NERD: Network Entity Reputation Database

Václav Bartoš
CESNET a.l.e.
Prague, Czech Republic
bartos@cesnet.cz

ABSTRACT

We present an open database of known malicious entities on the internet called *Network Entity Reputation Database*. It gathers alerts from a large number of diverse security monitoring tools and other sources and keeps detailed information about all network entities (IP addresses, ASNs, domain names, etc.) which have been reported as malicious. It also adds other related data from a multitude of sources, like whois registries, blacklists or geolocation databases. Due to the large amount, diversity and volatility of such data, creation of such a database system is not trivial. In the paper we describe the data model, system architecture and technologies used, as well as some statistics from the pilot deployment of the system. We operate the database as a free service for the cyber security community to help with prevention, defense, investigation of incidents as well as research and believe it will become a valuable contribution to the family of existing open cyber threat intelligence platforms.

CCS CONCEPTS

• **Security and privacy** → **Network security**; Intrusion detection systems; • **Information systems** → Information integration; Information storage systems.

KEYWORDS

Network security, Reputation database, Threat intelligence platform, CTI, OSINT

ACM Reference Format:
Václav Bartoš. 2019. NERD: Network Entity Reputation Database. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES 2019) (ARES '19), August 26–29, 2019, Canterbury, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3339252.3340512

1 INTRODUCTION

The amount and variety of cybersecurity threats are growing, especially on the network. Security professionals recognize this trend and develop and deploy various network monitoring and analysis tools that automatically detect and report diverse kinds of security incidents. Examples of such tools include honeypots, flow analysis systems, anomaly detectors or log analyzers.

Such tools usually generate some kind of data records about detected events. We call them *alerts*[1]. An alert is a low-level information describing a network security event, for example: "IP address A tried to connect to an SSH honeypot", "IP addresses A,B,C took part in a DDoS attack against IP address X", "IP address B scanned whole IP address range R on port 23" or "URL Y was identified as a command and control (C&C) server of a botnet". Such alerts represent a low-level cyber threat intelligence (CTI). They may be useful by themselves, but it is also common to gather alerts from multiple sources, or share them, and apply various correlation mechanisms to get a more high-level information about the current threats.

There exist other types of CTI. For example, results of malware analysis, or information on ongoing malware or phishing campaigns. Many types of such information are being shared among security professionals and organizations using various CTI sharing platforms. These include, for example, Warden[2] [5] for sharing low-level alerts, DShield[3] for IDS and firewall logs, or more generic CTI tools and platforms like MISP[4] [8], AlienVault Open Threat Exchange[5] or IBM X-Force Exchange[6]. Some of these platforms serve only for data sharing, others also perform some analysis or correlation and create a searchable database. Data in such systems are usually grouped primarily by events, campaigns or threat actors. We agree this is useful in many cases, but we propose another, complementary approach – to focus primarily on the network entities that may pose a threat – malicious IP addresses, networks, or domain names. For example, if an analyst encounters a suspicious IP address, he may need to quickly assess if it is known for any malicious activity and what kind of threat it may pose. A database specialized on assessing such entities may also allow to label them or rank them by various criteria, which in turn may be used to generate lists of the most dangerous entities or those that target a specific protocol or vulnerability, for example.

Of course, the idea of listing malicious entities is not novel. For a long time it is common to list IP addresses or domain names deemed as malicious in so called blacklists. They are well known for their usage to fight spam emails, but they are used in other areas as well. Blacklists are very easy to use, but they typically do not provide any further information about the entities, only whether it is listed or not.

There also exist reputation databases provided as part of various commercial cyber security solutions. These sometimes contain more detailed information about the malicious entities, but access to them

ARES '19, August 26–29, 2019, Canterbury, United Kingdom

ACM ISBN 978-1-4503-7164-3/19/08…$15.00
https://doi.org/10.1145/3339252.3340512

[1] There exist other names for this kind of information with the same or similar meaning, *e. g. events*, *incident reports* or *indicators of compromise*.
[2] https://warden.cesnet.cz/
[3] https://www.dshield.org/howto.html
[4] http://www.misp-project.org/
[5] https://otx.alienvault.com/
[6] https://exchange.xforce.ibmcloud.com/

is limited only to users of the particular security solution and due to their closed nature, it is usually unclear what are their data sources, how the data are processed or what rules are used to list or delist an entity.

