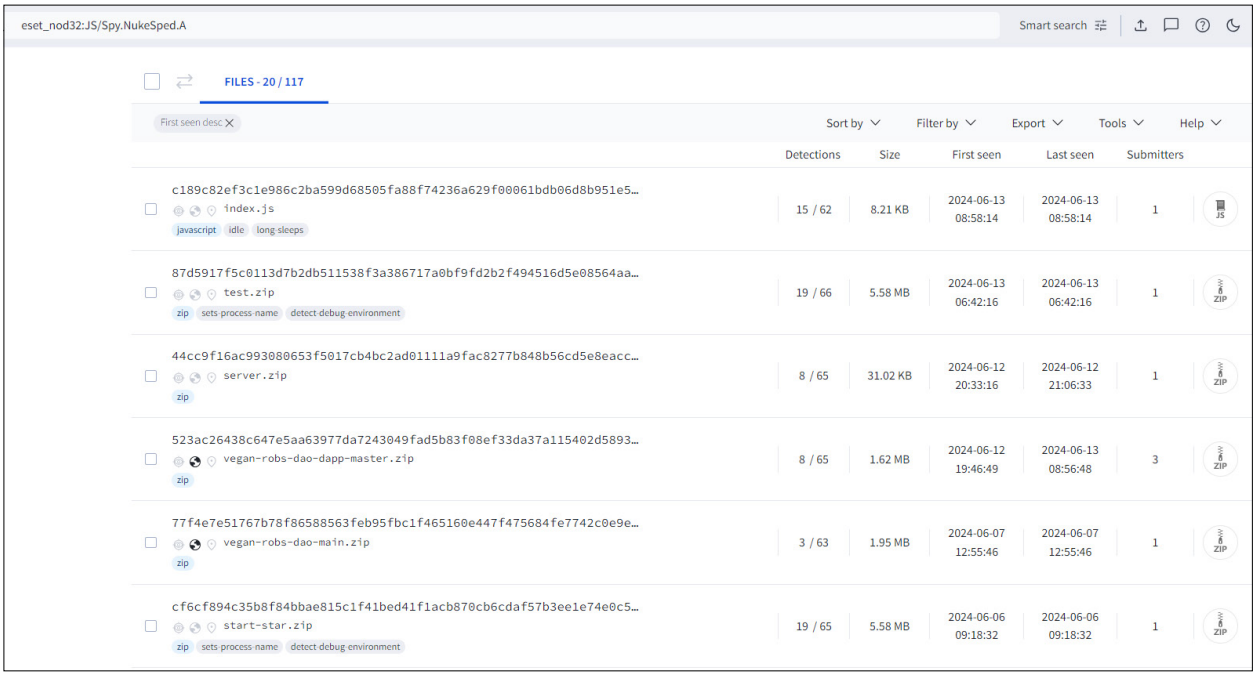

Figure 6: AV signatures used to identify new samples.

Metadata

Metadata in malware refers to various types of auxiliary information embedded within the malware files that can be used for tracking, analysing, and attributing malicious activities. The metadata can provide significant insights into the malware's origin, authorship and behaviour, aiding cybersecurity professionals in their efforts to defend against threats. Key aspects of metadata for tracking malware include file metadata and compiler metadata, each offering different types of information for analysis.

File metadata encompasses details such as timestamps, which include the creation, modification and access times of the malware file. These timestamps can reveal when the malware was compiled and can be cross-referenced with known campaigns. File properties, such as file size, version information, and digital signatures, help identify similarities between different malware samples. Compiler metadata includes elements like Rich headers, which contain information about the compiler, linker, and build environment, allowing analysts to cluster and attribute malware to specific toolchains or developers. Additionally, PDB paths, which are debugging symbols or paths to Program Database (PDB) files embedded in the executable, can provide insights into the development environment and sometimes even the developer's username or machine name.

Rich headers

The Rich header in a Portable Executable (PE) file contains encoded metadata about the build environment, such as compiler version and linked libraries. This data, although not immediately human-readable, provides valuable insights into the specific toolchains and configurations used during the software's creation. By extracting and decoding the Rich header, analysts can identify patterns and attributes unique to particular development environments, which is crucial for understanding the origins and characteristics of malware.

Rich header hashing involves generating a unique cryptographic hash from the decoded Rich header data. This hash acts as a fingerprint for the build environment of an executable. By comparing these hashes, analysts can cluster malware samples that share the same or similar build environments, aiding in the identification of related malware families. This process is essential for tracking APTs, as it allows for the attribution of new malware samples to known threat actors based on shared development practices.

Figure 7 shows a YARA rule example [5] that tracks the Gazer payload associated with the Turla APT using the rich header hash.

```
rule APT_RU_Turla_Gazer_Embedded_Resource_RichHeader
{
  meta:
    description = "lets get weird - track embedded Gazer payloads based on their shared rich headers"
    hash = "d0b169d2e753191a5c366a863d216bc5a9eb5e173f0bd5a61f126c4fd16484ac"
    hash = "473aa2c3ace12abe8a54a088a08e00b7bd71bd66cda16673c308b903c796bec0"
    hash = "a65bc4adbd61c098acf40ef81dc8b6b10269af0d9ebbdc18b48439df76c18cb3"
    DaysofYARA_day = "94/100"
    author = "Greg Lesnewich"
  condition:
     for any var_rsrc in pe.resources: (
      uint16be(var_rsrc.offset) == 0x4d5a and
      hash.md5(var_rsrc.offset+0x80, 0x80) == "dd006bd9a43c05c9457a9e6b9e1636ca")
}
```

Figure 7: YARA rule that tracks the Gazer payload associated with the Turla APT.

PDB

PDB (Program Database) files store debugging information for programs compiled with *Microsoft Visual Studio*. They include symbols, source line information, and stack traces. In malware analysis, PDB files are valuable because they reveal function and variable names, making it easier to understand the malware's structure and behaviour. Additionally, if malware authors forget to strip PDB paths or accidentally include PDB files, analysts can use this information to gain deeper insights into the malware's operations.

