CyberGreen Data Platform v2

Technical Architecture - Overview

L. Aaron Kaplan (CyberGreen, Cert.AT)
&
Dr. Rufus Pollock (Atomatic Ltd)

Version: 1.1

Table of Contents

Introduction

Cybergreen wants to improve cyber health through research, metrics and outreach.
Our modern economy is highly dependent on the Internet, which itself is dependent on information and network security. Threats to the Internet's security and stability can have effects on the global economy.

Only via repeatable measurements can we identify risks to global cyber health and address these. Measurement data must be made available to remediation teams, policy makers, CERTs and Cybergreen's users so that they can take collective action on it.

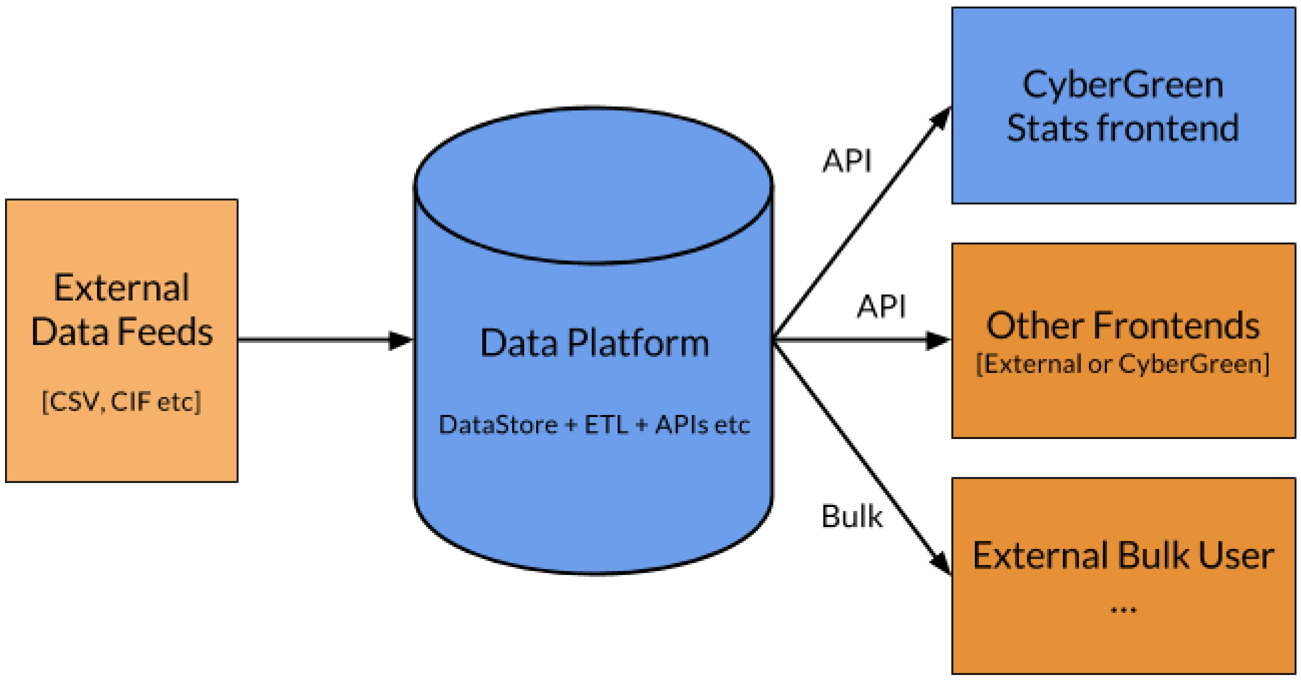
The platform will serve as a data hub for these kinds of aggregated internet health measurements, allowing researchers, policy makers and CERTs to inspect the data as well as giving them bulk access to the data stored in the platform[1].