In this work, we present a rich, open database of information about malicious entities called *NERD*, *Network Entity Reputation Database*. It gathers alerts from a large number of diverse sources and keep detailed information about all network entities reported as malicious (not only IP addresses, but also network prefixes, domain names, etc.). It not only lists the entities but also contains information on reasons why the entity was listed, when and based on what sources of information. It also provides additional information (like geolocation or network type) that might be useful for an analyst. All this information is provided for free to the cyber security community to help protect networks, investigate incidents or conduct research.

Due to the large amount, diversity and volatility of data stored in such a database system, its design and implementation is not trivial. Besides presenting the system as a service for the community, the goal of the paper is also to describe the data model, system architecture, data processing methods and technologies used to implement it.

The paper is organized as follows. In the next section the NERD system is described in detail. Section 4 summarizes the current state of implementation and Section 3 briefly describes access policy for the data. In Section 5 we show some basic statistics of data currently stored. Section 6 concludes the paper.

2 SYSTEM DESCRIPTION

This section describes the design of the system with all planned features. Not all of them are currently available (see Sec. 4 for details).

From a user's point of view, the Network Entity Reputation Database (NERD) system is an online portal where the user can search any IP address, domain name or another network identifier (an *entity*) and get all security related information known about it – list of all alerts that reported it as a source of some malicious activity, whether it is listed on some blacklists or other databases, related information from DNS, whois, geolocation, or data from internet-wide scanning services. It is also possible to search for entities that match various criteria and sort them by various attributes or by a score summarizing the associated threat level. There is also a REST API for easy integration of the data into any other security or threat intelligence system.

Behind the web portal and the API, which servers to access the data, there is a complex, modular system to acquire and process data from various sources, store them to the database and periodically update them.

2.1 Data sources and storage

The data model used by the reputation database must be highly flexible. That is because it is expected that new data sources will be added from time to time, while the old ones may be removed, either because the data source becomes discontinued or it turns out to be unreliable, for example. Also, the cyber security area as a whole changes over time so the requirements on the system and the data provided by it may change as well. Nevertheless, it is still possible to describe some general concepts that are not expected to change.

The main data unit is an entity record, a JSON-like document that stores all information about an entity. The entity may be an IP address, network, autonomous system number (ASN), domain name, etc. Each record has a number of attributes and may contain links to other entities.

Sources of the data stored in the records can be divided into two classes – *primary* and *secondary*. The primary data sources are those that report some particular network entities as malicious and are used to create entity records. It may be alerts from honeypots, IDS or anomaly detection systems (gathered directly or via an alert sharing system, such as Warden), data from cyber threat intelligence sharing platforms, such as MISP, or even simple blacklists if it is possible to download them[7].

When an entity record is created, after the entity was reported by some primary source, secondary data sources are used to fetch additional information about it. This include querying whois databases, DNS for mapping IP addresses to hostnames and vice versa, geolocation and querying IP or domain lists that are not used as primary sources (either because it is not possible to download the list, or because the listed entities are not necessarily malicious, as in case of dial-up or proxy lists, for example). It is also possible to query other databases, such as Shodan[8] or VirusTotal[9], and store summary of results locally.

The primary data are usually received as some kind of alerts or as records fetched from another system. Only meta-data, such as number of alerts per day and type, are stored in the entity records. However, a copy of the original record is often stored as well (in a separate database), so the system is able to provide full details. The secondary data are usually stored directly as attributes of the records.

Some attributes can also be computed from other attributes already present in the records. For example, the meta-data about received alerts can be used to assign tags based on the prevailing type of malicious activities performed by the entity, or some rules and regular expressions may be applied to hostnames to guess properties of the device or its connection, e.g. if it is a dynamically assigned IP address, NAT, cloud server or a DSL connection.

The system also allows to store data with some kind of uncertainty or those whose reliability decreases over time. Therefore, some of the attributes contain not only the value, but also the timestamp of when the value was acquired or checked, and/or a confidence value.

For some attributes whose values can change often and the history may be important, like presence of an IP address on blacklists, the database keeps not only the current value, but also all the history values and their timestamps over several last months.

