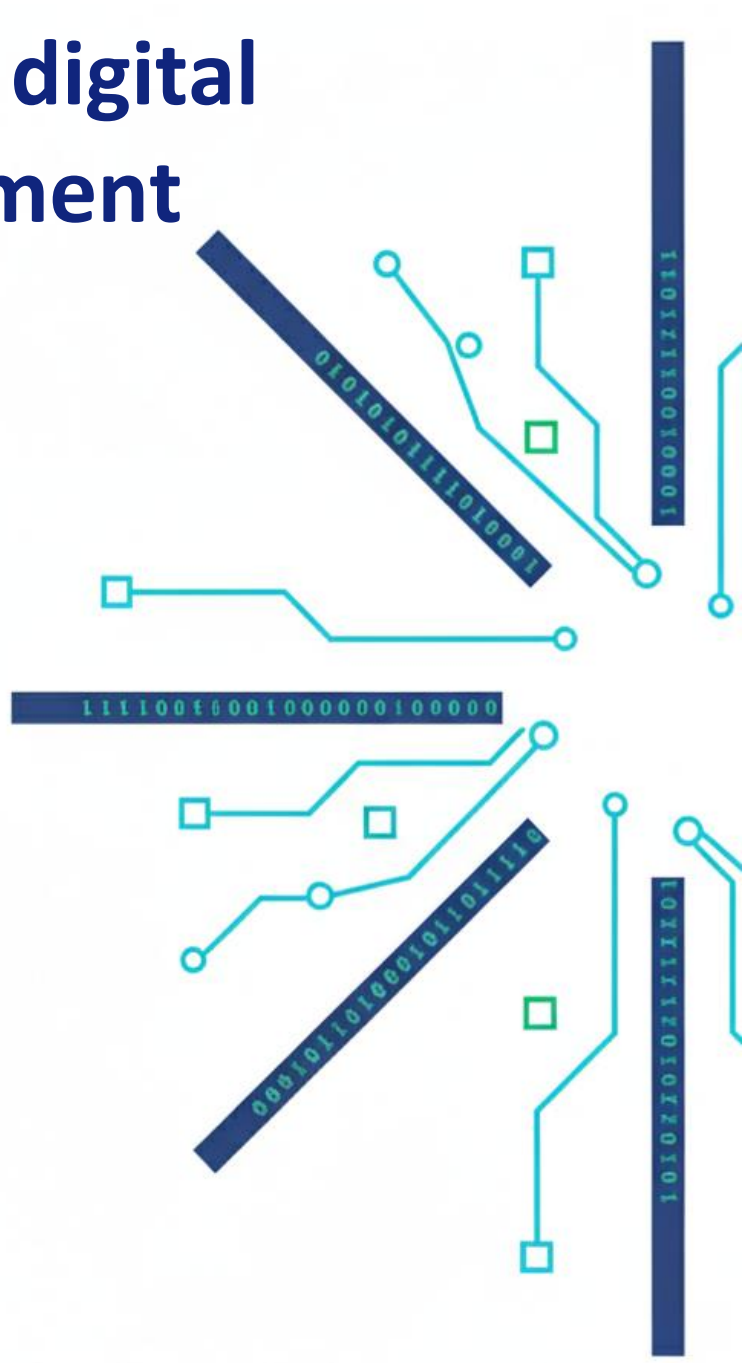
D18 – Model for digital evidence enrichment



The project Automatization of digital forensics and incident response (ADFIR) funded by the European Union – Next GenerationEU through the Recovery and Resilience Plan of the Slovak Republic under project No. č. 09-I05-03-V02-00079.

CONTENTS

1 Project description

The project **Automatization of digital forensics and incident response** (hereinafter referred to as **"ADFIR"**) is funded by the **European Union – Next GenerationEU through the Recovery and Resilience Plan of the Slovak Republic** under project No. č. 09-I05-03-V02-00079. This project addresses one of the key challenges in cybersecurity and information security – how to process the massive volume of digital evidence generated during cybersecurity incidents or forensic investigations. Currently, this process is highly demanding in terms of human resources and time. Therefore, automation using machine learning methods can significantly **improve the quality of digital forensic analysis** and reduce the time required to perform it. Overall, this enables security teams to respond more effectively to cyber threats. Main benefits of this project are:

- **Accelerated Resolution of Cybersecurity Incidents**. The project ADFIR introduces automated approaches to collecting, processing, and analyzing digital evidence. As a result, security teams can identify the causes of incidents more quickly and adopt effective measures to address them.
- **Reduced Workload for Forensic Analysts**. Routine and time-consuming tasks involved in processing digital evidence will be replaced by automated methods. This will allow analysts to focus on more complex cases and strategic decision-making.
- **Higher Quality and Consistency of Outputs**. The use of unified methodologies and tools ensures that the processed digital evidence will be more accurate, consistent, and easily verifiable. This significantly reduces the risk of errors caused by human factors.
- **Potential Use in Criminal Proceedings.** The project outputs will be developed in compliance with legal requirements and standards, allowing the digital evidence to be accepted as relevant evidence for investigations and court proceedings.

2 Digital Evidence Enrichment Options

As a first step, it is necessary to clarify what information can be obtained from digital evidence and which of them are suitable for further enrichment. The following criteria were considered in the selection:

- Is this a piece of information that can be used as an IoC (Indicator of compromise)?
  - For example, the IP address, URL or hash of a file are standard examples that can be further worked with. There are multiple trusted services publicly available which enable us to obtain additional information.
- Is this information that can usually be extracted from digital traces?
  - FQDN, MAC address, phone number or name of the running process occur, whether in the role of an important trace or irrelevant data, on the disk or in the event logs of majority of devices we might analyze.
- Are there sufficiently verified external sources through which the data can be enriched with knowledge from the community?
  - Similar constraints as to data that we can use as an IoC.

We summarized the selected types of data and identified the possibilities of their enrichment. The table 1 provides an overview of them and their importance for forensic analysis, as well as the most common artifacts in which a given type of data may be found. Individual methods for enriching digital artifacts are presented in the extended table that we attach to the document.

#	Data Type	# Enrichments	Key Use in Forensics	Forensic Sources (Artifacts) of Data Type
1	API Key / Token	5	Detect credential theft, unauthorized API usage, and key exposure	File system, configuration files, scripts
2	Cloud Resource ID	7	Investigate cloud breaches, misconfigurations, and unauthorized access	AWS/Azure/GCP - probably direct analysis in console, or otherwise obtained data about cloud infrastructure.
3	Code Signing Signature	6	Validate software legitimacy, detect signed malware	Executable´s signature
4	Cryptocurrency Address	5	Track ransomware payments and financial attribution	Ransom notes, malware samples. Personal cryptowallets, user´s transaction history.
5	CVE / Vulnerability	10	Determine exploitation likelihood and prioritize patching	Malware samples
6	Digital Certificate	13	Identify malicious infrastructure, detect certificate abuse	Certificate store (Machine, User certs)
7	DNS Query	6	Detect C2 beaconing, DNS tunneling, and DGA domains	DNS logs.

8	Email Address	10	Trace phishing campaigns, validate sender legitimacy, link personas	Email artifacts: local mailbox, email headers, MS365/Exchange logs and storage, browsing history (visited mailboxes).
9	File Hash	20	Identify malware, map to families, assess capabilities and attribution	Image file, HDD/SSD, exported documents - hashes calculated from its contents; Amcache.
10	FQDN / Domain Name	24	Track C2 infrastructure, phishing domains, and attacker registration patterns	Browsing history, PowerShell logs/trace, malware samples, network logs, esp.web proxy and similar.
11	Geolocation	4	Detect impossible travel, unauthorized access origins	Most probably cloud data sources (MS365, Google account - Google maps, Apple Account, ...), WiFi related registry keys (search for router´s or AP´s MAC address online might be possible).
12	Log Source / Facility	4	Ensure log integrity and proper timeline reconstruction	Log itself, management consoles of SIEM/EDR/XDR/SCCM, etc.
13	MAC Address	5	Track physical devices, detect spoofing and rogue devices	Registry keys, JumpLists, LNK Files (iirc). ARP tables, DHCP logs, switch CAM tables
14	MITRE ATT&CK ID	6	Map adversary behavior to framework, guide detection and response	Various artifacts
15	Mutex Name	2	Identify malware families and detect concurrent infections	Memory image, malware sample.
16	Named Pipe	2	Detect C2 frameworks and lateral movement techniques	Memory image, malware sample, event logs (if used as a service).
17	Network Flow	7	Identify data exfiltration, lateral movement, and C2 communication	Netflows, if available.
18	Network Port / Protocol	4	Detect protocol anomalies, tunneling, and unauthorized services	Firewall (rules and logs), IDS/IPS, EDR. Netflows
19	Person Name	5	Attribution, insider threat investigation, OSINT profiling	Documents, emails and similar, user-related data. Chat applications. Active Directory. Mobile phone and cloud data/contacts.
20	Phone Number	5	Investigate vishing, SMS phishing, and social engineering	Documents, emails and similar, user-related data. Active Directory (if contact details are listed in account´s properties). Mobile phone forensics (contacts). Saved contacts/vcf files. Cloud accounts - contacts (Google, Apple IDs etc.saved contacts).
21	Private IP Address	17	Map internal compromise scope, identify affected systems and segments	Sign-in logs (evtx and other), RDP logs, configuration and log files of remote access tools,

				malware samples. DHCP logs, DNS, Active Directory. Network documentation.
22	Process Name	9	Detect LOLBins abuse, masquerading, and malicious process execution	EDR, SIEM, evtx logs, memory image.
23	Public IP Address	33	Identify attacker infrastructure, hosting, geolocation, and reputation	Sign-in logs (evtx and other), RDP logs, configuration and log files of remote access tools, malware samples. DHCP logs, DNS, Active Directory.
24	Registry Key/Value	5	Identify persistence mechanisms and configuration changes	Windows Registry.
25	Scheduled Task / Cron	5	Identify persistence mechanisms and unauthorized automation	SchTasks registry key, event logs, definition files. Crontab. Note: should be already parsed in standard processing phase!
26	Sigma Rule	4	Validate log-based detections and map to ATT&CK	Misc, sigma rules created based on analysis, known IOCs, intel etc.
27	URL	10	Analyze phishing lures, payload delivery, and redirect chains	Browsing history, PowerShell logs/trace, malware samples, network logs - web proxy.
28	User Account	15	Detect compromised accounts, assess privilege escalation, audit access	Profile list, sign-in logs, Active Directory.
29	User-Agent String	5	Detect malicious tools, C2 frameworks, and anomalous clients	Browsing history, webserver logs.
30	WiFi SSID / BSSID	3	Detect rogue access points and track physical proximity	Registry keys related to networking.
31	Windows Event ID	5	Build attack timeline, detect specific adversary techniques	Windows Event Logs (evtx files).
32	Windows Service	6	Detect service-based persistence and privilege escalation	Registry keys related to services. Event logs (System.evtx, Security.evtx). Note: should be already parsed in standard processing phase!
33	YARA Rule	3	Scope malware infections and validate detection coverage	Misc, YARA rules created based on analysis, known IOCs, intel etc.

Table 1

3 The complexity of enriching a digital artifact – preprocessing

When designing the model, it is advisable to clarify which information that we will further enrich is usually already available in the parsed data, and which requires some form of pre-processing (beyond the already performed parsing of digital evidence). For the purposes of this document, we have divided the types of data for enrichment into three categories:

1. **Parsed**: Data that is usually in the output of parsing a digital evidence and can be immediately further worked with. These include, for example, IP addresses obtained from Event logs, usernames obtained from SAM and ProfileList, SHA1 hashes of files from AmCache, and so on.

2. **Computed**: Data such as hashes of files on the analyzed digital evidence source: we cannot obtain them by parsing standard artifacts, it is necessary to calculate the hashes first. We can also include data that we can extract through the use of regular expressions, such as IP addresses or email addresses, which are located outside the artifacts from which we obtain them by standard parsing. The IP address can be easily found in the event ID 4624 in Security.evtx (Parsed Artifact), but it can also be found in the configuration file, from where it would only be obtained through a more detailed search. However, we will classify IP addresses as such rather as parsed artifacts, as we are primarily considering their occurrence in common, parsed locations.