The "platform" as described consists of two major parts:

- "Data platform": the system for importing, cleaning, storing and making data available in bulk and via API to users and frontend applications
- "Stats frontend": this is a website which presents the Cyber Health Index and other main statistics in a way that is easy to access and use, pulling data from the data platform.

The following diagram illustrates the relationship of these different components both to each other and to third party resources and users.

[1] We will distinguish between aggregated data and raw data at a later stage in this document.



Overview of this document:

1. First we will discuss Cybergreen platform version 1 and lessons learned from it.
2. Next in "Overview of the data platform", we will describe version 2 of cybergreen, as far as the data platform (the core) is concerned.
3. We list key user story epics which outline the main requirements placed on the platform by different types of users.
4. After we clarified users' expectations, we can discuss the actual architecture in the next chapter.
5. Appendix: listing existing tools, a glossary and coding standards.

Scope

This technical architecture document **focuses almost entirely on the "data platform"**. Thus, for the rest of this document when we say "platform" we mean the "data platform" as set out above.

This data platform is a combination of a datastore (or datastores), collection mechanisms, connectors to existing collection mechanisms (such as CIF, IntelMQ, etc), filtering, data enrichment and aggregation and access methods including bulk download and web APIs. That platform does not include the visual frontend(s), visualizations or other websites such as:

- stats.cybergreen.net
- www.cybergreen.net (the landing site, blog etc)
- Any embeddable visualisations

This choice of focus is intentional. The data platform is where “architecture” is actually needed – the frontend site itself will not require large-scale technical architecture. The data platform is also the core of system and presents the primary technical challenges in implementation. Finally, the frontend site will be driven off the data platform so it forms a natural first point of focus.

Finally, this document does not focus on the Cybergreen metrics (“Cybergreen Index”).

Lessons learned from CyberGreen v1

Details on CyberGreen v1 can be found in the Appendix. This section focuses on the lessons learned and what they imply for CyberGreen v2.

- The Cybergreen Index version 1 was confusing: people intuitively tried to rank their respective countries or ASNs against each other, while the index did not lend itself to this type of comparison. This led to some users dismissing the whole project.
    - *=> This is being addressed with the creation of a new index definition later this year (separate from this technical architecture - though important context)*
- Data collection was sketchy at best. Countries with little data were showing as doing well in the index.
    - *=> Good data collection and processing, as well as good choices about missing data are key to providing robust, actionable intelligence.*
- The API was well hidden and it was hard to get out data from version 1.
    - *=> Providing high quality bulk and API access is important and should be an up-front design goal. Having a solid API will allow for multiple frontends. Downloading bulk data must be easy.*
- Cybergreen had challenges to obtain raw bulk data from “super-remediators” (raw data collectors). This was both for privacy reasons and because of potential conflict of interest esp with commercial providers who were collecting measurements of internet health and re-distributing them themselves to a global audience.
    - *=> Allow super-remediators to provide pre-aggregated data. Substantial work to build trust and collaborations.*
    - *=> Be flexible for connecting to multiple collection platforms such as CIF, IntelMQ, Abusehelper, or any other externally generated aggregated CSV files*
- Users want to access data via ISP or ASN as well as at the country level. This is especially important for actionable intelligence for CERTs.
    - *=> Make sure we include ASN and other key “axes” of analysis in the data we collect and aggregate. Version 2 should provide per ASN/per Risk data as well as per country data.*

- Cybergreen version 1 was not telling any stories to policy makers
    - *=> Design the front-end with policy-makers in mind and make sure the data platform supports the kinds of stories you want to tell.*
- The platform version 1 was bulky, cumbersome and created a lock-in effect on a small company who were the only ones being able to operate, modify and adapt the platform. There was a lack of standardised tooling and shared best practices which hampered scalability and sustainability.
    - *=> Design a platform that uses more standardized components, that draws on lessons from the data platforms and open data communities*
    - *=> Be flexible for connecting to multiple collection platforms such as CIF, IntelMQ, Abusehelper, or any other externally generated aggregated CSV files*
    - *=> De-coupling! While version 1 was a monolithic system, version 2 shall be loosely de-coupled and allow for integrating with multiple inputs as well as outputs.*
- Cybergreen v1 was a monolithic architecture, all centralised in one system.
    - *Cybergreen v2 aims at a distributed, even federated architecture which allows multiple parties to benefit from exchanging (raw or aggregated) data* ***in order to cooperatively address cyber health issues.***