2.2 Entity scoring

All the data about entities are available to users in full detail, but in order to provide simple and actionable information, it is beneficial to also provide some kind of summarization, e.g. by assigning a

[7] Some blacklist providers only allow to query a particular, already known IP address or domain name, which prevents usage of such a list as a primary data source.
[8] https://www.shodan.io/
[9] https://www.virustotal.com/

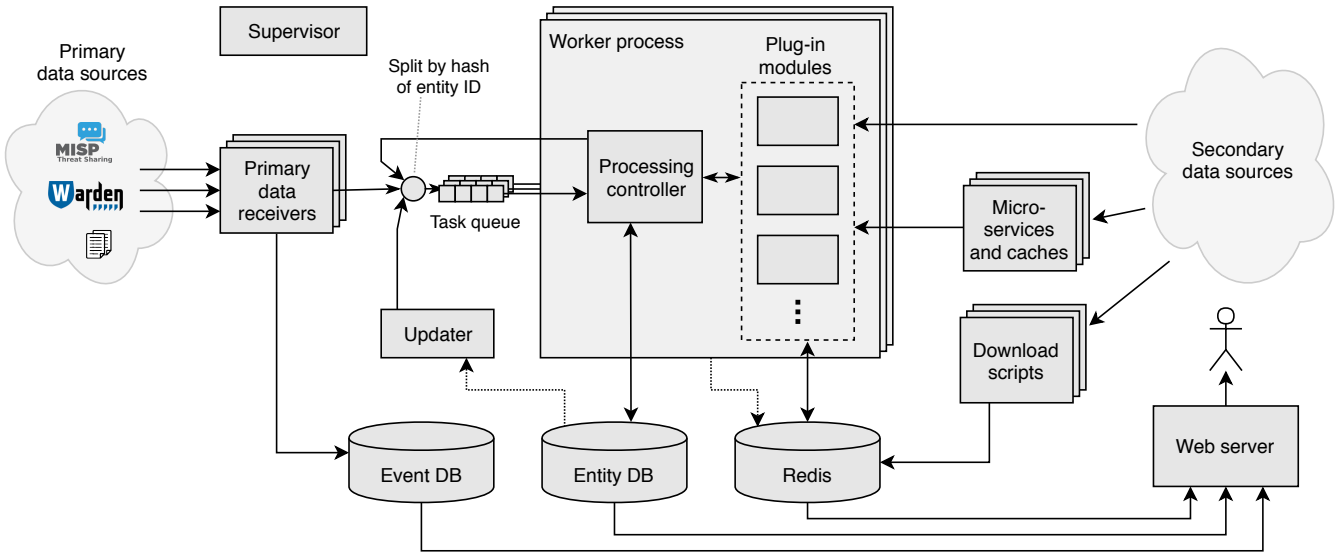

Figure 1: Architecture of the NERD system

number or a small set of numbers (scores) to each entity. Such a numeric summarization allows for quick comprehension of entity characteristics as well as ranking of entities, e.g. by the level of threat they pose. It can then be utilized by automated security tools, for example, to block traffic from the most offending IP addresses or domains, or by a user, for example, to prioritize investigation of reported incidents or to bring attention to a prevailing issue.

The NERD system implements a scoring mechanism based on the method presented in [2]. It assigns a score to each address based on the estimated probability of it performing malicious activity in the immediate future. The score is called *Future Misbehavior Probability (FMP)*. Several such scores can be computed for each address, each expressing the likelihood of different type of attack and for different length of the future time window (e.g. next 4 hours, 24 hours, a week). Thus the score expresses the level of threat each IP address poses in a specific context. A summary score, computed as a weighted sum of the individual ones, is available as well.

The focus on the future behavior is motivated by the fact, that the information about attacks and attackers that are likely to occur in the near future is exactly what is needed for effective prevention. However, since the prediction is based mostly on reports of previous malicious behavior of each address, the scores also serve well as a summarization of address' characteristics and previous activities.

The prediction is performed by a machine learning model based on gradient boosted decision trees (also known as xgBoost [3]) trained on historical data. More information about the method as well as examination of some potential uses of the score can be found in [2].

2.3 Architecture

Due to the need for flexibility, as well as scalability, the NERD system uses a highly modular architecture. It consists of a number of collaborating components. The architecture is depicted in Figure 1.

First, there are several databases. The main one, called *entity database*, stores all the entity records, as described in Section 2.1. There is also a separate database to store original data from primary sources (events, alerts). These may be actually multiple databases, since different technologies may be suitable for different types of data. At last, there is an instance of a fast in-memory key-value database (Redis), that serves for storage of various, often temporary or fast changing, data that need to be accessed globally by different components. It also serves for various logging and caching purposes.