3. **Compound:** For example, YARA or Sigma rules. In order to be able to create them, we usually have to combine several artifacts, evaluate the necessary logical conditions for the resulting rule to be functional, engage knowledge of designing such a rule, or obtain the necessary existing rules from an external source through threat intelligence.

4 Ways to enrich digital traces

In the following section, we'll introduce each type of data that we propose to enrich. We also included information on artifacts, in which each of “enrichable” data types could be found. Finally, for each data type we provide a description of how enrichment might be executed and what combination with additional data is necessary to obtain the most valuable and conclusive results.

4.1 API Key / Token

4.1.1 Meaning and use

Detect login theft, unauthorized API use, and key disclosure.
API-key or token enrichment can help us understand what a credential can access and how dangerous its exposure might be. In our team’s experience, the most useful elements are the associated service, effective scope, creation details, last-used time, and whether the key has appeared in public repositories. The main caveat is that the value of the key depends on its actual permissions and current validity, so the identifier alone is not enough to judge impact.

4.1.2 Typical artifacts containing data

On Windows systems, API keys and tokens are often recovered from scripts, source code, configuration files, environment variables, browser artifacts, cloud CLI history, PowerShell logs, memory captures, CI/CD files, developer notes, and EDR telemetry. The raw secret may be present in endpoint artifacts, but scope, creator, usage history, and public exposure usually come from cloud-side logs or repository-search tooling rather than the host itself. In practice, the strongest value comes from tying the key to a user, process, repository, or cloud action.

4.1.3 Ways and tools for enrichment

It is usual to identify the service from the token format or associated application, as the first step. Then we can proceed to review IAM policies, OAuth scopes, audit logs, and usage logs to confirm permissions, creator, and last use. We can also check repository-exposure tooling and public code sources to see whether the secret has leaked.

API-key enrichment is most useful when combined with endpoint evidence, cloud audit logs, and access-history review rather than interpreted from the token alone.

4.2 Cloud Resource ID

Investigate cloud breaches, misconfigurations, and unauthorized access

4.2.1 Meaning and use

From a forensic analyst's perspective, cloud-resource-ID enrichment should help us understand what the resource is, who owns it, where it lives, what it can access, and whether it is exposed to the internet. The most useful elements are resource type, owning account or subscription, region, attached IAM role, public exposure, creation details, and tags. However, we must keep in mind that cloud resources change quickly, so ownership and exposure should always be verified against current control-plane logs and metadata.

4.2.2 Typical artifacts containing data

On Windows systems, cloud resource IDs can be recovered from browser history, cloud CLI history, PowerShell logs, scripts, configuration files, EDR telemetry, memory captures, screenshots, developer notes, and exported cloud logs. The raw identifier may be present in endpoint artifacts, but details such as IAM role, public exposure, creator, and account ownership usually come from cloud-side telemetry rather than the endpoint itself. The strongest value comes from tying the resource ID to a user action, credential, script, or access event.

4.2.3 Ways and tools for enrichment

In suggested workflow, we usually look up the resources in the cloud console, audit logs, and IAM metadata to confirm its type, ownership, region, permissions, creation history, and tags. We also check posture or security tools to see whether the resource is publicly accessible or otherwise misconfigured.

Cloud-resource enrichment is most useful when combined with endpoint, identity, and audit-log evidence rather than interpreted from the ID alone.

4.3 Code Signing Signature

Validate software legitimacy, detect signed malware

4.3.1 Meaning and use

From a forensic analyst's perspective, code-signing enrichment helps us judge whether a signed binary is genuinely trustworthy or only appears trustworthy. Fields such as signer name, chain validity, timestamp, revocation status, known abuse, and certificate thumbprint are useful for validating publisher claims, spotting stolen or misused certificates, and clustering related signed malware. In our team's experience, a valid signature helps with context but is never enough on its own, because attackers routinely abuse legitimate certificates.

4.3.2 Typical artifacts containing data

On Windows systems, code-signing details are most often recovered from the executable file itself, but they may also appear in Amcache, EDR telemetry, AV detections, PowerShell execution artifacts, browser download records, and forensic collections of files from disk or memory. Signer names and thumbprints can sometimes be preserved in security-tool metadata even when the original file is no longer available. External details such as revocation state or known malicious use of the same certificate usually require enrichment outside the endpoint.

4.3.3 Ways and tools for enrichment

In our workflow, we usually inspect signatures with tools such as sigcheck, PEStudio, OpenSSL, and VirusTotal to verify signer identity, trust chain, countersignature, revocation, and certificate identifiers. We then pivot on the thumbprint or serial in threat-intelligence sources such as VirusTotal, ReversingLabs, or MISP to see whether the same signing certificate has been used on known malware. Based on our team's experience, code-signing evidence is most useful when combined with file-hash analysis, execution context, and other endpoint artifacts.

4.4 Cryptocurrency Address

Track ransomware payments and financial attribution.

4.4.1 Meaning and use

Cryptocurrency address enrichment helps us understand whether a wallet is just a payment endpoint or part of a larger criminal or regulated ecosystem. The most useful elements are transaction history, ransomware association, related-wallet clustering, exchange attribution, and sanctions screening. The main caveat is that blockchain visibility does not automatically equal attribution, so wallet conclusions should be treated as probabilistic unless supported by stronger external evidence.

4.4.2 Typical artifacts containing data

On Windows systems, cryptocurrency addresses are often recovered from ransom notes, chat logs, emails, browser history, clipboard artifacts, wallet applications, memory captures, scripts, documents, and EDR telemetry tied to extortion or fraud activity. The address itself may be present in endpoint artifacts, but balance history, clustering, exchange ownership, and sanctions context are usually analyst-added enrichments from blockchain and compliance sources. The strongest investigative value comes from tying the wallet address to the extortion event, user interaction, or payment workflow.

4.4.3 Ways and tools for enrichment

In our workflow, we suggest starting with a blockchain explorer (such as https://www.blockchain.com/explorer or many others) to review balance and transaction activity, then check ransomware reporting, wallet-clustering services (if such service is available for the investigator), exchange attribution sources, and sanctions lists. We can use these results to assess whether the wallet is linked to known campaigns, related wallets, exchanges, or restricted entities.

Cryptocurrency-address enrichment is most useful when combined with endpoint, communication, and incident-timeline evidence rather than interpreted from the wallet alone.

4.5 CVE / Vulnerability

Determine exploitation likelihood and prioritize patching

4.5.1 Meaning and use

From a forensic analyst's perspective, CVE data enrichment helps us judge how relevant a vulnerability is to an investigation and how urgently it should be addressed. Intuitively, the most useful fields are the CVE description, CVSS, affected products, public exploit availability, active exploitation, KEV status, patch availability, ATT&CK mapping, known threat-actor use, and EPSS score. The main caveat is that severity alone is not enough: a high CVSS may matter less than active exploitation or confirmed exposure in the environment.

4.5.2 Typical artifacts containing data

On Windows systems, the CVE identifier itself may appear in vulnerability scanner results, EDR findings, patch-management tools, SIEM alerts, ticketing systems, vendor advisory references, and incident notes. It is understandable that CVE details are not to be expected to occur in native endpoint artifacts or in their parsed version. We usually correlate CVE data with installed software, version evidence, patch state, and exploitation traces from logs, process activity, or forensic artifacts on the affected host. The strongest value comes from linking the CVE to an actually vulnerable asset and any signs of exploitation.

4.5.3 Ways and tools for enrichment

When gaining additional data on CVEs, it is advisable to start with NVD, CVE.org, MITRE, and vendor advisories to confirm the description, scoring, affected versions, and remediation guidance. We can then check sources such as ExploitDB, Metasploit, GitHub, KEV, threat-intelligence reporting, and EPSS to understand exploitability and real-world risk.

CVE enrichment is most useful when combined with asset inventory, patch management solutions, and evidence from investigated incident itself.

4.6 Digital Certificate

Identify malicious infrastructure, detect certificate abuse.

4.6.1 Meaning and use

Certificate enrichment helps the forensic analyst to understand what infrastructure a certificate belongs to and whether it looks routine, misconfigured, or suspicious. Fields such as subject, SANs, issuer, validity period, serial number, fingerprint, signature algorithm, key type, revocation status, CT log presence, self-signed status, free-CA use, and reuse across other hosts are useful for clustering related systems, identifying exposed services, and spotting weak trust or abandoned infrastructure. The main limitation is that certificates are strong infrastructure indicators but not proof of intent on their own, especially when shared hosting, CDNs, or automated certificate issuance are involved.

4.6.2 Typical artifacts containing data

On Windows systems, certificate details can appear in browser artifacts, Windows certificate stores, Schannel/TLS-related logs, EDR telemetry, web proxy logs, email client artifacts, memory captures, and packet captures containing TLS handshakes. We also sometimes recover certificate subjects, SANs, fingerprints, or issuer names from malware configs, downloaded files, PowerShell scripts, and forensic data of network connections. Some fields, such as CT log history, revocation status, or other hosts using the same certificate, usually come from external enrichment rather than native endpoint artifacts.

4.6.3 Ways and tools for enrichment

It is typical to extract certificate metadata with OpenSSL or similar tooling, then pivot to sources such as crt.sh, Censys, and Shodan for CT history and certificate reuse across infrastructure. We review issuer, validity, revocation, fingerprint, signature algorithm, and key properties, then correlate those findings with domains, IPs, and endpoint evidence.

Based on our team’s experience, certificates are most useful when analyzed alongside related host, domain, and network artifacts rather than in isolation.

4.7 DNS Query

Detect C2 beaconing, DNS tunneling, and DGA domains.

4.7.1 Meaning and use

We use DNS query enrichment to help investigators decide whether name-resolution activity is routine, suspicious, or clearly linked to malware behavior. In our team's experience, the most useful elements are the query type, the queried domain's reputation and infrastructure, the returned answer, query frequency, tunneling indicators, and the requesting process. The main disadvantage is that a single unusual query is rarely enough on its own; the pattern over time and the process behind it usually matter most. Therefore, amount of forensic evidence necessary to enable us to provide a legitimate conclusion might be relatively large and from various sources.

4.7.2 Typical artifacts containing data

Details related to DNS queries are commonly recovered from DNS client logs, Sysmon Event ID 22, EDR telemetry, packet captures, proxy logs, and sometimes memory captures. The query name, type, and response may be native to the artifact, while domain reputation and tunneling assessments are analyst-added enrichments. The strongest value comes from tying the DNS activity to a specific process, user session, and follow-up network connection.

4.7.3 Ways and tools for enrichment

Enrichment workflow should start by reviewing the query type and returned data. Then, investigators or proposed enrichment model shall apply full domain enrichment to the queried name. We can check frequency and volume in SIEM or DNS analytics, look for tunneling patterns such as long or high-entropy subdomains and unusual record types, and, where available, identify the requesting process from Sysmon or EDR.

DNS-query enrichment is most useful when combined with process, network, and timeline evidence.

4.8 Email Address

Trace phishing campaigns, validate sender legitimacy, link personas.

4.8.1 Meaning and use

From a forensic analyst's perspective, email-address enrichment helps us move from a simple sender or recipient string to a more complete understanding of legitimacy, provenance, exposure, and investigative relevance. The domain portion of the address is often the first and most valuable pivot, because applying full domain enrichment can reveal ownership, age, mail

infrastructure, reputation, and phishing or malware associations. Mailbox existence validation, breach-database presence, associated accounts or personas, and threat-intelligence associations can help us assess whether the address is real, previously exposed, operationally important, or linked to fraud or intrusion activity. Header-based results such as SPF, DKIM, DMARC, and sender IP extraction are especially important in email investigations because they help us distinguish between legitimate sending infrastructure, forwarding artifacts, and spoofing attempts. We also place strong value on phishing-report history, particularly when the same address or domain appears repeatedly in internal reporting or public reporting channels. At the same time, these fields should be treated carefully: mailbox verification may be blocked or misleading, breached-email presence does not by itself imply malicious intent, persona-linking can produce false positives, and SPF, DKIM, and DMARC outcomes must be interpreted in the context of forwarding, third-party senders, and mailing services rather than as standalone proof.