PDB strings and paths refer to specific information within a PDB file that can include paths to source code files on the original developer's machine. Extracting these strings can provide crucial clues about a malware's origin and development environment. For example, PDB paths often contain the directory structure of the developer's environment, which can reveal usernames, project names, and other identifiers. This information can help analysts trace the malware back to its source, identify related malware, and understand the development practices of the malware authors.

For example, by using the PDB path reported by security researchers, we can track malware samples associated with APT29 [6].

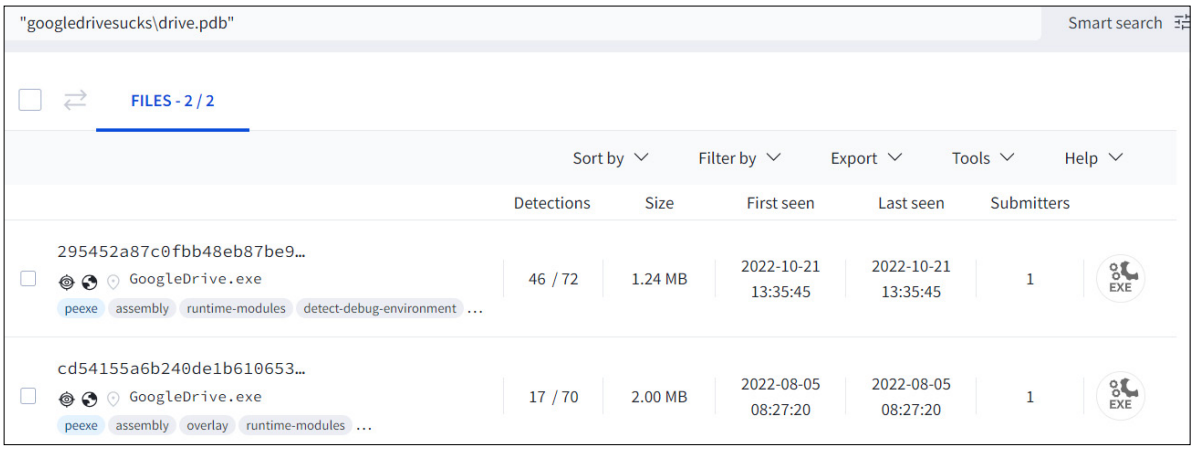

Figure 8: Using the PDB path to track malware samples associated with APT29.

Code-signing certificates

Code-signing certificates are digital certificates used to sign software, ensuring its authenticity and integrity. However, threat actors can misuse these certificates to sign malware, making it appear legitimate and bypassing security mechanisms. Notable examples include the Operation Aurora APT campaign, where attackers used stolen code-signing certificates to sign their malware, and the Stuxnet worm, which utilized legitimate but stolen certificates to evade detection. These cases highlight how the misuse of code-signing certificates can enhance the persistence and effectiveness of malware, posing significant challenges to cybersecurity defences.

Tracking and analysing code-signing certificates used in malware can provide critical insights for attribution and threat intelligence. By examining the details of certificates, such as issuers, serial numbers and signatures, researchers can identify patterns and link different malware samples to the same threat actors or groups. This process aids in the attribution of malicious activities, as consistently reused certificates can point to specific perpetrators. Tools like *Sigcheck* can be used to examine digital signatures and detect software signed with known malicious certificates.

In one of our most recent cases, we identified a new driver called WinTapix [7], which used code-signing certificates from 'Beijing JoinHope Image Technology Ltd.' and 'VeriSign Class 3 Public Primary Certification Authority - G5'. Although these certificates were not exclusive to WinTapix samples, they provided valuable points to narrow down our search for related samples associated with this driver. This highlights the importance of refining your search criteria when tracking malware and APTs.

For instance, in the WinTapix case, we combined the analysis of the used certificates with other factors, such as the targeted victims (in this case, users in Saudi Arabia) and the focus on driver files. By narrowing down our search criteria in this way, we were able to identify and track related samples effectively. This approach demonstrates how specific attributes, like code-signing certificates and geographic targeting, can be instrumental in uncovering and attributing malware to particular threat actors or campaigns.

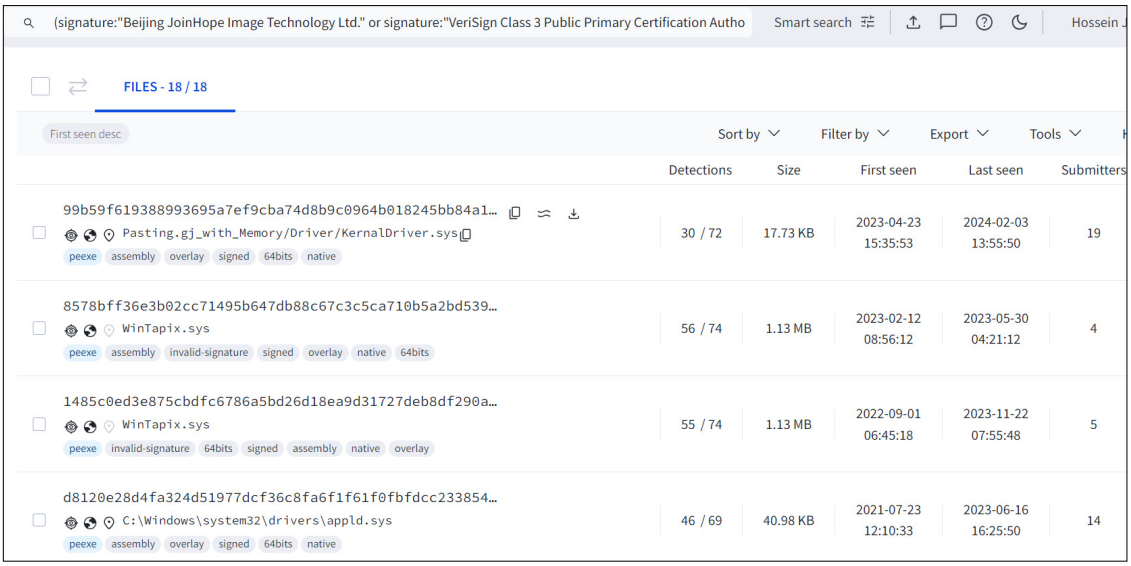

Figure 9: Analysis of code-signing certificates.