Raw versus Aggregated data

Cybergreen v2 shall only aim at serving raw OSINT[2] data where appropriate. All other data will be aggregated and only served in aggregated form to the public.
Aggregated data has the added benefit of being smaller as well as not touched by privacy issues.
Cybergreen will happily accept pre-aggregated data as long as the data source is trustworthy and provides accurate measurements. A format for pre-aggregated data can be found in the section “Interfacing externally aggregated CSVs”.

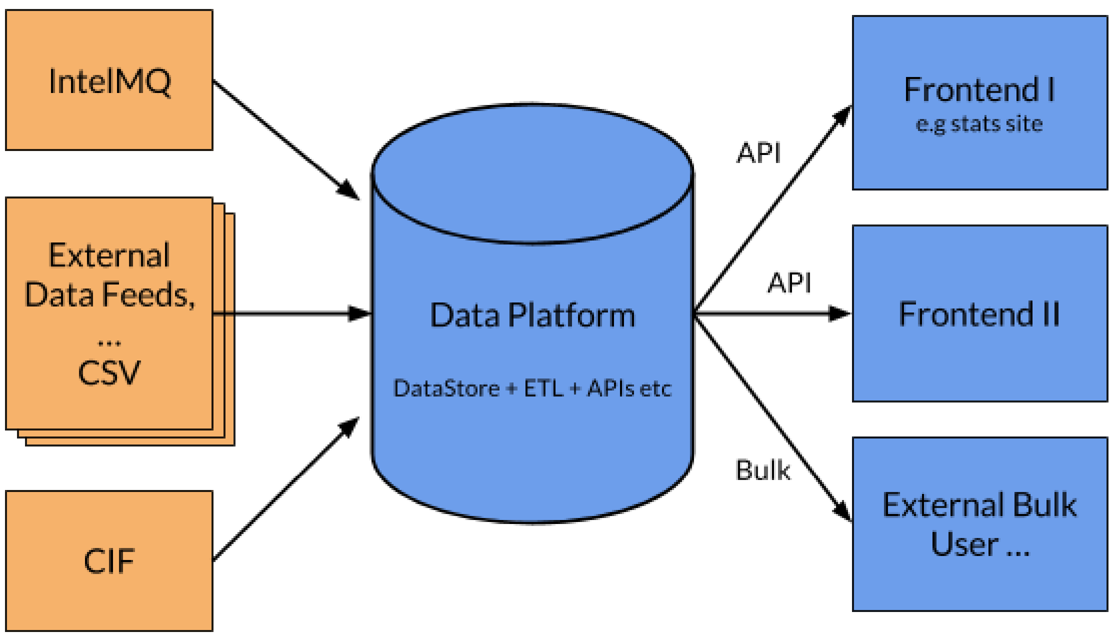
Cybergreen v2 will try to avoid conflicts with super remediators, who’s business it is to send out raw data.

Overview of the data platform

The data platform will:

- Aggregate and enrich diverse sources of external raw and aggregated data on cyber health
- Make that material available in a consistent form in bulk and via an API
- Do this in an efficient, fast and scalable manner

[2] Open Source Intelligence - meaning the data was already public in the first place, but might have been distributed and in different formats.



Bulk access. Bulk access is important for two reasons. First, credibility and trust – bulk access allows users to see the data behind the graphs. Second, maximizing value from the data via the “many minds principle” – bulk access provides power users such as technically minded cybersecurity professionals full access to the data in order to perform their own deep dives. This enlarges the potential for insights and also reduces technical demands that the platform support a myriad of possible analytical tools.