4.8.2 Typical artifacts containing data

When we investigate Windows endpoints, we most commonly recover email addresses from Outlook OST or PST files, Windows Mail artifacts, MBOX or EML exports, attachment metadata, contact stores, autocomplete caches, and message header data preserved in emails themselves. We also see them in browser history from webmail access or autofill data, Teams or collaboration artifacts, chat exports, CRM exports, downloaded CSV files, clipboard artifacts, and user documents such as spreadsheets, address books, and case notes.

In phishing and account-compromise cases, email addresses often appear in SMTP headers, security gateway logs, SIEM alerts, PowerShell scripts (false positives appear in publicly available modules and scripts), browser-saved form data, password-manager artifacts, and EDR telemetry tied to email-client or browser processes. Header-derived fields such as SPF, DKIM, DMARC, and sender IP are obtained from the raw message source, not from a simple mailbox view, so preserving full headers is critical during collection. Breach presence, persona linkage, and threat-intelligence associations must be added later by enriching email addresses recovered from those local sources. The strongest evidentiary value comes from correlating the email address with full message headers, user interaction artifacts, any embedded URLs or attachments, and surrounding authentication or network activity.

4.8.3 Ways and tools for enrichment

Email-address enrichment usually begins by normalizing the address and separating the local part from the domain so that all relevant domain and FQDN enrichments can be applied to the domain portion. We then review available email headers to evaluate SPF, DKIM, and DMARC results, parse Received lines to identify the likely sender IP or relay chain and compare those findings with the apparent sender identity. Where appropriate, it is possible to test mailbox existence using SMTP validation methods or commercial validation services, although results should be used cautiously because many mail systems deliberately limit or mask verification behavior. For exposure and identity context, we can check breach sources such as Have I Been Pwned or similar datasets, review OSINT tooling for associated accounts or

personas, and query threat-intelligence platforms such as MISP, OTX, or VirusTotal for known malicious associations. If available in investigated environment, it is advisable to also review internal phishing-reporting systems and email-security platforms to see whether the same sender address has appeared in prior incidents.

The most reliable process is to start with the original message and headers, enrich both the email address and its domain, validate technical sending controls, and then interpret the combined results together with endpoint artifacts, user activity, and any linked infrastructure such as URLs, domains, attachments, or sender IPs.

4.9 File Hash

Identify malware, map to families, assess capabilities and attribution.

4.9.1 Meaning and use

File-hash is one of the most used Indicators of Compromise (IOC). Their enrichment helps us turn a single cryptographic identifier into a much broader understanding of what a file is, how it behaves, how common it is, and whether it is linked to known malicious activity. By enrichment, we might be able to add data fields containing information such as antivirus detection rate, malware family classification, YARA matches, sandbox behavior, network IOCs, ATT&CK mappings, associated threat actor or campaign, and prevalence. These are especially valuable for triage and scoping because they help us distinguish commodity malware, targeted tooling, benign software, and previously unknown samples.

Static attributes such as file type, size, compile timestamp, packer detection, digital signature, import table, embedded strings, fuzzy hashes, and related parent or child hashes help us validate the identity of the sample and understand likely capabilities such as persistence, networking, credential access, or code injection. We also use file-name observations, first-seen and last-seen dates, and prevalence data to understand how the sample has appeared in the wild and whether it is rare in the enterprise. Despite obvious benefits, it is necessary to keep in mind limits of hash-based enrichment: hashes are exact identifiers and will miss related variants, AV naming is inconsistent across vendors, compile times may be falsified, digital signatures may be stolen or abused, sandbox results may vary by environment, and attribution fields should be treated as analytic leads rather than definitive proof.

4.9.2 Typical artifacts containing data

When we investigate file hashes on Windows endpoints, we usually begin with artifacts that preserve executed, downloaded, created, or referenced files. Common sources include, most prominently, the file system itself – by which we mean content of the investigated disk or disk image itself. Certain data might be provided by NTFS metadata, $MFT, $LogFile, USN Journal, Amcache, Shimcache, Prefetch, SRUM, Jump Lists, LNK files, browser download history, browser cache, Outlook attachments, Teams or collaboration downloads, Recycle Bin records,

and quarantine locations from AV or EDR tools. Hashes may also be present directly in EDR telemetry, Windows Defender logs, AV quarantine records, forensic triage collections, SIEM alerts, Sysmon events, PowerShell logs, script block logs, and memory-derived artifacts when files are loaded or referenced in execution chains. In malware cases, our team often correlates the primary sample hash with parent or dropper hashes, child or dropped-file hashes, and related network indicators recovered from sandboxes, endpoint telemetry, or memory analysis. Static characteristics such as file size, signature data, compile timestamp, imports, and embedded strings are to be derived from the recovered file itself, although traces of filenames, paths, or command lines often survive even when the original binary is gone.

The strongest forensic value comes from tying the hash to a specific file path, execution event, user context, and surrounding artifact set.

4.9.3 Ways and tools for enrichment

In suggested workflow, hash enrichment typically starts by querying multi-engine and threat-intelligence platforms such as VirusTotal, Hybrid Analysis, MetaDefender, MISP, and OTX to collect detection names, family classifications, first-seen and last-seen dates, prevalence, fuzzy-hash relationships, and known associations. Except these automated tasks, malware analysts might perform static analysis using tools such as file, ExifTool, PEStudio, sigcheck, Detect It Easy, PEiD, FLOSS, YARA, and where needed disassemblers such as IDA Pro, to determine true file type, metadata, packing or obfuscation, signature status, imports, and notable strings. Of course, these enrichments are not to be included in automated framework, we only include them for completeness of proposed enrichment options.

For behavioral understanding, it is possible to review sandbox outputs from platforms such as Any.Run, Joe Sandbox, Hybrid Analysis, or Cuckoo to identify processes created, files dropped, registry changes, persistence actions, network connections, and ATT&CK techniques. Related parent and child hashes are then used to build execution chains and cluster samples into broader malware sets.

Basic way to enrich hash data is to recover the file or its hash from endpoint evidence, validate the sample's local context (owner, MFT timestamps, evidence of execution, …), enrich it across both static and dynamic sources, and then interpret the combined result together with host artifacts, process lineage, and any associated network activity.

4.10 FQDN / Domain Name

Track C2 infrastructure, phishing domains, and attacker registration patterns.

4.10.1 Meaning and use

From a forensic analyst's perspective, we can use domain and FQDN enrichments to move from a simple hostname or URL to a much fuller picture of ownership, infrastructure, intent,

and risk. Registration-related fields such as WHOIS registrant details, registrar, registration and expiration dates, domain age, and privacy or redaction status are often useful for spotting disposable or suspicious infrastructure, especially when a domain appears newly registered, short-lived, or deliberately obscured. DNS-related fields such as name servers, A/AAAA, MX, TXT, CNAME, passive DNS history, and discovered subdomains help us understand how the domain is hosted, whether it supports email, whether it has anti-spoofing controls, and how its infrastructure has changed over time. We also place significant value on threat-oriented enrichments such as reputation, malware and phishing associations, certificate data, CT logs, screenshots, DGA scoring, typosquatting similarity, and parking or sinkhole status. These often help us distinguish legitimate business infrastructure from phishing, staging, or command-and-control infrastructure. At the same time, our experience is that none of these fields should be treated in isolation: WHOIS may be redacted or outdated, DNS changes quickly, shared hosting can blur attribution, and screenshots or reputation labels only reflect a point in time.

4.10.2 Typical artifacts containing data

When we investigate domains referenced on Windows endpoints, we commonly recover them from browser history, cache, cookies, download records, session files, and typed URL artifacts. We also see domains in DNS cache artifacts, Windows DNS Client logs, firewall logs, Sysmon network events, EDR telemetry, proxy records, VPN client logs, and memory captures, all of which can help us show that a domain was queried, resolved, or contacted from a specific host. In phishing and intrusion cases, our team frequently finds domain evidence in Outlook OST or PST files, email headers, embedded links, attachment metadata, Teams or collaboration artifacts, LNK files, Jump Lists, Office recent file lists, PowerShell history, scheduled task definitions, scripts, and malware configuration data. Certificate-related domain names may also appear in browser certificate stores, the Windows certificate store, Schannel-related artifacts, and live memory during TLS sessions. Some of the most valuable correlations come from tying a domain recovered from endpoint evidence to a specific user action, process execution, or network connection at a specific time. By contrast, fields such as registrar information, passive DNS history, CT history, DGA scoring, or domain categorization usually are not stored locally on the endpoint and instead must be added later through enrichment.

4.10.3 Ways and tools for enrichment

In our workflow, this enrichment is usually performed by combining registrar, DNS, certificate, web-observation, and threat-intelligence sources. Our analysts typically begin with WHOIS and registrar lookups to identify registrant details, registration timing, expiration, and privacy status. We then query current DNS records using tools such as dig or nslookup to collect NS, A, AAAA, MX, TXT, and CNAME data, and we pivot to passive DNS sources such as DNSDB, PassiveTotal, or SecurityTrails to understand historical hosting and IP changes. For broader infrastructure discovery, we use certificate transparency and subdomain enumeration sources such as crt.sh, Censys, Amass, and Subfinder. Reputation and abuse context are added from sources such as VirusTotal, OTX, IBM X-Force, ThreatFox, URLhaus, PhishTank, OpenPhish, and Google Safe Browsing. When needed, we can also review SSL/TLS details, web technologies,

and rendered page screenshots using platforms such as Shodan, SSL Labs, Wappalyzer, BuiltWith, and urlscan.io. For suspected deception, we may add DGA scoring and typosquatting analysis with tools such as dnstwist or URLCrazy.

The most defensible results should come from extracting the domain from endpoint artifacts first, then enriching it across multiple independent sources and interpreting the combined picture rather than relying on any single indicator.

4.11 Geolocation

Detecting impossible travel, unauthorized access origins.

4.11.1 Meaning and use

Geolocation enrichment helps us decide whether a location is expected, suspicious, or likely obscured. In our team's experience, the most useful elements are the derived country, region, and city, whether the location matches a known VPN or proxy exit, whether it creates an impossible-travel pattern, and whether the country is considered high risk under policy. The main caveat is that geolocation is approximate and often reflects network infrastructure rather than the user's true physical location.

4.11.2 Typical artifacts containing data

On Windows systems, location clues are often recovered from login records, VPN logs, browser artifacts, email headers, mobile-device backups, wireless profiles, application telemetry, EDR data, and cloud sign-in logs rather than from classic local artifacts alone. The raw IP or coordinates may be present in those records, but VPN-exit identification, impossible-travel logic, and country-risk classification are usually analyst-added enrichments. In practice, the strongest value comes from tying the location to a specific account, device, session, and timeline.

4.11.3 Ways and tools for enrichment

GeoIP or GPS-derived location data are a good starting point of enrichments efforts. Then we might check whether the endpoint matches known VPN or proxy infrastructure. We compare the location with prior sign-ins to test for impossible travel and then assess the destination country against internal (client´s) risk policy or sanctions-related guidance. Geolocation enrichment should be combined with authentication, device, and network evidence rather than interpreted from location data alone.

4.12 Log Source / Facility

Ensure log integrity and proper timeline reconstruction.

4.12.1 Meaning and use

Log Source enrichment helps us understand what produced a record and how much trust to place in its timestamps and fields. In our team's experience, the most important elements are device type and vendor, parser or normalization method, retention period, and timezone. These determine how we interpret the data, whether important fields may have been transformed, how far back we can investigate, and whether timestamps need adjustment for timeline work.