TTP-based tracking

TTPs are fundamental in tracking and attributing APTs. Tactics represent the overarching goals of adversaries, such as gaining initial access or maintaining persistence. Techniques outline the general methods used to achieve these goals, like exploiting vulnerabilities or deploying phishing campaigns. Procedures detail the specific implementations of these techniques, such as using particular tools or malware variants. Analysing TTPs helps cybersecurity professionals recognize patterns, link new attacks to known adversaries, and enhance attribution efforts, thus improving defences against evolving cyber threats.

The MITRE ATT&CK framework is a comprehensive matrix that categorizes TTPs used by adversaries based on real-world observations. It provides a detailed view of the behaviours and methods employed by threat actors across various stages of an attack. This framework serves as a valuable resource for cybersecurity professionals to understand, detect and mitigate threats by mapping specific activities to known TTPs. Using this framework and its standards, we can effectively track APTs based on the specific sets of TTPs they employ in their operations.

Complementing the ATT&CK framework is the Malware Behavior Catalog (MBC), which focuses specifically on the behaviours of malware. MBC categorizes and describes the actions malware can perform, such as data exfiltration, credential theft, and persistence mechanisms. By integrating MBC with ATT&CK, analysts can gain a more comprehensive understanding of both adversary strategies and malware behaviours. This integration aids in identifying consistent attacker behaviours, improving attribution accuracy, and developing tailored defences.

For example, understanding the detailed TTPs associated with APT28 allows anticipation and identification of future attacks by recognizing similarities in their tactics, such as spear-phishing and exploitation of software vulnerabilities, and techniques like custom malware deployment and credential theft.

Network-based tracking

Infrastructure tracking for APTs involves monitoring and analysing the various components and systems that APT groups use to conduct their operations, such as domains, IP addresses, servers, communication channels, and other network artifacts. This tracking is crucial as it helps cybersecurity professionals uncover the infrastructure used by APTs, identify patterns, and link different attacks to the same threat actor. Key techniques in infrastructure tracking include passive DNS analysis, SSL/TLS certificate analysis, IP reputation databases, WHOIS data analysis, network traffic analysis, and fingerprinting techniques.

The benefits of infrastructure tracking are significant. It allows for the early detection of malicious activity, disrupts APT operations by identifying and neutralizing their infrastructure, and improves overall cybersecurity defences by understanding the methods and resources APTs use. However, this approach also presents challenges, such as the need for continuous monitoring and updating of intelligence data, the complexity of correlating disparate data sources, and the potential for APTs to rapidly change their infrastructure to evade detection. Despite these challenges, infrastructure tracking remains a vital component of modern cybersecurity strategies, providing a proactive approach to identifying and mitigating the threat posed by sophisticated APT groups.

Infrastructure reuse

It is common for some APT groups to reuse infrastructure due to limited resources. Over time, we have observed that certain APTs reuse IP addresses and domains in their campaigns. A new feature in *VirusTotal* (nethunt [8]) allows users to develop YARA rules to identify samples that use the same infrastructure. These samples could be part of a new campaign that reused the infrastructure, or they could be part of the same campaign we analysed.

The following rules detect samples associated with the BeaverTail malware linked to the Lazarus APT group that are using the same infrastructure:

```
rule C2_IPs_Lazarus
{
  meta:
    author = "Fortinet TI"
    description = "Rule to find files that contact malicious IPs used by North Korea's
Lazarus group"
    target_entity = "ip_address"
  condition:
    // CTI-194
    vt.net.ip.raw matches /^67.203.7.171/ or
    vt.net.ip.raw matches /^147.124.212.89/ or
    vt.net.ip.raw matches /^147.124.214.129/ or
    vt.net.ip.raw matches /^147.124.214.131/ or
    vt.net.ip.raw matches /^147.124.214.237/
}
```

In addition to infrastructure reuse, in some cases, there are overlaps between infrastructure used by a threat actor in different campaigns. These overlaps can be useful for tracking and attribution. In the following case, we found [9] infrastructure overlaps between campaigns used by the Lazarus APT group.

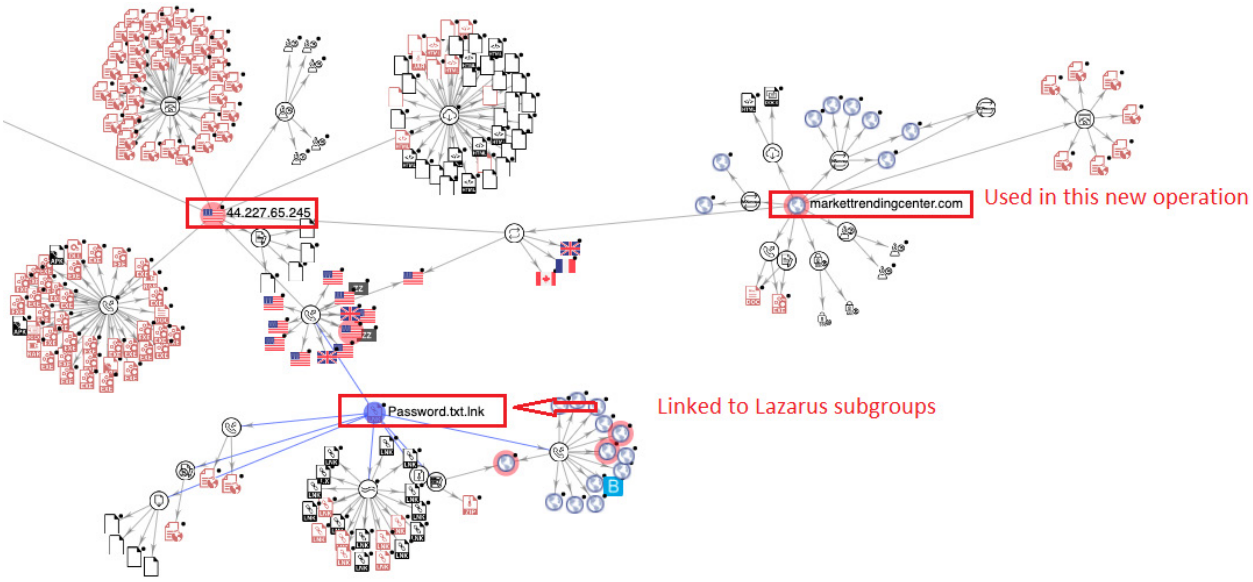

Figure 10: Infrastructure overlaps between Lazarus APT campaigns

Domain names

Using domain names to track APTs involves monitoring and analysing domain names and their activities to identify malicious behaviours associated with APT groups. Here are some ways domain names can be used for tracking APTs:

- **Passive DNS**

  Passive DNS (pDNS) is a technique for collecting and storing DNS resolution data passively, capturing the responses to DNS queries as they occur across the internet. This data includes domain names, IP addresses, query types and timestamps, and is stored in databases for historical lookups. Unlike traditional DNS, which actively resolves domain names to IP addresses in real time, pDNS focuses on historical data, allowing researchers to analyse how domain names have resolved over time. This provides valuable insights into the infrastructure and activities associated with various domains.

  Tracking APTs with passive DNS involves analysing historical DNS data to uncover the infrastructure used by threat actors. By examining patterns in domain resolutions and IP address associations, security researchers can identify malicious domains and their connections to known APT campaigns. pDNS helps detect domain generation algorithms (DGAs) used by malware, map out the network of related domains and IPs, and attribute attacks to specific threat groups. This information is crucial for proactive defence measures, enabling organizations to block or monitor suspicious domains and IP addresses linked to APT activity.

  A good example of using passive DNS is the monitoring of APT42 activity, which has been documented by security researchers. The first domain (jpostpress.com) used by this actor was created in January 2022. Using passive DNS, we can see that this domain resolved to a few IPs, including 91.195.240.12.

  By analysing the historical data associated with this IP, we identified additional domains used by the same threat actor, which were created in September 2022 and March 2023 (themedealine.org, maariv.net, khaleejtimes.org). This analysis highlights how passive DNS can uncover the infrastructure and timeline of APT activities, aiding in the attribution and mitigation of threats.

- **WHOIS data analysis**

  WHOIS databases contain registration details of domain names. By analysing WHOIS data, security teams can identify patterns and commonalities in the registration information used by APT groups. For example, if multiple malicious domains share the same registrant email address, it could indicate a common threat actor.

  Additionally, it is worth noting that tracking domains used by threat actors typically requires a combination of methods. For example, you can utilize specific techniques such as monitoring the themes used by threat actors for domain registration, identifying specific top-level domains (TLDs) commonly used, and tracking the registrars frequently utilized by threat actors.

IP addresses

IP addresses can provide a wealth of information for tracking APTs. Details such as WHOIS data, ASNs, hosting providers, CIDR blocks, server types, running services, and ports can all be instrumental in identifying the infrastructure used by APTs. Efficiently tracking threat actors often requires combining multiple methods; for example, an actor might consistently use a specific ASN, hosting provider, and set of ports.

Here is an example of IP metadata that is used to track APTs:

ASN 20473 (CHOOPA) is known to be utilized by various threat actors, including Vicious Panda and IndigoZebra [10], both of which are associated with Chinese APTs. It's important to note that relying on a single indicator is insufficient; a combination of factors is necessary for precise threat actor tracking.

Since most IP addresses are running HTTP or HTTPS services, we can leverage artifacts in those services to track APTs. These artifacts can be categorized into three main categories: response headers, response content, and certificates.

Certificates and their fingerprints

SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are cryptographic protocols designed to provide secure communication over a computer network. TLS, the successor to SSL, ensures data privacy and integrity between communicating applications, such as web browsers and servers. These protocols use certificates to establish a secure connection. An SSL/TLS certificate contains the server's public key and identity information, verified by a Certificate Authority (CA). When a client connects to a server, it uses this certificate to authenticate the server and establish an encrypted communication channel, ensuring that the data exchanged remains confidential and tamper-proof.

Fingerprinting in the context of SSL/TLS involves creating unique identifiers based on the characteristics of the TLS handshake. JA3 is a technique used to create a fingerprint of a TLS client by hashing specific fields from the TLS Client Hello message, including the SSL/TLS version, accepted cipher suites, list of extensions, elliptic curves, and elliptic curve formats. JA3+ extends JA3 by incorporating additional details from subsequent handshake messages, enhancing the accuracy of client identification. Similarly, JA4 and JA4+ are methods to fingerprint TLS servers, focusing on the server's handshake parameters and behaviours. JA4+ provides even more detail by including data from extended handshake messages. JARM is a tool developed by *Salesforce* that fingerprints TLS servers by analysing their responses to specific Client Hello messages, helping identify and track malicious or unauthorized servers. It is particularly useful in cybersecurity for identifying command-and-control (C2) servers and ensuring compliance with security policies.

These fingerprints are valuable in cybersecurity for identifying and tracking APTs. APTs often use specific tools and techniques to evade detection, and their unique TLS handshake patterns can be identified using JA3, JA3+, JA4, JA4+, and JARM fingerprints. By analysing these fingerprints, security analysts can detect anomalies, identify malicious traffic, and attribute activities to specific threat actors.

In this section we provide some examples in which certificates and fingerprints are used to track APTs.

JARM fingerprint

Storm-0558 is a threat actor group identified by *Microsoft* that has been involved in cyber-espionage activities primarily targeting government agencies, NGOs, IT services, and telecommunication firms across Europe and the Americas. This group is known for its sophisticated TTPs and has been linked to attacks leveraging OAuth tokens to gain access to email systems.

The threat actor utilized dedicated infrastructure running SoftEther proxy software, providing researchers an opportunity to track the associated APT infrastructure. A specific JARM fingerprint has been identified for this threat actor, which aids in tracking the proxy software. However, relying solely on the JARM hash is insufficient for tracking the infrastructure associated with this threat actor, as it could be used by other actors as well. Therefore, researchers emphasize the importance of combining multiple artifacts to effectively trace the threat actor's infrastructure.