The main input of the system is represented by a set of primary data receivers. They receive messages, alerts, or entity lists from the primary data sources and put *tasks* (called *update requests*) to a global queue to create or update records of related entities.

The tasks are processed by the core of the system – a set of *workers*. They apply the requested updates on given entity records. Also, the workers fetch data from external (secondary) sources and compute other attributes. This functionality is handled by plug-in modules, so it is easy to add, change or remove secondary data sources or computed attributes.

The workers may run in any number of instances in parallel, which makes the system easily scalable. Tasks are distributed to workers according to a hash of the entity identifier (a task always involves processing of a single entity record only), so the same record is always handled by the same worker. This helps to avoid a need for record locking and other concurrency issues. If the processing core or a plug-in needs to store a global state information, it is stored outside the workers, usually in the Redis database.

Most of the plug-ins fetch data from external sources. Depending on the type and availability of the data, three methods of data acquisition are used: (i) the data are queried directly from their original sources by the plug-in module (for example, whois data are got this way); (ii) there is a special microservice or a cache to provide easier or more efficient access to the data (an example

is a caching recursive DNS resolver used by hostname resolving modules as well as for querying DNS blacklists; passive DNS module gets data via a microservice that queries several external passive DNS systems and merges the results); (iii) the data are periodically downloaded, preprocessed and cached locally, as entries in Redis or as local files (for example, downloadable blacklists or a geolocation database).

Periodic refreshing of data in the entity records is controlled by a component called updater. It checks timestamps of the last update stored in each record and emits a task to update a set of attributes when they are older then a specified time.

The last component is the web server, that presents all the data in databases to users, either via web interface or REST API.

Whole system is managed by the process control system Supervisor[10].

2.4 Data processing

The data processing in workers is driven by tasks called *update requests*, which are fetched from the main queue. An update request is a simple message specifying a set of operations that should be done with attributes of a specific entity record (e.g. increment the total number of alerts by 1 for IP address X).

Upon receiving an update request, the worker fetches the corresponding entity record from the database (or create a new one, if it does not exist yet) and applies the requested changes. But that is just the beginning. Each plug-in module can register callback functions that are called when a specific event occurs. This may be a special named event, such as addition of a new record, or a change of value of a specific attribute. For example, a DNS module may register a function on creation of a new IP address record. This function resolves the hostname assigned to the IP address and stores it to the record. This change of the hostname attribute may trigger another function, registered by another module. It may, for example, assign some tags based on the keywords found in the hostname. This way, a single update request may result in a complex cascade of subsequent updates. Only when the whole cascade of changes and callback functions is processed, the modified record is written back into the entity database. The processing then continues by fetching another update request from the main queue.

Each update request, including all subsequent updates in the cascade, work with a single entity record only. If a callback function wants to make changes in some other record (e.g. update the record of an autonomous system when a new IP address is added), it enqueues a new update request into the main queue.

2.5 Implementation technologies

Most of the data gathering and processing components were implemented from scratch as programs in Python, but of course, the system also uses a number of well-known third party software.

Because of the diversity of the stored data and the aforementioned need for flexibility, we decided to use MongoDB as the main storage of entity records. It is a schema-free document based database system, so it allows to easily add new attributes without the need to reconfigure database schema, it naturally supports arrays and subobjects and allows to create indices on any fields to enable fast queries.

The database for copies of raw primary data (currently only alerts form Warden) do not have such requirements for flexibility or special data types, so we use a simple table in a classic SQL database – PostgreSQL. Also, as already mentioned, there is a Redis database for communication between components and various caching and logging purposes.

Besides the databases, all the communications between components – most notably the main task queue, but also some notifications or special queries from web interface – are implemented by RabbitMQ queues.

All these data processing components are being managed using the Supervisor tool.

The web portal and API are implemented in Python Flask framework. The frontend part uses jQuery and several JavaScript microlibraries.

The whole system is currently running on a single server. However, the architecture allows to easily distribute it on multiple servers when such a need arises. The only change needed would be to replace the Supervisor by an orchestration system able to control such a distributed system.

3 ACCESS POLICY

The NERD system gets data from many different data sources, some of them are public, others have a limited access. Also, some pieces of the data may have character of private information (e.g. data about connections made by concrete IP addresses) or may be otherwise sensitive (e.g. IP addresses of honeypots). Therefore, full access to all the data is limited to trusted members of cyber security community and restrictions apply to allowed use of the data.