4.12.2 Typical artifacts containing data

On Windows systems, this information is usually not stored inside the endpoint artifact itself unless the source is a local Windows log. More often, it comes from SIEM onboarding records, parser definitions, collector settings, device configuration, and log-management documentation. We often correlate it with EVTX metadata, collector timestamps, NTP settings, and ingestion delays to explain gaps or time skew during analysis. It is understandable that this type of enrichment is rather difficult to automize in a general way.

4.12.3 Ways and tools for enrichment

During forensic analysis we usually confirm the source type and vendor from SIEM or logging configuration, review which parser or normalization pipeline handled the records, check the retention policy, and verify the source time zone or NTP configuration. Based on our team's experience, this enrichment is essential for defensible timeline analysis and for avoiding misinterpretation caused by parser issues or timestamp offsets.

4.13 MAC Address

Track physical devices, detect spoofing and rogue devices

4.13.1 Meaning and use

From a forensic analyst's perspective, MAC-address enrichment helps us identify what kind of device we are dealing with and where it appeared on the network. The most useful elements are OUI vendor lookup, associated IPs, device-type inference, switch port and VLAN, and whether the address looks randomized. The main disadvantage is that OUI-based identification is only approximate, and randomized or spoofed MACs can reduce confidence.

4.13.2 Typical artifacts containing data

On Windows systems, MAC addresses commonly appear in registry network-interface data, ARP cache artifacts, ipconfig /all output, DHCP-related artifacts, EDR telemetry, memory captures, and packet captures. MAC address is recorded in JumpList artifact as well, which is easily observed especially if the artifact contains evidence of file access or other activity associated with remote devices, besides accessing local resources. Associated IPs are often better reconstructed from DHCP, ARP, and switch records than from the endpoint alone, while switch port and VLAN details usually come from network infrastructure rather than native host artifacts. The strongest value comes from tying the MAC to a specific host, time period, and network segment.

4.13.3 Ways and tools for enrichment

In our workflow, we usually start with an OUI lookup to identify the likely vendor, then correlate the MAC with DHCP leases, ARP tables, and switch CAM tables to recover associated IPs and physical network placement. We also check whether the locally administered bit suggests a randomized MAC and use that when judging attribution confidence. Based on our team's experience, MAC enrichment is most useful when combined with DHCP, endpoint, and switch evidence rather than interpreted on its own.

4.14 MITRE ATT&CK ID

Map adversary behavior to framework, guide detection and response.

4.14.1 Meaning and use

ATT&CK-ID enrichment helps us turn a technique reference into something operational for investigation and detection. The most useful elements are the official MITRE technique description, tactic, real-world procedure examples, recommended data sources, related Sigma or YARA content, and mitigations. The main caveat is that an ATT&CK technique is an analytic framework, not evidence by itself, so it should guide and augment standard investigation. It definitely does not replace artifact-based findings.

4.14.2 Typical artifacts containing data

On Windows systems, the ATT&CK ID itself cannot be found in endpoint artifacts unless it was added by a security product, alert, detection rule, case note, or report. Analyst can identify the underlying behaviors in logs and artifacts such as process creation, registry changes, scheduled tasks, services, PowerShell activity, network connections, authentication events, and file activity. The real value comes from mapping those observed artifacts back to the ATT&CK technique to structure analysis and explain what the activity may represent.

4.14.3 Ways and tools for enrichment

In standard forensic analysis, we would start with the MITRE ATT&CK entry to confirm the technique name, description, tactic, data sources, and mitigations, then review known procedure examples and related detection content such as Sigma, Elastic, or YARA rules. We could use that mapping to decide which data from parsed artifacts, and which related events to correlate. ATT&CK enrichment would be reliable and useful when it supports evidence-driven analysis and detection engineering, it cannot be used in isolation.

4.15 Mutex Name & Named Pipe

Identify malware families and detect concurrent infections.

4.15.1 Meaning and use

Mutex and named-pipe enrichment helps us decide whether an object name is routine application behavior or a sign of malware, tooling, or post-exploitation activity. In our team's experience, the key value is in checking whether the mutex or pipe has a known malware association, maps to a known C2 framework, or is instead commonly created by legitimate software or standard Windows services. The main caveat is that names can be reused, randomized, or intentionally chosen to look benign, so the name alone is only a clue and not proof of maliciousness.

4.15.2 Typical artifacts containing data

Mutex and pipe names are most often recovered from memory captures, EDR telemetry, sandbox reports, Sysmon if configured, handle enumeration, process analysis, and malware reverse-engineering output. We also sometimes find them in forensic triage collections, strings extracted from binaries, debugger traces, and tool-specific logs. Named pipes can sometimes be found in System.evtx, as they might be "installed" as a service. Strongest value comes from tying the mutex or pipe to the creating process, execution time, loaded modules, and any related network or persistence artifacts.

4.15.3 Ways and tools for enrichment

The observed name of a mutex or a named pipe should be compared against malware reports, VirusTotal, sandbox results, Sigma content, threat-intelligence references, C2 framework documentation, and Microsoft or vendor documentation for legitimate software. We then check whether the object is a known Windows or application artifact, or whether it better fits common malware or C2 naming patterns. In general, mutex and pipe enrichment is most useful when combined with process, memory, and file evidence rather than interpreted on the object name alone.

4.16 Network Flow

Identify data exfiltration, lateral movement, and C2 communication

4.16.1 Meaning and use

Netflow enrichment helps us understand who communicated with whom, for how long, how much data moved, and whether the pattern looks routine or suspicious. Elements like source and destination IP context, bytes in and out, session duration, beaconing behavior, JA3 or JA3S fingerprints, and geo-destination are most useful in forensic investigations (for example, to detect data exfiltration). The main downside is that a single flow is rarely conclusive on its own, so the strongest conclusions come from patterns across time and correlation with host evidence.

4.16.2 Typical artifacts containing data

On Windows systems, these details are commonly recovered from firewall logs, EDR telemetry, Sysmon network events, proxy logs, packet captures, NetFlow records, Zeek data, and sometimes memory captures. Some fields, such as source and destination IPs, timing, and data volume, are native to flow artifacts, while beaconing analysis, geolocation, and TLS fingerprint interpretation are usually analyst-added enrichments. In practice, the strongest value comes from tying the flow to a specific process, user session, and related DNS or authentication activity.

4.16.3 Ways and tools for enrichment

Source and destination IPs should be enriched first, possibly using methods described in this document. Then, review bytes transferred, duration, and whether the timing suggests beaconing. Where available, we compare JA3 or JA3S fingerprints against known tools or malware and check whether the destination geography is expected for the asset or user. Based on our team's experience, network-flow enrichment is most useful when combined with endpoint, DNS, and timeline evidence.

4.17 Network Port / Protocol

Detect protocol anomalies, tunneling, and unauthorized services.

4.17.1 Meaning and use

Port and protocol enrichment helps us decide whether observed network activity is routine, suspicious, or clearly inconsistent with its claimed purpose. The most useful elements are the standard service assignment for the port, known malware use of that port, whether the observed protocol matches what should normally run there, and whether the port is allowed

by policy. The main issue to watch out for is that port numbers alone are weak indicators, because attackers routinely tunnel unexpected protocols over common allowed ports.

4.17.2 Typical artifacts containing data

On Windows systems, port and protocol details are commonly recovered from firewall logs, Sysmon Event ID 3, EDR telemetry, packet captures, proxy logs, netstat output, memory captures, and application logs. The port number is often native to the artifact, while service mapping, malware associations, and firewall-policy context are analyst-added enrichments. In practice, the strongest investigative value comes from tying the port to a specific process, destination, and observed traffic pattern.

4.17.3 Ways and tools for enrichment

In our workflow, we usually start by mapping the port to its standard IANA service, then compare that with the actual observed protocol using packet inspection, Wireshark, IDS/IPS, or similar telemetry. We also review known malware use of the port from threat-intelligence sources and check firewall rules or security policy to see whether the traffic should have been permitted. Results of port enrichment are most useful when combined with process, host, and network context.

4.18 Person Name

Attribution, insider threat investigation, OSINT profiling.

4.18.1 Meaning and use

Person name enrichment helps us assess whether a name can be tied to real-world accounts, contact points, public records, domains, or internal access. In our team's experience, the most useful elements are social profiles, linked email addresses, public-record data, domain registrations, and organizational role or access. The main caveat is that names are ambiguous (how many "Ján Novák"s are there in Slovakia?) and can produce false matches, so identity conclusions should be based on multiple corroborating data points rather than the name alone.

4.18.2 Typical artifacts containing data

Person names are often recovered from emails, chat logs, documents, browser history, contact lists, HR exports, account profiles, Teams artifacts, spreadsheets, screenshots, and various case notes (for example, interviews with stakeholders, conducted during an IR). Internal role and access details may also appear in directory exports, IAM reports, and endpoint artifacts tied to user profiles or login activity. Most OSINT-style enrichments such as social profiles, public records, linked emails, and reverse-WHOIS results are usually added after collection rather than stored natively on the endpoint.

4.18.3 Ways and tools for enrichment

It is usual to start with OSINT tools and public-record sources to identify likely social accounts, linked emails, public records, and domain registrations, then compare those findings with internal HR, IAM, and Active Directory data when relevant. We look for overlaps in name, email, employer, role, or other identifiers to reduce false positives. Person name enrichment is most useful when combined with account, communication, and organizational evidence.

4.19 Phone Number

Investigate vishing, SMS phishing, and social engineering.

4.19.1 Meaning and use

From a forensic analyst's perspective, phone-number enrichment helps us assess whether a number looks ordinary, disposable, spoofable, or whether it is already linked to fraud. In our team's experience, the most useful parameters of a phone number, that we would add to the enrichments pool, are the carrier, number type, country or region, spam-report history, and any public identity associations. The main caveat is that phone numbers are easy to spoof, reassign, or temporarily rent, so they are useful leads but not strong attribution on their own.

4.19.2 Typical artifacts containing data

On Windows systems, phone numbers are often recovered from emails, chat logs, browser history, contact files, mobile-device backups, CRM exports, Office documents, screenshots, clipboard artifacts, and collaboration tools. The number itself may be present in endpoint artifacts, but carrier details, VoIP classification, scam reports, and OSINT identity links need to be added by an analyst. In practice, the strongest value comes from tying the number to a message, call record, account, or user interaction.

4.19.3 Ways and tools for enrichment

Enrichment process can start with carrier and HLR-style (HLR stands for Home Location Register) lookups to identify provider and number type, then use numbering-plan references to confirm country and region. We also check community scam-report sources and public-record or OSINT tools for any linked identities or accounts. As observed with previous artifacts and enriched data, phone-number enrichment is most useful when combined with communication artifacts, local and domain account information, and timeline context.

4.20 Private IP Address

Map internal compromise scope, identify affected systems and segments.

4.20.1 Meaning and use

These enrichments are used to turn a private IP address into a concrete understanding of the internal asset behind it: what device it is, where it is in the network, who uses it, how it authenticates, how critical it is, and who is responsible for it. Fields to augment the IP itself such as hostname, MAC address, DHCP history, subnet, gateway, VLAN, and DHCP-versus-static assignment are essential for identifying the endpoint and placing it correctly within the internal network. Device role, operating system, AD domain membership, logged-on users, installed security agents, and authentication type help establish how the system is used and whether it fits expected enterprise patterns. Asset criticality, last vulnerability scan result, physical-versus-virtual status, and responsible owner are especially valuable for incident scoping and prioritization, because they indicate business impact and response paths. The main investigative limitation is that private IPs are reused, reassigned, and visible only within internal context, so a private IP alone is rarely sufficient evidence unless it is tied to time-bounded DHCP, authentication, and endpoint telemetry; similarly, CMDB and asset inventories may be outdated, and NAT, VPN, and multi-homed hosts can complicate attribution.