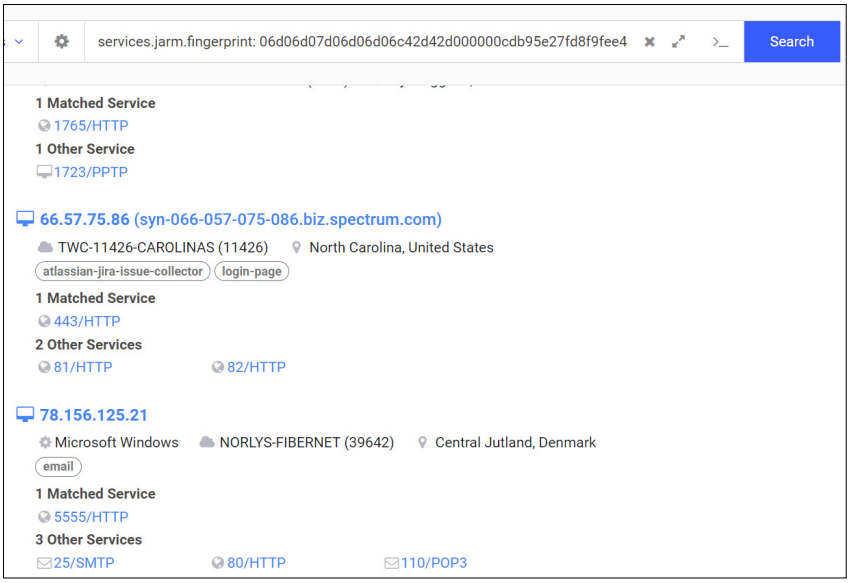

Figure 11: Use of JARM fingerprint.

To track the infrastructure linked to this APT, researchers [11] have identified three main indicators that must be used together to locate all servers used by the threat actor:

- Hosts in this network exhibit a JARM fingerprint consistent with SoftEther VPN: 06d06d07d06d06d06c42d42d000000cdb95e27fd8f9fee4a2bec829b889b8b.
- The x509 certificate presented by the infrastructure expires on 31 December 2037.
- Importantly, the subject information in the x509 certificate does not include the term 'softether'.

Certificate values/hash

SSL/TLS certificates contain several values that can be used for tracking and identification purposes. These values include:

- Common Name (CN): This is the primary identifier for the certificate and typically represents the domain name or server name for which the certificate is issued. It helps identify the entity that owns or controls the server.
- Subject Alternative Names (SANs): These are additional domain names or IP addresses for which the certificate is valid. They expand the scope of domains or servers covered by the certificate.
- Issuer: The entity that issued the certificate, often a CA. This can provide information about the trustworthiness of the certificate.
- Serial Number: A unique identifier assigned by the issuer to distinguish the certificate from others issued by the same CA.
- Validity Period: The dates during which the certificate is considered valid. This helps in understanding the lifecycle of the certificate.
- Public Key: The public key associated with the certificate's private key, used for encryption and authentication.
- Signature Algorithm: The algorithm used by the CA to sign the certificate, indicating the security level and method used in the certificate's creation.
- Key Usage: Specifies the cryptographic operations for which the public key in the certificate can be used.
- Extended Key Usage (EKU): Specifies additional purposes for which the public key can be used, beyond the basic key usage.

You can use these values or certificate hashes to track a threat actor. For example, you can use a certificate hash to track Kimsuky APT but, like other cases, the use of certificates alone is not enough to track threat actor, so you need to use a combination of factors to be able to find servers associated to specific threat actor.

```
services.certificate="9de541b039cfdb96c7810df49efd958b28cc2df73e314f67c1a91469a2b19796" and
autonomous_system.asn= 19318 and service_count=3
```

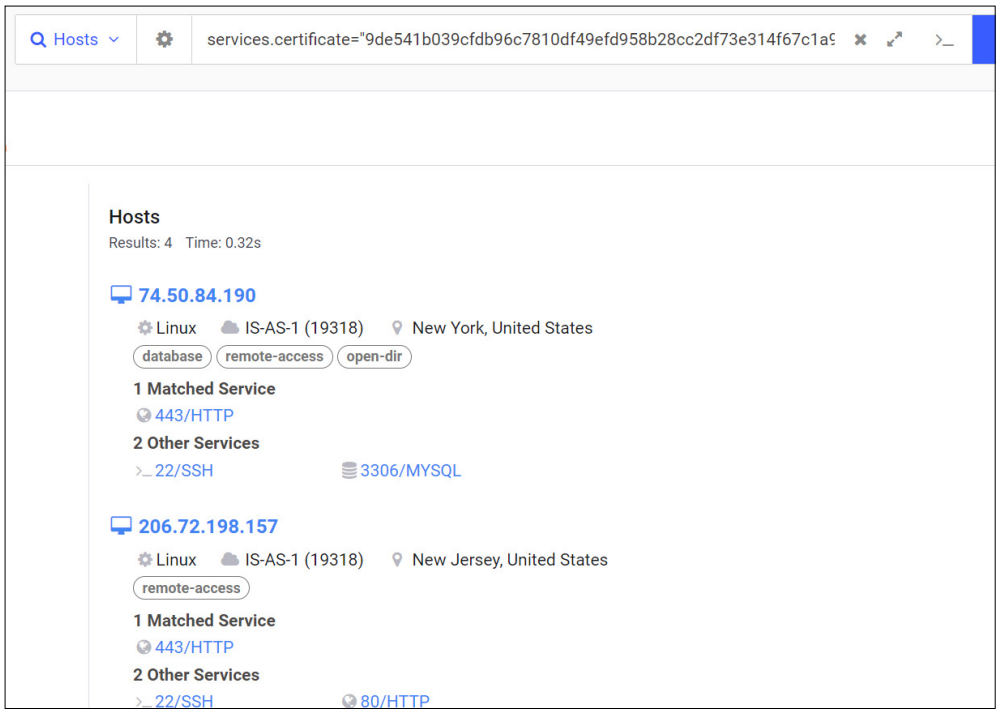

Figure 12: Use of certificate hash for tracking.

In some cases, regex can be used to track APTs. While the subject or common name may not be unique enough to search by its exact value, it often follows a specific pattern that can be identified using regex. Utilizing regex allows for

comprehensive tracking of all infrastructure associated with a threat actor. For a practical example, you can find a good illustration of this concept in [12].