API access will be important for powering presentational applications such as the CyberGreen “Cyber Health Index” and data explorer (“stats.cybergreen.net”), data visualizations and other analytical and exploratory tools.

Key user story epics

Note: these user stories focus on the technical data **platform**, not the stats frontend or other “end-user” analysis or use (e.g. we do not have user stories like “X wants to see which country is performing best on cyber health” as that is a story for frontend).

Note: these are high level “epics” rather than classic lower-level user stories.

Note: for the sake of user story-telling, most user stories are written in the first form “As Alice, I want to ...”

Personas:

- Alice, a data wrangler loading data into the platform

- Bob, the administrator of the platform
- Charlie, a technical “client” (user) of the data platform e.g. front-end developer of the stats platform, a data scientist

Access to data

Access via an API

As Charlie I want an API for the data that allows me to pull aggregates of the data filtered by the main axes of interest (place, time, ISP) so that i can present these aggregates to users

- *Main axes:*
  - *Time (year, month, day)*
  - *Place (country, region)*
  - *Network / network owner: ASN/ISP*
- *Average response time for queries should be below 500ms*
- *Access should not require an API key but we are considering this in future so it should be a consideration for implementers*

As Charlie I want good documentation for the API so that I can use it without consulting an expert or its author (or spend long time trying to figure it out by hand).

Normalized data

As Charlie I want to access non-normalized as well as normalized data (e.g. infection counts divided by network size) so that I can present more easily compared data to my users.

Bulk Data

As Charlie I want to download major slices of the data (or even all data) easily so that I can perform my own analyses with my own tools.

Import data into the platform

[Scalability] As Alice I want to ingest large amounts (TBs) of raw CSV data easily and efficiently into the platform so that the data is stored in the platform and available from it

- *Focus is on source data in CSV format. There may need to be support for some other formats – details provided at a later stage.*
- *Will want to “filter” data on ingest … example: NTP data has false positives. We need to be able to filter these out.*

[Scalability] As Alice I want to be able to add new data sources and configure their ingestion quickly and easily so that new data sources can be added efficiently

As Alice, I want have a very robust (in the sense of failure tolerant) parsing mechanism for data feeds, so that I don't always have to adjust my parsers to (possibly slightly) changing data feeds. Alice recognizes that major changes in format of the data feed will require of course new parsers. But the parsers shall be tolerant enough to allow for slight mistakes in the format.

As Alice, I want basic performance metrics for the ETL cycles so that I can estimate the turn around times and monitor the system's performance.

As Alice, I want feedback of how many raw data log lines were parseable and how many were skipped/assigned to the "any" category.

Automatic Enrichment and Validation

As Alice I want to have my raw data automatically enriched, for example have ASN, country and location information added to a given by geolocating IPs

As Alice I want to have my raw data automatically checked to see if there are any anomalies or potential errors so that these can be flagged early and corrected

As Alice I want to carry out automatic normalization on my data so that it is "pre-cached" in the platform (as opposed to being performed on the fly as part of API access)

Automation

As Alice I want to automate the above so that we can collect and process different data feeds over time automatically without Alice's constant manual attention so that running the platform does not depend solely on Alice and can scale

As Alice, I want to be alerted if some data feed can't be collected or processed, so that I can take corrective actions in case some data feed does not work anymore.