4.20.2 Typical artifacts containing data

On Windows endpoints, these data points can often be found or corroborated in a range of forensic artifacts. Hostname, domain membership, and OS version commonly appear in the SYSTEM registry hive, (more precisely in SYSTEM\CurrentControlSet\Control\ComputerName, SYSTEM\CurrentControlSet\Control\Services\Tcpip\Parameters), setup and update logs, EDR metadata, and WMI repository outputs. Network details such as private IP, gateway, subnet mask, DHCP enablement, DHCP lease data, and DNS suffixes may be present in registry locations under network interface keys, ipconfig /all captures, DHCP client operational logs, firewall logs, memory captures, and EDR telemetry. MAC addresses may be recoverable from ARP cache artifacts, interface configuration, registry keys dedicated to networking, and volatile memory, while DHCP lease history is often better preserved on DHCP servers than on the endpoint itself. Logged-on users and authentication type can be reconstructed from Windows Security logs such as 4624 and related logon events, Terminal Services logs, Sysmon, RDP artifacts, cached logon data, user profile directories, and EDR timelines. Installed security agents may appear in uninstall registry keys, services, scheduled tasks, driver listings, program files, SCCM records, and EDR inventories. Information such as VLAN, switch context, asset criticality, last vulnerability scan result, and responsible owner is usually not native to the endpoint and instead must be correlated from enterprise artifacts such as CMDB, SIEM, scanner consoles, switch CAM tables, IPAM, ServiceNow, and Active Directory.

4.20.3 Ways and tools for enrichment

The enrichment is performed by correlating endpoint, directory (Active Directory, in MS environment), network, and asset-management sources. Identity and naming are usually established from DHCP logs, internal DNS, Active Directory, LDAP, and CMDB records. Addressing and network-placement details come from IPAM, DHCP server data, switch

configuration, ARP and CAM tables, routing and gateway documentation, and sometimes endpoint registry or EDR snapshots. Host posture is enriched through EDR platforms, SCCM, MDM, vulnerability scanners such as Nessus, Qualys, or Rapid7, and asset inventories that record OS, agent deployment, and patch status. User and authentication context is derived from Windows Event Logs, AD logs, RADIUS or other authentication systems, SIEM correlation, and remote access telemetry. Finally, ownership and criticality are confirmed through CMDB, risk registers, virtualization platforms, cloud consoles, and ticketing systems such as ServiceNow. In practice, the strongest result comes from building a time-aware mapping from private IP to DHCP lease, MAC, hostname, user, and asset record, then validating that mapping against endpoint artifacts and network infrastructure logs.

4.21 Process Name

Detect LOLBins abuse, masquerading, and malicious process execution.

4.21.1 Meaning and use

From a forensic analyst's perspective, process-name enrichment helps us decide whether a running or recorded process is normal, suspicious, or potentially masquerading as something legitimate. The most useful fields to add during enrichment are the known legitimate application match, expected path, expected parent, LOLBin classification, command line, signature status, executable hash, network activity, and prevalence in the environment. A familiar process name alone means very little; the real value comes from checking whether it ran from the right location, with the right parent, with expected arguments, and in a way that is common for that environment.

4.21.2 Typical artifacts containing data

On Windows systems, process execution details are commonly recovered from Sysmon, Security logs when the process auditing is configured (Event ID 4688), EDR telemetry, Prefetch, Amcache, Shimcache, SRUM, RAM captures, and the file system itself. Command lines and parent-child relationships are especially well preserved in Sysmon Event ID 1 and many EDR platforms, while network connections are often available in Sysmon Event ID 3, firewall telemetry, or EDR records. Signature status and hashes usually come from the binary on disk or security-tool metadata, and prevalence is generally an enterprise enrichment from EDR or SIEM - not a native endpoint artifact.

4.21.3 Ways and tools for enrichment

In an enrichment model, we could start by validating whether the process name maps to a known legitimate application using references such as Microsoft documentation, NSRL, or EchoTrail. We then compare the observed file path, parent process, and command line against expected patterns, check whether the binary is a known LOLBin, verify its signature and hash, and review any network connections it initiated. Based on our team's experience, process-

name enrichment is most reliable when tied to process tree context, file metadata, and surrounding host activity.

4.22 Public IP Address - external

Identify attacker infrastructure, hosting, geolocation, and reputation.

4.22.1 Meaning and use

These enrichment fields are used to turn a raw IP address into actionable data. DNS and passive DNS enrichments, such as reverse DNS, forward DNS, and historical resolutions, help identify hostnames, co-hosted domains, and infrastructure changes over time. Registration and routing enrichments, including WHOIS, CIDR, ASN, ASN organization, and BGP history, show who controls the address space, what network it belongs to, and whether routing behavior is normal or suspicious. Infrastructure and service enrichments, such as geolocation, hosting type, open ports, HTTP banners, and SSL/TLS certificates, help profile the system's role and exposure. Finally, security-focused enrichments like Tor/VPN flags, reputation scores, blacklist presence, malware family links, threat actor associations, timestamps, and OSINT references support attribution, prioritization, and incident response.

4.22.2 Typical artifacts containing the data

On a Windows endpoint, public IP addresses that become candidates for this kind of enrichment can often be recovered from standard forensic artifacts such as browser history and cache, DNS cache, Windows Firewall logs, Sysmon network connection events, Security event logs, PowerShell logs, EDR telemetry, memory captures, prefetch-linked execution context. Another sources are email client artifacts, RDP artifacts, and application-specific logs from browsers, VPN clients, cloud sync tools, and remote administration software.

4.22.3 Ways and tools for enrichment

The enrichment should be performed by combining several categories of sources rather than relying on a single lookup. Basic ownership and allocation details come from registries and routing sources such as ARIN, RIPE, APNIC, BGPView, Team Cymru, and ipinfo.io. DNS-centered context is gathered through direct tools like dig and nslookup and through passive DNS platforms such as PassiveTotal, SecurityTrails, and DNSDB. Exposure and service fingerprinting are derived from internet scanning and observation platforms such as Shodan, Censys, Nmap, crt.sh, and HTTP header collection tools such as curl. Threat and abuse context is added from intelligence repositories and reputation feeds including AbuseIPDB, VirusTotal, OTX, IBM X-Force, ThreatFox, Spamhaus, and MISP, while broader public references are found in search results, vendor blogs, and published threat reports. In endpoint practice, an examiner may extract a public IP from artifacts such as the Windows DNS Client cache, Event ID 3 records from Sysmon, firewall log entries in pfirewall.log, browser SQLite databases, Outlook metadata, RDP cache and jump list evidence, or strings and socket structures in RAM,

then pivot that IP through the enrichment sources to determine ownership, exposure, reputation, and historical associations.

Taken together, these enrichments provide a layered view of a public IP address across ownership, infrastructure, behavior, exposure, and threat relevance. They help a researcher distinguish benign shared hosting from attacker-controlled infrastructure, connect an IP to malware delivery or command-and-control activity, detect anonymization or relay use, and identify escalation paths such as abuse reporting or provider notification. The Windows artifacts add important evidentiary grounding because they show where and when the endpoint truly observed or communicated with that public IP, which can support timeline building, user activity reconstruction, and correlation with external intelligence. At the same time, there are important caveats: geolocation is approximate, passive DNS and scan data are time-sensitive, reputation scores can be noisy, and cloud or CDN infrastructure may serve both legitimate and malicious traffic. Forensic conclusions are strongest when the public IP is corroborated across multiple local artifacts and multiple enrichment categories.

4.23 Public IPs, owned by an organization itself

4.23.1 Meaning and use

These enrichment fields are used to understand a public IP address that belongs to the organization itself, with the goal of determining what the organization is intentionally exposing to the internet, who owns operational responsibility for it, and whether it presents security or compliance risk. Service-related fields such as published internet-facing services, associated FQDNs, SSL certificate details, firewall rule summaries, and WAF or DDoS protection status help define the system's external attack surface and how it is protected. Ownership and governance fields, including assigned business unit or department and change management history, provide internal accountability and operational context. Security posture fields such as vulnerability scan findings, patch and compliance status, and blacklist or reputation status help assess whether the organizational IP is properly maintained, exposed unnecessarily, or already attracting abuse-related attention.
Enrichments give investigators a complete picture of an organizational public IP across exposure, ownership, protection, hygiene, and operational change. They are especially useful during incident response, attack surface reviews, and breach scoping because they help determine whether a vulnerable or abused IP is part of an approved service, whether it is adequately protected, which team must respond, and whether recent changes might explain new behavior. They also help identify gaps between intended and actual exposure, such as undocumented services, expired certificates, overly broad firewall rules, missing patches, or blacklist listings affecting reputation. The main caveat is that no single source is sufficient: internal inventories may be outdated, external scans may be incomplete or time-sensitive, and security findings must be interpreted considering business purpose and recent change history.

4.23.2 Typical artifacts containing the data

On a Windows endpoint, public IP addresses that become candidates for this kind of enrichment can often be recovered from forensic artifacts such as browser history and cache, DNS cache, Windows Firewall logs, Sysmon network connection events, Security event logs, PowerShell logs, EDR telemetry, memory captures, prefetch-linked execution context, email client artifacts, RDP artifacts, and application-specific logs from browsers, VPN clients, cloud sync tools, and remote administration software.

4.23.3 Ways and tools for enrichment

Enrichment is typically performed by combining internal asset and governance records with external observation data. Internal sources such as CMDB, IPAM, asset inventory, firewall management platforms, PKI systems, SCCM, BigFix, Tanium, ServiceNow, Jira, and change management databases provide authoritative information about ownership, configuration intent, patch status, and approved changes. External and hybrid sources such as Shodan, Nmap, vulnerability scanners, external DNS, certificate transparency logs, Qualys SSL Labs, MXToolbox, Spamhaus, AbuseIPDB, Cloudflare, Akamai, and AWS Shield help validate what is visible from the internet and whether the organization's controls are working as expected. In practice, the analysis compares what should be exposed according to internal records with what is exposed and observable from outside.

4.24 Registry Key/Value

Identify persistence mechanisms and configuration changes

4.24.1 Meaning and use

Registry-key enrichment might help an analyst to determine whether a key is normal Windows behavior, a persistence point, or a suspicious configuration change. In our team's experience, the most useful fields are the key's legitimate purpose, whether it is a known autostart location, whether the current value matches the expected default, when it was last modified, and which user SID it belongs to. The main caveat is that many legitimate applications also use common persistence locations, so the value and context matter more than the key path alone.

4.24.2 Typical artifacts containing data

Details about registry keys and values come directly from registry hives such as NTUSER.DAT, USRCLASS.DAT, SYSTEM, SOFTWARE, and related transaction logs. Last-write times are preserved in the hive metadata, and the associated SID can usually be inferred from the hive path or profile mapping (when it comes to user registry hives NTUSER.DAT and USRCLASS.DAT). It is possible to correlate registry findings with Prefetch, Amcache, scheduled tasks, services, LNK files, and EDR telemetry to confirm whether a suspicious registry value actually led to execution.

4.24.3 Ways and tools for enrichment

It is possible to compare the key path and value against Microsoft documentation, forensics references/literature, known Autoruns, and baseline systems (for example with a golden image of an endpoint in the organization) to determine expected behavior. We can then classify whether the key is part of a known persistence mechanism, review its last modified time, and map it to the owning user or system context. Based on our team's experience, registry enrichment is most reliable when the key is analyzed together with execution artifacts and surrounding timeline evidence.

4.25 Scheduled Task / Cron

Identify persistence mechanisms and unauthorized automation.

4.25.1 Meaning and use

Enriching the scheduled task helps us decide whether a task is normal administration or a persistence mechanism. In our team's experience, the most useful elements are the task name, command, creator, trigger, and running account, because they quickly show intent and privilege. The main caveat is that many legitimate tools also use scheduled tasks, so the command and context matter more than the task name alone.

4.25.2 Typical artifacts containing data

On Windows systems, details are usually recovered from Task Scheduler entries, task XML files, the Tasks folder, registry references (TaskCache registry key), Autoruns output, Sysmon, and related (not only task scheduler) event logs. We often correlate the task with Prefetch, Amcache, process-creation logs, scripts, and file-system artifacts to confirm whether it really executed and what it launched. In practice, the strongest value comes from tying the task to the creating user, the payload, and the execution timeline.