However, a subset of data is available to general public. Those are data that do not require any special protection and come from public sources and also an aggregated information from some restricted sources. This limited data set is still quite large and can be utilized in many ways, so the system can be used immediately by anyone without a need to request the full access.

4 CURRENT STATE

A pilot deployment of the system runs as a free service available at https://nerd.cesnet.cz/. The source code is released as open-source and available on GitHub[11].

This section briefly describes the current state of this implementation at the time of writing (April 2019).

4.1 Supported features and data sources

The core of the NERD system, as well as many data processing modules and the web interface, are fully implemented, only the number of data sources and supported types of entities are limited.

The primary entity type supported are IPv4 addresses. There are also records of BGP prefixes and ASNs as groupings of IP addresses according to the routing information. Similarly, there are records of IP blocks and organizations according to data from whois databases of regional registries. All currently supported entities and their relationships are depicted in Figure 2.

[10] http://supervisord.org/

[11] https://github.com/CESNET/NERD/

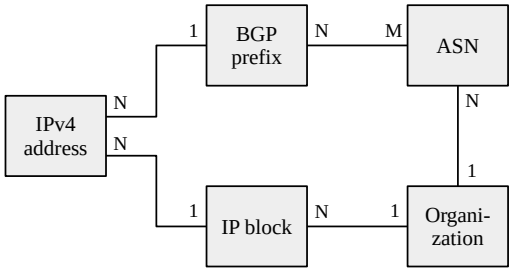

Figure 2: Currently supported entities and their relationships

NERD currently uses only one primary data source – Warden, an alert sharing system operated by CESNET [1, 5]. This system gathers alerts from more than 30 security monitoring systems of various types (honeypots, NetFlow analysis tools, IDS, etc.) deployed in several large networks and provides them in a unified format, IDEA[12] [6]. The average number of alerts received from Warden is around 2 million per day (23 per second). The meta-data stored about alerts are the number of alerts per day, attack category and the reporting detector.

We plan to add several more primary data sources very soon, namely MISP [8] and selected feeds from AlienVault Open Threat Exchange. We also plan to use several blacklists as the primary source. Currently, blacklists are used as secondary sources only, which means they are only queried for IP addresses that already has a record in NERD, i.e. they were reported to Warden. When used as primary source, each IP address on the blacklist will get a record in NERD, even if it was not reported by any other source.

The list of secondary data sources is longer. It includes resolving hostname via a reverse DNS query, geographic location (using GeoLite2 database from MaxMind[13]), or originating BGP prefixes and ASNs, together with related data from whois databases. As already mentioned, many public blacklists are queried (some are periodically downloaded and cached locally, others are queried via DNSBL services). A number of tags are assigned to each IP address based on numbers of alerts as well as the other information, like the hostname or presence on some specific blacklists. Last but not least, there is a module implementing the prediction model for assigning the FMP score to IP addresses, as described in Sec. 2.2.

Some information is also got for ASN records. Besides the name and the description from whois records, it is a BGP rank provided by CIRCL[14] or a network type (e.g. enterprise or transit) according to a dataset[15] provided by CAIDA.

Both web interface and API allow users to get details about any entity in the database as well as search IP address by various criteria, like IP prefix, hostname, geolocation, presence on blacklists or tags. The API also allows to download records of all addresses matching a given filter, or to get FMP score of a long list of addresses in a single query[16].

[12] Intrusion Detection Extensible Alert, https://idea.cesnet.cz/
[13] https://dev.maxmind.com/geoip/geoip2/geolite2/
[14] https://www.circl.lu/projects/bgpranking/
[15] The CAIDA UCSD AS Classification Dataset, https://www.caida.org/data/as-classification/
[16] See full API documentation at: https://github.com/CESNET/NERD/wiki/RESTful-API

Table 1: Average number of records in NERD

Entity type	Count
IPv4 address	1,200,000
BGP prefix	123,000
ASN	16,000
IP Block	38,000
Organization	16,000

4.2 Performance metrics

Data processing in the backend of the system is triggered by two types of events – an incoming alert from Warden or a periodic update of an existing entity record.

As mentioned above, on average there are around 23 alerts per second received from Warden. A new IP address is reported and thus a new entity record is created approximately once per second (old alerts are removed in a similar rate). The rest of the alerts are related to addresses observed earlier so they result in just a simple update of an existing record.