SysAdmin and Platform Analytics

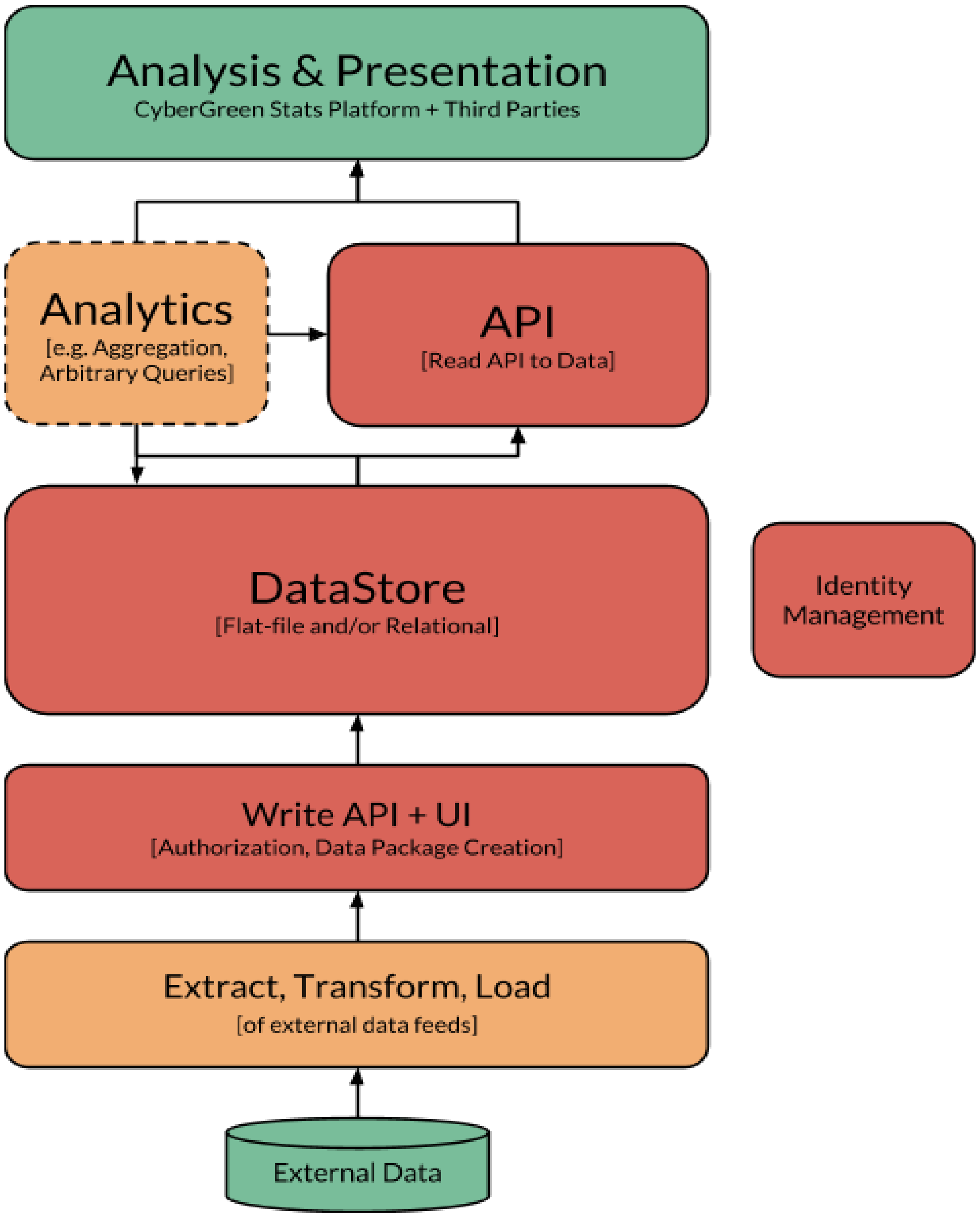
As Bob I want to see how many queries are happening a day so that I can report that to platform managers and sponsors.

As Bob I want to know how the platform is performing (e.g. no of queries, query response time) so that I can identify any performance issues.