4.25.3 Ways and tools for enrichment

Review the task definition to identify the executable or script, creation time, author, trigger conditions, and run context, then compare those against known-good administrative patterns. We also check whether the task runs as SYSTEM or another privileged account and whether the command points to suspicious paths or scripts. Based on our team's experience, scheduled task analysis and enrichment should be combined with execution and persistence artifacts to provide most precise and conclusive results.

4.26 URL

Analyze phishing lures, payload delivery, and redirect chains.

4.26.1 Meaning and use

From a forensic analyst's perspective, URL enrichments help us move from a raw URL reference and understand what a user or process was being directed to, whether the destination was benign or malicious, and how the content accessed by a user behaved at the time of access. Fields of enrichment might contain URL reputation, HTTP response code, final redirect destination, content type, page title, embedded objects, shortener expansion, archive snapshots, and rendered screenshots. These would be particularly useful in phishing, malware-delivery, and user activity investigations. They help us determine whether a URL was live (active), whether it redirected through multiple layers, whether it ultimately led to a credential-harvesting page or file download, and whether the visible page matched the apparent intent of the link. Domain extraction is especially important because many investigative pivots happen at the domain level rather than the full URL level, allowing us to connect a single link to broader infrastructure, reputation, and historical activity. Main limitations are that URL content is highly time-sensitive, redirects and payloads can change quickly, some pages serve different content by geography or user agent, and reputation results may lag behind live abuse. For that reason, a URL should be treated as a time-bound artifact, and therefore, investigators should try to preserve and investigate both the original string and the observed behavior as early as possible.

4.26.2 Typical artifacts containing data

When we investigate Windows endpoints, we commonly recover URLs from browser history, cache, cookies, download records, session restore files, and registry artifacts - TypedURL. We also find them in Outlook OST or PST files, email headers, chat and collaboration artifacts, Office documents with embedded links, PDF metadata, Teams messages, clipboard artifacts, LNK files, Jump Lists, PowerShell history, RunMRU entries, scheduled tasks, and script files.

URLs frequently appear in EDR telemetry, proxy logs, firewall logs, Sysmon events, Windows Event Logs, memory captures, in Prefetch, and malware configuration or staging files. Shortened URLs may be preserved in emails or chat messages even when the browser later records only the expanded destination, which is why we try to collect both the original referral source and the actual network destination. Visual elements such as the rendered page, title, and downloads are usually not fully preserved in native endpoint artifacts unless supported by browser cache, screenshots, memory, or external telemetry, so some URL-related context must often be reconstructed after collection. The strongest evidentiary value comes from correlating the URL with a user action, a used browser, a redirect chain, and any resulting download or authentication event.

4.26.3 Ways and tools for enrichment

In suggested workflow, URL enrichment usually starts by parsing the URL into its components and extracting the domain so that various domain and FQDN enrichments can be applied. We then query reputation and scanning services such as VirusTotal, urlscan.io, and Google Safe Browsing to determine whether the URL has already been classified as malicious, phishing,

suspicious, or clean. To understand live behavior, our analysts typically inspect the HTTP response code, follow redirects to the final destination, confirm the served content type, and review page title and metadata using tools such as curl, web fetch utilities, wget, BeautifulSoup, and urlscan.io. Where needed, enrichment model should be able to examine embedded objects, scripts, iframes, and downloads through urlscan.io, sandbox platforms such as Any.Run, or other controlled dynamic-analysis environments. If a shortener is involved, we expand it with services such as CheckShortURL or similar tooling before continuing analysis. For historical context, we may consult archive snapshots such as the Wayback Machine, and for visual confirmation we review rendered screenshots from urlscan.io or browser automation frameworks such as Puppeteer. The most reliable process would be to preserve the original URL, record its observed redirect and content behavior, extract and enrich the underlying domain, and correlate that result with endpoint artifacts and any downstream execution or network activity.

4.27 User Account

Detect compromised accounts, assess privilege escalation, audit access.

4.27.1 Meaning and use

From a forensic analyst's perspective, user-account enrichment in Windows helps us move from a username or SID to a clearer understanding of identity, privilege, risk, and likely impact. Properties such as account type, group memberships, privileged access level, account status, last logon, last password change, MFA status, failed logon count, login anomalies, associated devices, and SPNs are especially important for distinguishing ordinary user activity from administrator, service-account, or potentially compromised behavior.

Organizational context such as department, manager, title, and employment status also matters because it helps us assess whether the observed activity fits the user's expected role and whether the account should even be active. Weight should be placed especially on privileged memberships, stale passwords, missing MFA, suspicious geographical locations or devices, recent lockouts, and SPN assignments because these may indicate elevated compromise risk, ongoing password spraying attack, Kerberoasting exposure, or misuse of dormant accounts. At the same time, assess this data carefully: directory attributes might not reflect current reality, lastLogonTimestamp is not always precise, service accounts may behave differently from human users, and unusual login patterns (for example Impossible login scenarios) are not necessarily malicious unless correlated with device, network, and authentication evidence.

4.27.2 Typical artifacts containing data

When we investigate user accounts on Windows systems, we commonly recover relevant information from Security event logs, especially successful and failed logon events such as 4624, 4625, 4634, 4648, 4672, 4768, 4769, and 4776, along with Sysmon, EDR telemetry, and

SIEM-correlated authentication records. Account names, SIDs, group membership, logon types, source hosts, and timestamps may also appear in registry hives, profile directories, scheduled tasks, services, RDP artifacts, Jump Lists, LNK files, PowerShell logs, remote access clients, cached credentials, and RAM memory captures. On domain-joined systems, AD-related context can often be correlated through logon server information, Kerberos ticket artifacts, token privilege, SPN-related ticket activity, and cached domain data preserved locally or in security tooling. We also frequently use endpoint artifacts to tie an account to associated devices, user sessions, browser activity, executed tools, and VPN or remote-access activity observed in separate logs. Some enrichment fields, such as employment status, manager, formal account type, IAM state, MFA enrollment, and authoritative group membership, usually do not live natively on the endpoint and instead must be added from Active Directory, IAM, HR, PAM, Intune, SCCM, EDR, VPN platforms, and central authentication systems. Strongest evidentiary value comes from correlating directory and IAM data with host artifacts that show when, where, and how the account actually authenticated and what it did afterward.

4.27.3 Ways and tools for enrichment

In our workflow, user-account enrichment typically starts with directory and IAM lookups to establish account type, group memberships, privilege level, account status, password-change timing, SPNs, and organizational attributes. We then pivot to authentication sources such as Windows Security logs, domain-controller logs, SIEM, Azure AD or Entra sign-in logs, VPN telemetry, MFA-provider records, and UEBA (User and Entity Behaviour Analytics) analytics to validate recent logons, failed attempts, anomaly patterns, and remote-access history. Associated-device context is usually built from EDR, SCCM, Intune, AD computer relationships, and endpoint login telemetry, while employment and reporting-line context is confirmed through HR and IAM systems. For privileged or service accounts, we also review PAM data, administrative group assignments, Kerberos service-ticket activity, and SPN registrations to assess misuse or Kerberoasting exposure. Based on our team's experience, the most reliable process is to start with the account identifier, confirm its directory state, validate authentication behavior across available logs, correlate that with endpoint artifacts and device usage (what devices the account signed in to?), and only then draw conclusions about compromise, privilege abuse, or anomalous access.

4.28 User-Agent String

Detect malicious tools, C2 frameworks, and anomalous clients

4.28.1 Meaning and use

User-Agent enrichment helps us to get a more meaningful view of what likely generated a web request and whether that activity fits expected behavior. Ability to add fields such as browser and version, operating system, known malware matches, bot or crawler indicators, and anomaly or forgery detection is especially useful when we are trying to distinguish normal

user browsing from scripted activity, commodity tooling, malware communications, or attempts to disguise traffic. A parsed User-Agent can help us quickly assess whether a request is consistent with the user's workstation and installed software, while known-tool or malware-linked User-Agents can provide valuable leads in intrusion investigations. We also use rare or malformed User-Agent strings as indicators for triage, especially when they do not align with the host's actual platform or appear statistically unusual in the environment. However, User-Agent data should be treated cautiously: it is easy to spoof, some applications use generic or misleading values, security tools may normalize headers, and many legitimate automated services also identify as bots or crawlers. For that reason, we view User-Agent data as a useful contextual signal rather than a standalone attribution artifact.

4.28.2 Typical artifacts containing data

When we investigate incidents in a Windows environment, we most often encounter User-Agent strings in proxy logs, web gateway logs, IIS or application logs, EDR network telemetry, packet captures, browser developer data, and memory captures containing HTTP requests.

On the endpoint itself, User-Agent evidence may also appear in browser cache, browser session artifacts, download history, email security telemetry, PowerShell web request history, script files, malware configuration files, and command-line artifacts tied to tools such as curl, PowerShell, Python requests libraries, or living-off-the-land download utilities. In some cases, we can correlate a User-Agent with a specific browser executable, a scripted process, or a suspicious parent-child process chain using EDR, Sysmon, prefetch, Amcache, Shimcache context, and command execution artifacts.

Windows event log by default does not usually preserve full User-Agent strings in a consistent way, so much of this evidence comes from network-facing telemetry rather than classic local OS logs. In our experience, the most valuable forensic use of a User-Agent is when we can connect it to a specific process, user session, destination URL, and timestamp, allowing us to assess whether the claimed browser and operating system do match the endpoint that made the request.

4.28.3 Ways and tools for enrichment

In suggested model workflow, User-Agent enrichment could start with parsing the raw string using standard UA parser libraries or reference services to derive browser family, browser version, operating system, and device hints. We then compare the string against internal detections, threat intelligence, Sigma-style detection content, malware reports, and documentation for known offensive frameworks to determine whether it matches commonly observed malicious or scripted tooling. For automation analysis, check bot and crawler databases and look for strings associated with search engines, scrapers, headless browsers, or library-based HTTP clients. We can also perform environment-specific anomaly analysis in the SIEM or UEBA platform to determine whether the User-Agent is rare, previously unseen, inconsistent with the source asset population, or obviously fabricated, such as impossible browser and operating system combinations.

Advised analytical approach is to parse the User-Agent, validate it against known host facts such as installed browser version and operating system, compare it with known benign and malicious patterns, and then interpret it alongside other evidence (process, destination, and timing).

4.29 WiFi SSID / BSSID

Detect rogue access points and track physical proximity.

4.29.1 Meaning and use

Wi-Fi SSID and BSSID enrichment helps us determine whether a wireless network is expected in its environment, or is suspicious, or weakly protected. In our team's experience, the most useful elements are whether the SSID or BSSID matches an authorized network, where the access point appears to be located, and what encryption type is in use. The main caveat is that SSIDs can be copied easily, so the BSSID and surrounding context usually matter more than the network name alone.

4.29.2 Typical artifacts containing data

On Windows systems, SSIDs and BSSIDs are often recovered from WLAN profiles, registry artifacts (see artifacts containing IP data), event logs, EDR telemetry, memory captures, wireless adapter data, and command output such as "netsh wlan show interfaces" or "netsh wlan show profile**"**. We also see them in mobile-device backups, screenshots, and user documents in some cases. Geolocation and rogue AP assessment are usually analyst-added enrichments rather than native endpoint data.

4.29.3 Ways and tools for enrichment

We advise comparing the SSID and BSSID against authorized network inventories or WIPS data, check whether the BSSID maps to a plausible physical location, and review the encryption type to assess wireless risk. We also look for signs of impersonation, such as a familiar SSID with an unknown BSSID or weaker security settings. Based on our team's experience, Wi-Fi enrichment is most useful when combined with device, location, and timeline evidence.

4.30 Windows Event ID

Build attack timeline, detect specific adversary techniques.

4.30.1 Meaning and use

From a forensic analyst's perspective, Event ID enrichment helps us understand what a Windows log entry means and how important it is in context. In our team's experience, the key value comes from translating an Event ID into plain meaning, mapping it to possible ATT&CK techniques, identifying related Sigma detections, estimating its forensic importance, and knowing which other Event IDs should be reviewed alongside it. The main disadvantage is that an Event ID rarely proves malicious activity by itself; its meaning depends heavily on log source, surrounding events, host role, and timing.