The periodic updates of secondary data are triggered once per day for each existing record. During such updates, values of all attributes are determined in a single batch and all updates are written to the database together, which means one database update per record per day.

There are usually between 1 and 1.5 million records in the database. Together with processing the Warden alerts, this means 3 to 3.5 million database updates per day (up to 40 per second).

On the frontend side, there is slightly over 100,000 queries per day (1.2 per second) on average, mostly via the API.

The whole system is currently running on a single virtual server with 8 vCPUs and 32 GB of RAM without any performance issues. The entity database takes only around 3.5 GB of disk space (thanks to compression performed automatically by MongoDB).

Our experience shows that the performance bottleneck is data processing in the Python application itself (the worker process). The current load can barely be processed by a single worker, but it is easy to scale up by adding more workers.

The other components (most notably the databases) are able to process several times more updates and reads even in the current single-instance deployment and all of them are ready to be distributed onto multiple machines if needed.

5 STATISTICS

Although the main goal of NERD is to provide detailed data about individual entities, in order to provide more insight into the amount and characteristics of the stored data, this section shows some basic statistics from the current deployment of the system.

Each IP address record is currently kept in the database for 14 days since the last related alert is received. Records of other entities are kept while another entity references them (e.g. IP block record is removed when there are no more IP addresses in the block). With this setting, there is usually around 1.2 million IP addresses in the database. Average numbers of records (during April 2019) for all entity types are shown in Table 1.

Most IP addresses are observed as malicious only for a short period of time, i.e. most records are created just because of one or

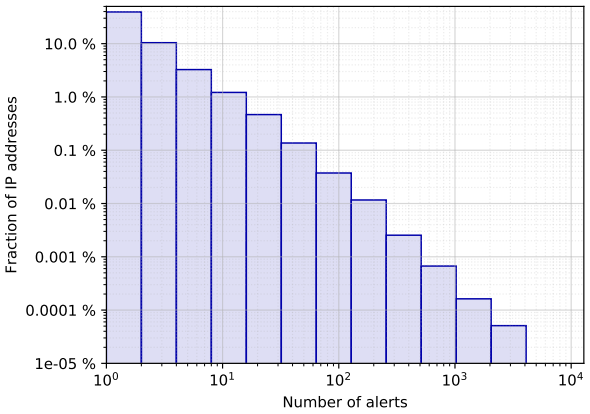

Figure 3: Histogram of number of alerts reported in last month per each IP address. Note the logarithmic scale on both axes, which also means the bins do not have the same size.

a few alerts and removed after 14 days of inactivity. This can be observed in Figure 3 which shows a histogram of the number of alerts reported in last month per each IP address in the database. Indeed, most addresses have just one or a few alerts stored. These are either dynamically assigned IP addresses or low-activity attackers. In general, they are usually not considered as a significant threat (although it depends on the type of the reported malicious activity as well). On the other side of the plot, there are addresses that are reported by very large numbers of alerts. Despite their relatively low number, these highly active addresses are responsible for large amount of received alerts. A deeper look into the data shows, for example, that only around 8000 (0.7 %) addresses are detected more than 1000 times in a month, but these are responsible for more than half (57 %) of all alerts received.

Another example of a simple statistical analysis that can be done with the data in NERD utilizes the geolocation information. It is well known that malicious IP addresses are geographically distributed highly nonuniformly [1, 7, 9]. This can be observed in our data as well. Table 2 shows top-10 countries by the number of IP addresses reported as sources of port scanning and unauthorized login attempts. Although the lists of countries as well as their relative shares are slightly different depending on the attack type, in both cases these ten countries alone contain approximately 58 % of all addresses in our database.

At last, we show a statistic of presence of IP addresses on blacklists. For each IP address in the database, NERD checks 50 public blacklists. Table 3 shows the number of IP addresses by the number of blacklists listing them. Almost 74 % addresses are found on at least one of them, many are listed on multiple ones. On the other hand, this means that over 26 % of IP addresses reported as malicious via Warden has not been recognized by any major blacklist provider, yet, and NERD thus provides a completely new information there. No IP address is simultaneously listed by more than 10 blacklists. This is not suprising, since many of the queried blacklists

Table 2: Top-5 countries by the number of IP addresses reported as a source of malicious traffic.