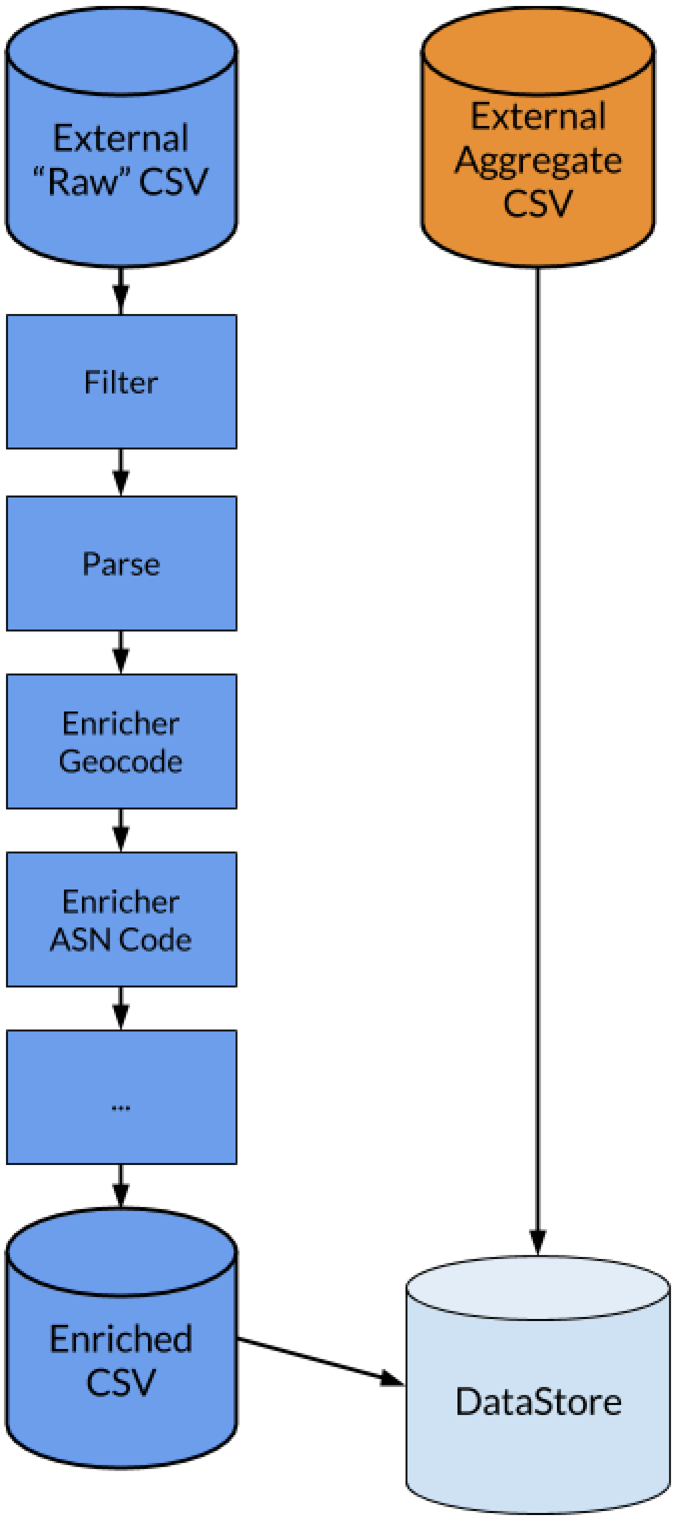
As Bob I want to know what the cost of running the platform is month to month so that I can budget.

Architecture

Overview of Platform Components and Relationship



Overview of Data ETL (Extract, Transform, Load)



API Specification

The Read API should follow RESTful principles. Rough outline as follows:

```
/place/{place-id}/
/risk/{risk-id}/
/risk/{risk-id}/{place-id}/

/aggregate/?place=x&risk=y
```

Here, “place” denotes a country or ASN.

Interfacing external raw CSVs

External raw CSVs shall be converted into a standardised internal format for further processing:

```
<timestamp>,<ip>,<risk>, (… any other field…)
```

Ideally, external raw data already comes in this format. If not, it shall be converted.