4.30.2 Typical artifacts containing data

On Windows systems, these details come directly from EVTX files such as Security, System, Application, PowerShell, Sysmon, TaskScheduler, and other operational logs. Event ID is native to the artifact, while ATT&CK mappings, significance ratings, Sigma matches, and other correlations are enrichments added by an analyst. In practice, we usually correlate the event with adjacent records, logon sessions, process activity, and endpoint telemetry to understand what actually happened.

4.30.3 Ways and tools for enrichment

Microsoft and SANS references can be used to confirm the event's meaning, then map it to ATT&CK and review relevant Sigma rules. We also check common companion Event IDs to build a fuller timeline and assign a rough forensic significance based on the investigation type. Event ID enrichment can be most useful when it supports broader event correlation.

4.31 Windows Service

Detect service-based persistence and privilege escalation

4.31.1 Meaning and use

Windows service enrichment helps us decide whether a service is normal system behavior or a likely persistence or privilege abuse mechanism. The most useful additional elements are the display name, binary path, start type, service account, known legitimate status, and signature of the backing binary. The main caveat is that a familiar service name can be misleading, so it is necessary to check the configured path and the context in which the service runs.

4.31.2 Typical artifacts containing data

On Windows systems, these details are commonly recovered from the SYSTEM and SOFTWARE hives (key Services), service configuration in the registry, "sc qc" command output, Autoruns data, EDR telemetry, and the service binary on disk. We often correlate the service

with Prefetch, Amcache, process creation logs (Security.evtx, event ID 4688), and file metadata to confirm what ran and under which account. In practice, the strongest value comes from tying the service definition to execution evidence and the actual binary.

4.31.3 Ways and tools for enrichment

In our workflow, we usually review the service configuration to identify the ImagePath, start mode, run account, and descriptive metadata, then compare the service against known Windows and approved third-party baselines. It is also advisable to compare process execution trees to well-known process trees, such as presented in SANS poster “Hunt evil”. We also verify whether the backing executable is signed and whether it lives in an expected path. Service enrichment must be combined with persistence, execution, and file-trust evidence.

4.32 YARA and Sigma Rule

Scope malware infections and validate detection coverage

4.32.1 Meaning and use

YARA and Sigma rule enrichment helps us understand what a detection is meant to find, how specific it is, and how much weight to place on a match. The most important parameters are the rule name and purpose, the targeted malware family or TTP, match count in the environment, required log sources, ATT&CK mapping, and known false-positive guidance. The main caveat is that a rule hit is an analytic lead, not proof on its own. Therefore, we always judge it in the context of the underlying file, log event, host, and timeline. Verification of a rule match is necessary.

4.32.2 Typical artifacts containing data

On Windows systems, YARA hits are usually tied to files, memory regions, process dumps, malware samples, EDR collections, or retroactive hunts, while Sigma hits are tied to logs such as Security, Sysmon, PowerShell, TaskScheduler, firewall, and other SIEM-ingested sources. **The rule metadata itself is not a native forensic artifact**; what we typically recover is the matched file, process, memory object, or event record, and then add the rule context afterward. In practice, the strongest value comes from linking the rule match to the exact artifact that triggered it and checking whether similar matches appear elsewhere in the environment.

4.32.3 Ways and tools for enrichment

We suggest starting by reviewing the YARA or Sigma metadata to confirm what the rule is intended to detect, which malware family or behavior it targets, what ATT&CK techniques it maps to, and what false positives are expected. For YARA, we then measure match count

across files or memory; for Sigma, we verify that the required log sources and fields exist and review the matching events in context. Rule enrichment is most useful when it supports artifact-driven investigation.

5 Schematic design of a model for enriching digital traces

5.1 Mining data to enrich from parsed data

First of all, it is necessary to define which IOCs will be enriched y our model, and how we can extract them from the parsed evidence. As we mentioned in chapter 3.2, The complexity of enriching a digital artifact – preprocessing, not all data that can be enriched is found directly among the parsed artifacts. So even before the enrichment process itself, we need to make sure that:

1. We know where (in which artifact) to look for data.
2. The way in which we get to the information is defined (either directly by parsing or by additional processing of the parsed data).

In practice, this means that the already automated extraction and parsing of relevant artifacts is ready for post-processing. For example, the enriched data can be tagged with an appropriate tag during or after parsing – for example,IP_PUBLIC, IP_PRIVATE, HOSTNAME, PROCESS_NAME and the like (we will first focus on atomic data that can be categorized as simply as possible). The enrichment model requires this data at the input, and based on it, it runs a designed enrichment process over all data that has a certain tag.

Tagging could also be implemented in the form of a register of all possible enrichments that would be available to the model (as separate enrichment functions).

In case the data for enrichment needs to be calculated, the procedure should be included in the parsing phase of the digital evidence processing. For example, by calculating its hash (MD5 for speed, SHA1 and SHA256 for greater flexibility and security) from each file that is identified through a record in the MFT or is restored using carving.

The ideas described above are just **an example of** how data could be pre-processed for the model. However, we will not develop this in depth within the project, as it depends on the specific data that needs to be enriched. We try to define the model as generally as possible and state the prerequisites – not a specific way to fill in these props in detail.

The individual data to be enriched - the entities that will enter the enrichment model - should meet certain basic parameters.

A suitably designed entity, created from a parsed forensic artifacts, should have fields for its identification, enrichment, tracing of origin (in the original digital evidence) and correlations (with other artifacts and/or entities).

Identity identification

`id` - stable unique identifier, for example ip:8.8.8.8 or hash:<sha256>
`type` - entity type - ako `ip, domain, url, file_hash, user_account, process, registry_key`
`value` – raw value, i.e. IP address, domain, hash, username, etc., see the previous chapter
`normalized_value` – Standardized form usable for machine processing - pairing and deduplication

Context of the parsed artifact

`source_artifact` – where the information comes from, Security.evtx, Amcache.hve, NTUSER.DAT, browser history, ...
`source_record_id` - for consideration, it should be a record number, event record ID, column number in a table, or another reference specific to the artifact
`host` - hostname or the device where the artifact was found
`user` - associated user, if known
`timestamp` - time stamp, associated with the time of observation of the artifact
`artifact_field` - the specific field from which the entity originates, e.g. SourceWorkstationName, DestinationIp, URL, ImagePath, ...

Case and correlation metadata

`case_id` - case ID
`parent_entities` - previous entities that led to this
`related_entities` - already known related entities
`relationship_type` - how the entities are associated, such as `resolved_to, downloaded_from, executed_by, queried_by`

Enrichment status

`attributes` – accumulated enrichment outputs
`enrichment_status` – e.g. not_started, partial, complete, failed
`applied_enrichments` - list of enrichment modules that have already been launched
`pending_enrichments` – modules that are yet to run

Quality and origin of the entity

`confidence` – the trustworthiness of entity extraction and normalization (for example, if the IP address is directly from the Event log, it is more trustworthy than if it was found in another type of source data, for example, using regular expressions)
`provenance` - the source of the entity and the source of each enrichment
`validation_status` - whether the value has been confirmed, derived or unverified
`tags` – optional tags, such as `high_priority, external, ioc, rare, requires_review`

```
base_entity = {
    "id": None,                     # stable unique ID, e.g. "ip:8.8.8.8"
    "type": None,                   # ip, domain, url, file_hash,
user_account, process
    "value": None,                  # raw observed value
    "normalized_value": None,       # normalized/canonical representation

    "source_artifact": None,        # e.g. Security.evtx, Amcache.hve,
Chrome History
    "source_record_id": None,       # row/event/record identifier
    "artifact_field": None,         # parsed field name, e.g. DestinationIp,
URL, UserName
    "host": None,                   # endpoint/server where observed
    "user": None,                   # associated user if known
    "timestamp": None,              # observation time from artifact

    "case_id": None,
    "parent_entities": [],          # entities that led to this one
    "related_entities": [],         # linked entities
    "relationship_type": None,      # optional if created from a pivot

    "attributes": {},               # type-specific enrichment results
    "applied_enrichments": [],
    "enrichment_status": "not_started",

    "confidence": None,             # extraction confidence
    "validation_status": "observed",   # observed / inferred / validated /
unverified
    "provenance": [],               # extraction and enrichment provenance
    "tags": [],
    "priority": 0
}
```

5.1.1 IP address entity schema

Can be used for both public and private IP addresses, they can be distinguished by the attribute `ip_scope`.

```
ip_entity = {
    "id": "ip:192.168.1.10",
    "type": "ip",
    "value": "192.168.1.10",
    "normalized_value": "192.168.1.10",

    "ip_version": 4,
    "ip_scope": "private",          # private / public
    "is_internal": True,
    "port": None,                   # optional if observed in connection
context
    "protocol": None,               # tcp / udp / icmp if known
    "direction": None,              # source / destination / both

    "attributes": {
        "hostname": None,
        "reverse_dns": [],
```

```
        "fqdn_candidates": [],
        "mac_address": None,
        "dhcp_lease_history": [],
        "subnet": None,
        "gateway": None,
        "vlan": None,

        "asn": None,
        "asn_org": None,
        "whois_org": None,
        "whois_country": None,
        "cidr": None,
        "geolocation": {},
        "hosting_type": None,
        "reputation": None,
        "blocklist_status": [],
        "tor_flag": None,
        "vpn_proxy_flag": None
    }
}
```

5.1.2 Domain / FQDN entity schema

```
domain_entity = {
    "id": "domain:example.com",
    "type": "domain",
    "value": "example.com",
    "normalized_value": "example.com",

    "is_fqdn": True,
    "registered_domain": "example.com",
    "subdomain": None,             # e.g. login if value is
login.example.com
    "tld": "com",

    "attributes": {
        "whois_registrant": {},
        "registrar": None,
        "registration_date": None,
        "expiration_date": None,
        "domain_age_days": None,
        "whois_privacy": None,

        "name_servers": [],
        "a_records": [],
        "aaaa_records": [],
        "mx_records": [],
        "txt_records": [],
        "cname_chain": [],
        "passive_dns_history": [],
        "subdomains": [],

        "ssl_certificates": [],
        "ct_log_entries": [],
        "web_stack": [],
        "category": None,
```

```
        "reputation": None,
        "phishing_association": None,
        "malware_association": [],
        "dga_score": None,
        "typosquat_similarity": {},
        "parking_status": None
    }
}
```

5.1.3 URL entity schema

```
url_entity = {
    "id": "url:https://example.com/login?user=test",
    "type": "url",
    "value": "https://example.com/login?user=test",
    "normalized_value": "https://example.com/login?user=test",

    "scheme": "https",
    "domain": "example.com",
    "port": 443,
    "path": "/login",
    "query": "user=test",
    "fragment": None,

    "attributes": {
        "domain_entity_id": "domain:example.com",
        "reputation": None,
        "scan_results": [],
        "http_status": None,
        "redirect_chain": [],
        "final_url": None,
        "content_type": None,
        "page_title": None,
        "page_metadata": {},
        "embedded_objects": [],
        "downloaded_files": [],
        "shortener_expanded_url": None,
        "archive_snapshots": [],
        "screenshot_refs": []
    }
}
```

5.1.4 File hash entity schema

```
hash_entity = {
    "id": "file_hash:sha256:abcd1234...",
    "type": "file_hash",
    "value": "abcd1234...",
    "normalized_value": "abcd1234...",

    "hash_algorithm": "sha256",
    "file_name": None,              # if known at time of extraction
    "file_path": None,              # if known
    "host_observed": None,
```

```
    "attributes": {
        "file_names_observed": [],
        "file_type": None,
        "magic_type": None,
        "file_size": None,
        "compile_timestamp": None,
        "packer": None,
        "digital_signature": {},
        "av_detections": [],
        "detection_ratio": None,
        "malware_family": None,
        "yara_matches": [],
        "embedded_strings": [],
        "imports": [],
        "sandbox_summary": {},
        "network_iocs": [],
        "attack_mapping": [],
        "threat_actor_association": [],
        "first_seen": None,
        "last_seen": None,
        "prevalence": None,
        "ssdeep": None,
        "tlsh": None,
        "parent_hashes": [],
        "child_hashes": []
    }
}
```