Response headers

A response header in the context of HTTP is metadata sent by a web server to a client as part of an HTTP response. It provides additional information about the server's response and instructions on how the client should handle the received data. Response headers are structured as name-value pairs and are transmitted alongside the actual content of the response.

APTs can be tracked using various aspects of response headers, revealing patterns and anomalies in network traffic indicative of malicious activities.

Values in response headers can be used to track APTs:

- **User-agent analysis**: Identifying unusual or spoofed user-agent strings.
- **Host and referer tracking**: Monitoring requests to detect suspicious or unauthorized hosts and referer headers.
- **Cookie analysis**: Inspecting cookies for unusual patterns or values.
- **Custom headers**: Identifying non-standard or custom headers used by malware.
- **ETag and cache-control headers**: Detecting specific ETag values or cache-control mechanisms.
- **HTTP method anomalies**: Monitoring unusual use of HTTP methods.
- **Content-type and encoding analysis**: Inspecting content types and encoding methods.

More specifically when you want to track APTs you must look for unique header combinations, custom headers or header values or patterns:

- Unique header combinations: Some APT groups use distinctive combinations of HTTP headers, aiding in server clustering and identification.
- Custom headers in malware: Malicious actors employ custom headers for internal operations or C2 communication, signalling potential threats.
- Regex patterns for identification: Regular expressions can uncover patterns in header values, facilitating APT infrastructure detection.

Here are some examples in which HTTP headers have been used to track APTs:

- MuddyWater, a known Iranian cyber-espionage group, has been observed [13] deploying its own web servers on purchased VPSs. The group utilizes unique ETag hashes, which are used to identify the servers and their activities. The detected ETag hashes associated with MuddyWater include:

  - 2aa6-5c939a3a79153
  - 2aa6-5b27e6e58988b
  - 2aa6-5c939a773f7a2

  These ETag hashes have been linked to servers known to be used by MuddyWater for malicious activities. Additionally, some of these servers have connections to various malicious files or software used in attacks, including legitimate SimpleHelp installers.

- Cobalt Strike Framework:

  The 'CS-Bid' HTTP header is utilized by Cobalt Strike for communication between compromised hosts and C2 servers. This header serves to uniquely identify instances of Cobalt Strike activity across networks, allowing security analysts to monitor and respond to potential compromises effectively. Detection rules are often configured to trigger alerts on outbound HTTP traffic containing CS-Bid headers, enabling timely detection and mitigation of malicious activity associated with this tool.

Response content

HTTP response content refers to the data sent by a web server to a client in response to an HTTP request. This content is part of the HTTP response and includes various elements that can provide useful information about the web server, the content being served, and any actions the server might want the client to take.

Analysing HTTP response content is crucial for identifying and tracking APTs. Various components of HTTP responses, such as HTML, stylesheets, JavaScript, splash pages, favicons, and more, can provide valuable insights into the TTPs used by APT groups.

- **HTML content**:

  HTML hashes: Unique hashes of HTML content help identify known malicious pages. By comparing these hashes against databases of known threats, analysts can quickly identify if a page is part of an APT campaign.

- **Stylesheets (CSS)**:
  CSS patterns: Distinctive CSS files or styles can be fingerprinted. Analysing stylesheets for unique patterns or anomalies helps identify pages designed by the same threat actor.
- **JavaScript**:
  JavaScript code: Malicious JavaScript used for activities like form grabbing or keylogging can be identified by its unique functions or obfuscation techniques. Recognizing these patterns aids in tracking APT activities.
- **Splash pages**:
  Splash page characteristics: Analysing the design, content, and behaviour of splash pages can reveal reused elements indicative of a specific APT group, such as consistent graphics or loading sequences.
- **Favicons**:
  Favicon fingerprinting: Unique or unusual favicons can be fingerprinted to identify related malicious sites, linking them to the same APT group.
- **Embedded links and resources**:
  Resource URLs: Consistent use of certain domains or resource paths in HTTP responses can indicate the same APT group, as these URLs are often reused across malicious sites.
- **Redirects and response codes**:
  HTTP status codes: Unusual patterns in status codes or unexpected redirects can indicate malicious activity and help identify a specific APT's behaviour.
- **Content obfuscation and encoding**:
  Encoding techniques: Identifying specific obfuscation or encoding methods, such as Base64 or gzip, can reveal patterns used by APT groups.
- **Custom error pages**:
  Custom 404 or error pages: Analysing the content and design of custom error pages can link different malicious sites through unique design elements or messages.
- **Dynamic content and APIs**:
  Dynamic content: Monitoring how dynamic content is generated and served can reveal backend infrastructure details linked to APT activities.
- **Third-party analytics and tracking codes**:
  Embedded tracking codes: Reuse of analytics tracking codes across multiple sites can indicate a common threat actor, as seen with shared *Google Analytics* IDs in phishing campaigns.

In the following we will provide some examples in which you can see how response content is used to track an APT.

Kimsuky APT

In some cases, APT groups may use specific response pages that can be tracked using the hash of the page. For instance, you can use the hash of a response page to identify and monitor related APT activity. Additionally, specific strings within the response body can be used to track servers associated with threat actors like Kimsuky [14].

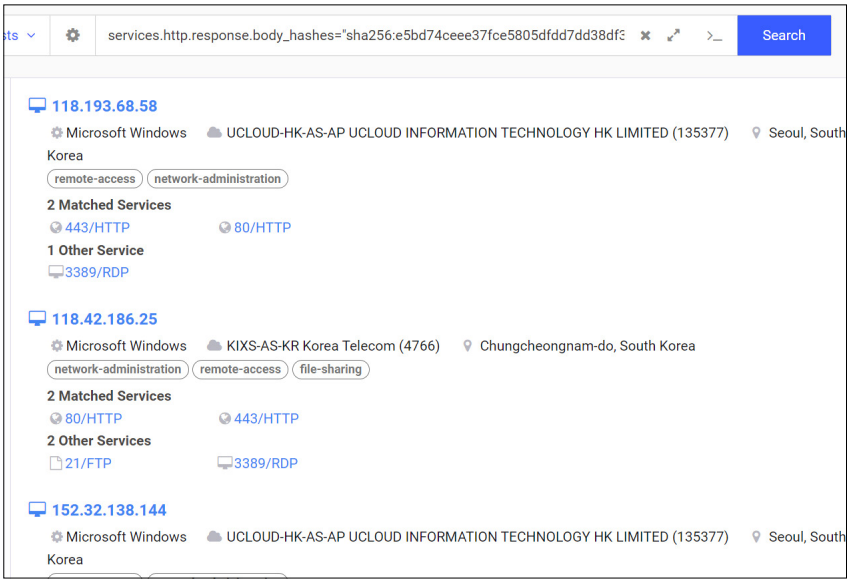

Figure 13: Use of response body hash for tracking.