Port scans	
Country	IPs
China	8.9 %
Vietnam	8.6 %
Russia	8.4 %
Brazil	6.4 %
USA	5.9 %
India	5.8 %
Indonesia	5.8 %
Iran	3.9 %
Taiwan	2.9 %
Ukraine	2.2 %

Login attempts	
Country	IPs
China	15.2 %
USA	9.7 %
Brazil	7.2 %
Russia	5.6 %
India	4.2 %
Iran	3.6 %
S. Korea	3.6 %
Taiwan	3.1 %
Indonesia	3.0 %
France	2.7 %

Table 3: Numbers of IP addresses by the number of blacklists on which they are listed

Blacklists	IPs	
0	308012	26.5 %
1	346903	30.0 %
2	393634	33.8 %
3	88795	7.6 %
4	18484	1.6 %
5	5350	0.46 %
6	1875	0.16 %
7	692	0.060 %
8	232	0.020 %
9	71	0.006 %
10	16	0.001 %

are specialized for particular types of attacks and contain just a small number of entries.

All the numbers in this section come from a single day in April 2019. However, the statistics are quite stable and data from other days look very similar.

6 CONCLUSIONS

We presented a system and a service that provides an open database of known malicious IP addresses and other entities – Network Entity Reputation Database (NERD). It is available for free to a broad cyber security community to help with prevention, defense, investigation of incidents as well as research.

We believe that thanks to its ability to process large amount of data, open access and the API for easy integration with other systems, it will become a valuable contribution to the family of existing open cyber threat intelligence platforms. There already are active users of NERD in several organizations and the FMP score provided via its API is used by the platform for threat intelligence sharing and analysis developed in project PROTECTIVE[17]. We also work on implementing the idea of using the FMP score from NERD as an input to a DDoS mitigation algorithm, where it should help to distinguish good and bad traffic as described in [4].

[17] https://protective-h2020.eu/

ACKNOWLEDGMENTS

This research was supported by the Security Research Programme of the Czech Republic 2015 - 2020 (BV III / 1 VS) granted by the Ministry of the Interior of the Czech Republic under No. VI20162019029 The Sharing and analysis of security events in the Czech Republic.

Thanks also goes to CAIDA organization for providing us access to the CAIDA UCSD AS Classification Dataset (http://www.caida.org/data/as-classification).

The system includes GeoLite2 data created by MaxMind, available from https://www.maxmind.com.

REFERENCES

[1] Václav Bartoš. 2016. *Analysis of alerts reported to Warden.* Technical Report 1/2016. CESNET.

[2] Vaclav Bartos, Martin Zadnik, Sheikh Mahbub Habib, and Emmanouil Vasilomanolakis. 2019. Network entity characterization and attack prediction. *Future Generation Computer Systems* 97 (2019), 674–686. https://doi.org/10.1016/j.future.2019.03.016

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.

[4] Tomáš Jánský, Tomáš Čejka, Martin Žádník, and Václav Bartoš. 2018. Augmented DDoS Mitigation with Reputation Scores. In *Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES 2018)*. ACM, New York, NY, USA, 54:1–54:7. https://doi.org/10.1145/3230833.3233279

[5] Pavel Kacha, Michal Kostenec, and Andrea Kropacova. 2015. Warden 3: Security Event Exchange Redesign. In *19th International Conference on Computers: Recent Advances in Computer Science.*

[6] Pavel Kácha. 2013. IDEA: Designing the Data Model for Security Event Exchange. In *17th International Conference on Computers: Recent Advances in Computer Science.*

[7] Kurt Thomas, Rony Amira, Adi Ben-Yoash, Ori Folger, Amir Hardon, Ari Berger, Elie Bursztein, and Michael Bailey. 2016. The Abuse Sharing Economy: Understanding the Limits of Threat Exchanges. In *Research in Attacks, Intrusions, and Defenses (RAID) (LNCS 9854)*. Springer, 143–164. https://doi.org/10.1007/978-3-319-45719-2_7

[8] Cynthia Wagner, Alexandre Dulaunoy, Gérard Wagener, and Andras Iklody. 2016. MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform. In *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*. ACM, 49–56.

[9] Jing Zhang, Ari Chivukula, Michael Bailey, Manish Karir, and Mingyan Liu. 2013. Characterization of Blacklists and Tainted Network Traffic. In *Passive and Active Measurement (LNCS 7799)*. Springer Berlin Heidelberg, 218–228. https://doi.org/10.1007/978-3-642-36516-4_22