5.1.5 User account entity schema

Similar to IP addresses, we can use the "scope" attribute to distinguish between local and domain accounts.

```
user_account_entity = {
    "id": "user_account:CORP\\jsmith",
    "type": "user_account",
    "value": "CORP\\jsmith",
    "normalized_value": "corp\\jsmith",

    "account_name": "jsmith",
    "domain": "CORP",
    "sid": None,
    "account_scope": "domain",      # domain / local / cloud / unknown

    "attributes": {
        "display_name": None,
        "account_type": None,       # user / admin / service / shared
        "group_memberships": [],
        "privilege_level": None,
        "account_status": None,     # enabled / disabled / locked
        "last_logon": None,
        "last_password_change": None,
        "password_policy_compliance": None,
        "mfa_status": {},
```

```
        "failed_login_count": None,
        "login_anomalies": [],
        "associated_devices": [],
        "department": None,
        "manager": None,
        "title": None,
        "employment_status": None,
        "vpn_history": [],
        "spns": []
    }
}
```

5.1.6 Process name entity

```
process_name_entity = {
    "id": "process_name:powershell.exe",
    "type": "process_name",
    "value": "powershell.exe",
    "normalized_value": "powershell.exe",

    "attributes": {
        "known_legitimate_application": True,
        "expected_paths": [
            r"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
        ],
        "expected_parents": [
            "explorer.exe",
            "cmd.exe",
            "services.exe"
        ],
        "lolbin_classification": True,
        "common_benign_usage": [],
        "common_malicious_usage": [],
        "prevalence_in_environment": None
    }
}
```

5.1.7 Process execution entity

```
process_execution_entity = {
    "id": "process_exec:HOST1:1234:2025-04-23T10:15:11Z",
    "type": "process_execution",
    "value": "powershell.exe -enc ...",
    "normalized_value": "powershell.exe -enc ...",

    "process_name": "powershell.exe",
    "pid": 1234,
    "ppid": 456,
    "command_line": "powershell.exe -enc ...",
    "image_path":
r"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe",
# using Python literal strings not to treat "\" as an escape character
    "attributes": {
        "parent_process_name": "cmd.exe",
```

```
        "expected_path_match": True,
        "expected_parent_match": False,
        "lolbin_classification": True,
        "digital_signature": {},
        "binary_hash": None,
        "network_connections": [],
        "user_context": None,
        "integrity_level": None,
        "session_id": None,
        "prevalence_in_environment": None
    }
}
```

5.2 Enrichment level 1

The first level of enrichment is the application of flows to the primary data entities. This can be querying a cloud/web service, an internal database, or cross-searching other parsed artifacts.
A typical example would be to search for a public IP address, obtained from the parsed Security.evtx log, in the AbuseIPDB service, or to search for a domain name and find information about an SSL/TLS certificate through Censys.io.

5.3 Enrichment levels 2 to N

The second to the nth level of enrichment is the iteration of the enrichment process on the data obtained by initial enrichment. For example, if we enrich the name of a process that has been identified in a RAM image, we can correlate its name with the hash of the executable file obtained from the disk image. We had to calculate the hash beforehand, and we were probably already enriching it. The name of the process can also be identified in the review of network connections, as communicating with a remote private IP address. If we enrich this IP address, we also enrich the process that triggered the connection.

N as the number of iterations should have a top limit set to avoid endless running of the model. At the same time, it is necessary that each newly identified information (entity) obtained through the enrichment process is verified for duplication. It is possible that we will identify a DNS record and an IP address at the input, which would refer to themselves in the enrichment process.

5.4 Evidence processing and enrichment process diagram

Figure 1 details the initial evidence processing phases up to data extraction, while Figure 2 outlines the subsequent data enrichment procedures.

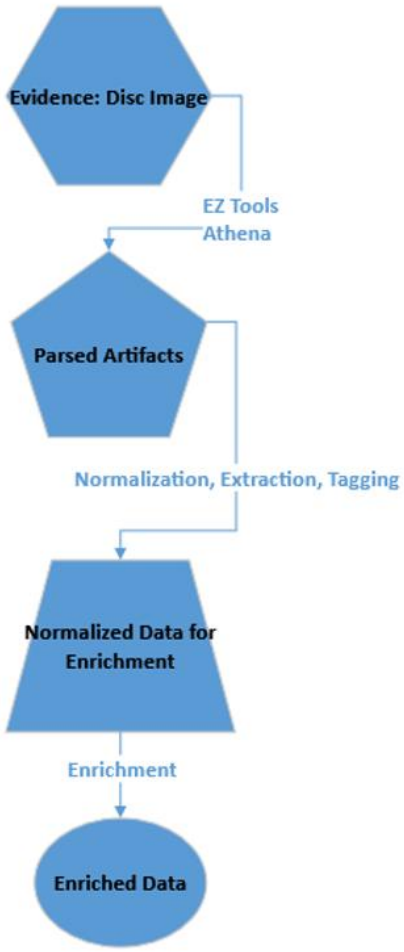


Figure 1: Evidence processing scheme up to enrichment of extracted data

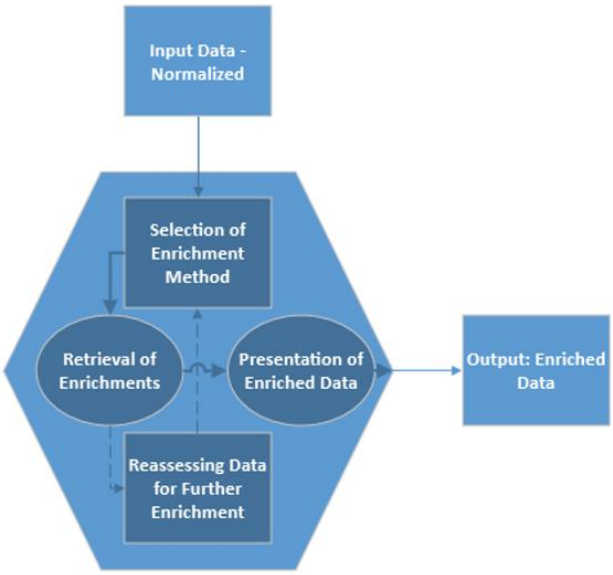


Figure 2: Diagram of the data enrichment process

5.5 Pseudocode

The algorithm processes normalized forensic records as an iterative "work list." It counts on the extraction of enrichable entities, but we do not address this in more detail within the model. The algorithm applies entity-specific enrichment modules, stores the returned context with a reference to its origin, and relists newly discovered entities for subsequent enrichment rounds until there are no new entities left or a predefined iteration limit is reached.

```
def enrich_dataset(records, registry, max_rounds=10):
        """
        Iteratively enrich normalized forensic records.

        Parameters
        ----------
        records : list
        Pre-processed and normalized forensic records obtained from parsed
        artifacts.
        These records may come from tools that parse EVTX, Registry, Prefetch,
        browser history, Amcache, LNK files, etc.

        registry : dict
        Mapping of entity type -> list of enrichment functions.
        Example:
        {
        "ip": [enrich_ip_whois, enrich_ip_reputation],
        "domain": [enrich_domain_dns, enrich_domain_whois],
        "hash": [enrich_hash_vt, enrich_hash_signature]
```

```
    }

    Each enrichment function is expected to accept one entity and return a
    result object such as:
    {
    "success": True,
    "data": {...},
    "source": "VirusTotal",
    "confidence": 0.95
    }

    max_rounds : int
    Safety limit to prevent infinite enrichment loops. This matters because
    one enrichment can reveal new entities, which can trigger more enrichment
    in later rounds.

    Returns
    -------
    output : dict
    Dictionary keyed by entity ID, containing:
    - the original entity
    - accumulated enriched attributes
    - provenance for each enrichment step
    - child entities discovered from enrichment
    """

# Initial worklist of entities extracted from the normalized records.
# This function is not described in depth.
# Enrichment model assumes that input is already in necessary format.
# Examples:
# - IP addresses from firewall logs
# - domains from browser history
# - hashes from Amcache, EDR, or calculated
# - usernames from event logs and ProfileList
worklist = extract_entities(records)

# Tracks which entity IDs have already gone through enrichment.
# This prevents processing the same entity repeatedly.
processed = set()

# Final enriched output.
# One entry per entity, keyed by a stable identifier such as:
# "ip:8.8.8.8"
# "domain:example.com"
# "hash:<sha256>"
output = {}

# Run enrichment in rounds.
# Each round processes the current worklist and may discover new entities
# that will be handled in the next round.
for _ in range(max_rounds):
# Stop early if there is nothing left to enrich.
if not worklist:
break
```

```
# Collect entities discovered during this round.
# These become the next round's worklist.
new_worklist = []

# Process each entity currently waiting for enrichment.
for entity in worklist:
# Build or retrieve a stable unique ID for the entity.
# This ID is used for deduplication and for storing results.
key = get_entity_id(entity)

# Skip entities that have already been fully processed.
# This avoids loops such as:
# domain -> IP -> domain -> IP
if key in processed:
continue

# Mark this entity as processed before running enrichments.
processed.add(key)

# Ensure an output record exists for this entity.
# "attributes" will accumulate enrichment data across modules.
# "provenance" records where each enrichment came from.
# "children" tracks newly discovered related entities.
output.setdefault(key, {
"entity": entity,
"attributes": {},
"provenance": [],
"children": []
})

# Select enrichment functions based on entity type.
# Example:
# entity["type"] == "domain"
# -> run DNS, WHOIS, reputation, passive DNS enrichments etc.
for enrich in registry.get(entity["type"], []):
# Run the enrichment function for this entity.
result = enrich(entity)

# Ignore unsuccessful enrichments.
# We suggest that in PoC model, failures are logged separately.
if not result["success"]:
continue

# Merge newly returned attributes into the entity's attribute set.
# Example:
# domain enrichment may add:
# {"registrar": "...", "age_days": 3, "mx_records": [...]}
#
# Note:
# Later enrichments may add more keys or overwrite earlier values,
# depending on how update() is handled in the final implementation.
output[key]["attributes"].update(result["data"])

# Record provenance for auditability and forensic defensibility.
# This is important because enriched findings should be traceable
```

```
# back to the source and method used.
output[key]["provenance"].append({
"enrichment": enrich.__name__,
"source": result.get("source"),
"confidence": result.get("confidence")
})

# Some enrichment results reveal new entities that can themselves
# be enriched in a later round.
#
# Examples:
# - domain enrichment reveals resolved IPs
# - hash enrichment reveals contacted URLs
# - email enrichment reveals sender IP or related domain
# - sandbox analysis reveals dropped file hashes
for child in extract_new_entities(result["data"]):
child_key = get_entity_id(child)

# Preserve the fact that this parent entity led to discovery
# of another potentially relevant entity – which should be enriched separately.
output[key]["children"].append(child_key)

# Queue the new entity for later enrichment, but only if it
# has not already been processed.
if child_key not in processed:
new_worklist.append(child)

# Prepare the next round.
# This step should remove duplicates and may also prioritize entities.
# For example, public IPs, domains, hashes, or rare indicators could be
# processed before lower-value entities.
worklist = deduplicate_and_prioritize(new_worklist)

# Return the accumulated enriched dataset.
return output
```

6 Conclusion

The digital evidence enrichment model presented in this paper shows that systematically adding context to digital evidence significantly increases their value for forensic analysis. The aim was to create a comprehensive framework that would allow artifacts to be effectively linked to external sources, indicators of compromise (IoC) and defensive intelligence (Threat Intelligence).

The result is a flexible and scalable model that supports different levels of enrichment – from simple attributes to complex composite information. This approach enables more accurate evaluation and resolution of incidents, faster attribution and better prioritization of activities during IR and forensic analysis. The model represents a step towards data-driven digital forensics and forms a solid foundation for further research and practical deployment.