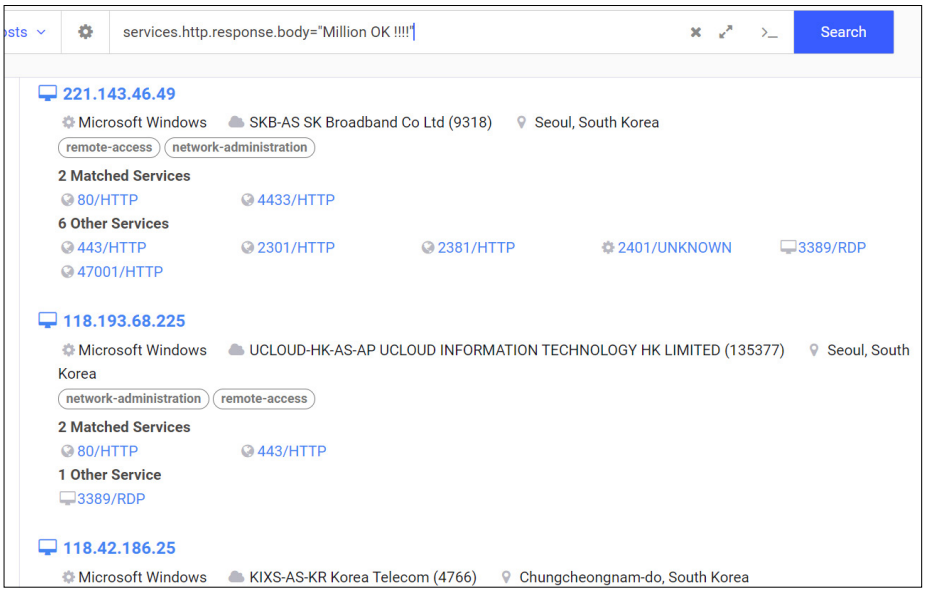

Figure 14: Use of response body content for tracking.

Qbot

In some cases, you can use HTTP response content, specifically the HTTP response title, to track servers associated with a threat actor or specific malware. Researchers [15] have identified that the HTML title is unique for Qbot malware, and by leveraging this, they were able to track servers associated with this malware.

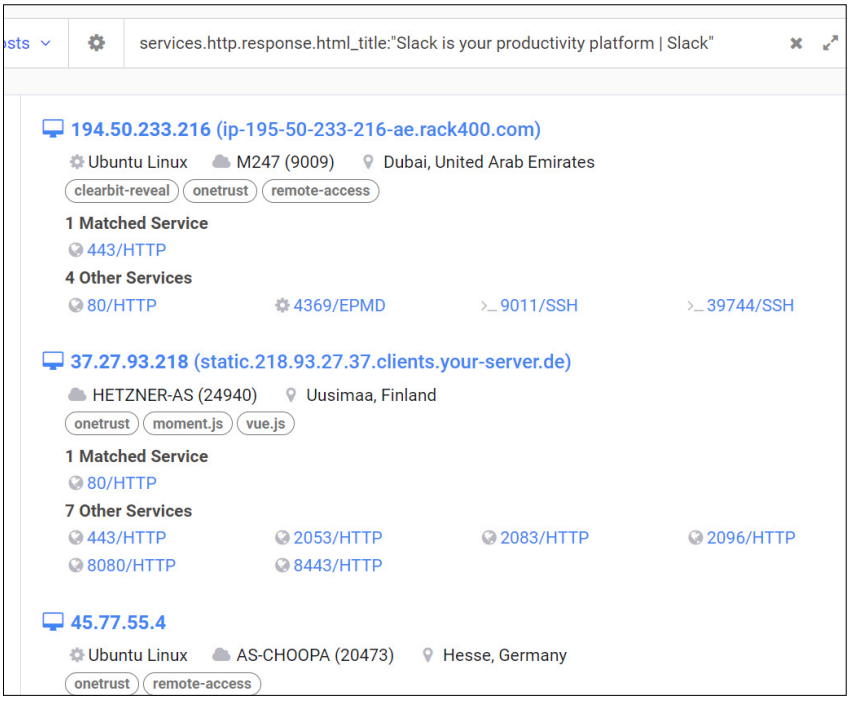

Figure 15: Use of response HTML title for tracking.

TOOLS FOR APT TRACKING

Tracking APTs requires a variety of specialized tools and platforms that provide visibility into different aspects of internet infrastructure and cybersecurity threats. Here are some notable tools commonly used for tracking APTs:

- *Shodan*: A search engine for internet-connected devices. It allows users to discover servers, routers, webcams, and other devices accessible on the internet, providing insights into potential vulnerabilities and exposure points.
- *Censys*: Similar to *Shodan*, *Censys* scans and monitors the internet for devices and services. It offers detailed information about hosts, networks and websites, aiding in identifying potential security risks and malicious activities.

- *DomainTools*: Provides DNS and domain-related information, including historical data, WHOIS details, and domain ownership records. It helps in tracking domain infrastructure used by threat actors for malicious purposes.
- *FOFA*: A search engine for discovering internet assets based on keywords, including domains, IP addresses, and specific HTTP headers. It's used to find potential attack surfaces and identify indicators of compromise (IOCs).
- *VirusTotal*: A web-based platform that analyses files and URLs to detect malware and other malicious content. It aggregates data from multiple antivirus engines and threat intelligence sources, providing insights into potential threats associated with files and domains.
- *Maltego*: A powerful tool for data mining and link analysis. It enables investigators to visualize complex networks of relationships between people, groups, and online assets, making it useful for mapping APT infrastructure.
- *ThreatConnect*: A threat intelligence platform that integrates with various data sources to provide insights into threat actor TTPs. It supports collaborative threat analysis and incident response efforts.
- *PassiveTotal*: A tool that specializes in passive DNS and passive SSL analysis. It aggregates data from various sources to provide enriched intelligence on domains, IP addresses, and internet entities. This tool assists in investigating threats, monitoring infrastructure, and mapping attacker activities, making it valuable for tracking APTs by correlating threat indicators and identifying malicious patterns over time.

ATTRIBUTION

Attribution of APTs involves identifying the individuals, groups, or nation-states behind cyber attacks based on evidence collected from various sources. This process often overlaps significantly with APT tracking, which focuses on monitoring, analysing, and identifying patterns in malicious activities and infrastructure. Effective attribution relies on a combination of technical indicators, such as malware analysis, network traffic analysis, and forensic investigation, as well as contextual factors like geopolitical insights and historical attack patterns.

APT tracking forms the foundational basis for attribution efforts by establishing a detailed understanding of threat actor TTPs. Tools and techniques used for tracking, such as passive DNS analysis, threat intelligence platforms, and correlation of IOCs, provide critical insights into APT operations over time. These insights aid not only in identifying ongoing threats but also in connecting disparate incidents to specific threat actors.

Furthermore, attribution extends beyond technical indicators to include behavioural analysis, motive assessment, and geopolitical context. By integrating these elements, cybersecurity analysts can attribute APT activities to known threat groups or state-sponsored actors with greater confidence. This holistic approach to attribution enhances strategic response planning, facilitates diplomatic discussions, and supports legal actions against threat actors.

In summary, while APT tracking focuses on identifying and monitoring malicious activities, attribution complements this effort by linking those activities to specific threat actors or entities. Together, they provide a comprehensive understanding of APT operations, enabling proactive defence measures and informed decision-making in cybersecurity and national security domains.

CONCLUSION

In conclusion, tracking and attributing APTs requires a multifaceted approach integrating technical tools, threat intelligence, and analytical methodologies. Tools like *Shodan*, *Censys* and *PassiveTotal* provide critical capabilities in monitoring infrastructure, analysing threat indicators, and correlating data to uncover malicious activities. By leveraging these tools alongside advanced analytics and threat intelligence feeds, cybersecurity professionals can effectively identify, track, and attribute APT operations.

Throughout this paper, we have covered various real-world examples demonstrating successful APT tracking efforts. These examples highlight the effectiveness of the covered methods in tracking APTs and aiding in attributing activities to specific threat actors.

REFERENCES

[1] Stockley, M. New Mac variant of Lazarus Dacls RAT distributed via Trojanized 2FA app. Malwarebytes. 6 May 2020. https://www.threatdown.com/blog/new-mac-variant-of-lazarus-dacls-rat-distributed-via-trojanized-2fa-app/.

[2] CERT-FR. FIN12 UN GROUPE CYBERCRIMINEL AUX MULTIPLES RANÇONGICIELS. 18 September 2023. https://www.cert.ssi.gouv.fr/uploads/CERTFR-2023-CTI-007.pdf.

[3] Jazi, H. Lazarus APT conceals malicious code within BMP image to drop its RAT. Malwarebytes. 19 April 2021. https://www.threatdown.com/blog/lazarus-apt-conceals-malicious-code-within-bmp-image-to-drop-its-rat/.

[4] Palo Alto Networks Unit 42. Hacking Employers and Seeking Employment: Two Job-Related Campaigns Bear Hallmarks of North Korean Threat Actors. 21 November 2023. https://unit42.paloaltonetworks.com/two-campaigns-by-north-korea-bad-actors-target-job-hunters/.

[5] g-les / 100DaysofYARA. https://github.com/g-les/100DaysofYARA/blob/main/100_days_of_yara.yar.

[6] Harbison, M.; Renals, P. Russian APT29 Hackers Use Online Storage Services, DropBox and Google Drive. Palo Alto Networks Unit 42. 19 July 2022. https://unit42.paloaltonetworks.com/cloaked-ursa-online-storage-services-campaigns/.

[7] Revay, G.; Jazi, H. WINTAPIX: A New Kernel Driver Targeting Countries in The Middle East. Fortinet. 22 May 2023. https://www.fortinet.com/blog/threat-research/wintapix-kernal-driver-middle-east-countries.

[8] VirusTotal. Network hunting: Writing YARA rules for Livehunt/Hunting network assets. https://virustotal.readme.io/docs/nethunt.

[9] Saini, A.; Jazi, H. North Korea's Lazarus APT leverages Windows Update client, GitHub in latest campaign. Malwarebytes. 21 January 2022. https://www.threatdown.com/blog/north-koreas-lazarus-apt-leverages-windows-update-client-github-in-latest-campaign/.

[10] Check Point Research. IndigoZebra APT continues to attack Central Asia with evolving tools. 1 July 2021. https://research.checkpoint.com/2021/indigozebra-apt-continues-to-attack-central-asia-with-evolving-tools/.

[11] Microsoft. Analysis of Storm-0558 techniques for unauthorized email access. 14 July 2023. https://www.microsoft.com/en-us/security/blog/2023/07/14/analysis-of-storm-0558-techniques-for-unauthorized-email-access/.

[12] Embee Research. How To Build Advanced Threat Intelligence Queries Utilising Regex and TLS Certificates - (BianLian). https://www.embeeresearch.io/building-advanced-censys-queries-utilising-regex-bianlian/.

[13] Group-IB. SimpleHarm: Tracking MuddyWater's infrastructure. 18 April 2023. https://www.group-ib.com/blog/muddywater-infrastructure/.

[14] @Cyberteam008. https://x.com/Cyberteam008/status/1765624539273183623.

[15] Embee Research. Threat Intelligence Query Examples – Real World Queries for Identifying Malware Infrastructure. https://www.embeeresearch.io/shodan-censys-queries/.