Interfacing externally aggregated CSVs

The platform can accept pre-aggregated CSVs. In order to be able to import these into the system, externally aggregated CSVs must adhere to the following format:

risk	year	place	count
openntp	2014	JP	330292
opensnmp	2015	BR	163306
opendns	2015	US	99630
opensnmp	2015	US	78197
opendns	2015	CN	78102
opensnmp	2015	KR	74141

Overview table

- The risk must be a unique ASCII string and there needs to be a (markdown) description of its meaning and implication.
- Place is the standard ISO 2 letter country code. See http://data.okfn.org/data/core/country-list/
- Count is the absolute number of occurrences of this risk

Appendix

Appendix A: References

[1] “The Cybergreen project: metrics and measurements for global cyber hygiene”, L. Aaron Kaplan 2016, www.cybergreen.net
[2] “Standards and tools for exchange and processing of actionable information”, ENISA, https://www.enisa.europa.eu/publications/standards-and-tools-for-exchange-and-processing-of-actionable-information
[3] “Shades of Green", Dan Geere, http://geer.tinho.net/geer.cybergreen.ii16.PDF, www.cybergreen.net

Appendix B: Glossary

Term	**Explanation**
User	A user of the platform. We address national CERTs, policy makers on a national level responsible for cyber security issues as well as network operators.
Risk	By Risk we refer to any of the two: vulnerability or infection.
Infection	An infection happened when a computer is under the control of an unauthorized third party and is running some kind of malicious software on its’ behalf.
Vulnerability	A vulnerability is a property of software. It is is a weakness which allows an attacker to reduce a system's information assurance.
Raw data	By raw data we mean log entries in the form `<timestamp>,<ip>,<risk>, (… any other field…)`
Aggregated data	Summarised data: counts of infections or vulnerabilities over time, ASN, country.
Data Store	Storage for raw and processed data. Likely a combination of flat-file and relational and/or NoSQL storage.
Data platform	The system for importing, cleaning, storing and making data available in bulk and via API.
Stats frontend	By frontend we mean a web UI or any other interface which accesses the data platform via API.
API	Application Programming Interface
ETL	Extract, Transform, Load

ASN	Autonomous System Number, a unique number assigned to an ISP (although an ISP might have multiple ASNs).
ISP	Internet Service Provider. Used synonymously with "network operator in this document".

Appendix C: Coding Standards

Development of the CyberGreen v2 should adhere to the following "coding standards":

- Languages: the platform must be built using one or more of the following languages C, Perl,Python, NodeJS (or Go)
    - C, Perl,Python or Go for ingestion and processing
    - C, Perl,Python or NodeJS (or Go) for web applications
- All code must come with tests with reasonable coverage.
- Code should be version controlled (in git).
- Continuous integration is strongly recommended (e.g. Travis CI).
- Recommend S3 (and AWS generally) for data platform
- Tools should be open source where possible and code developed by this project will be open source (AGPL or MIT license or similar. Preference on AGPL)
- We encourage reuse of existing tooling (and contribution back to those tools)

Appendix D: Existing Tools for data collection

Abusehelper

Summary: Abusehelper is a tool for collecting, filtering, enriching data feeds of IT security events. It was one of the first data-flow oriented tools using XMPP as a "message bus". It was developed by Clarified Networks with CERT.fi and the Estonian CERT. There are two versions of Abusehelper: open source and closed ("AbuseSA"). Many CERTs encountered that setting up the open source version was too time consuming. IntelMQ is the direct result of frustration with the open source version and (at that time) lacking documentation.

- URL: https://en.wikipedia.org/wiki/AbuseHelper
- Git: https://github.com/abusesa/abusehelper
- Documentation: https://github.com/abusesa/abusehelper/tree/master/docs
- Data format: https://github.com/abusesa/abusehelper/blob/master/docs/Harmonization.md

IntelMQ

Summary: IntelMQ ("intelligence message queue") is a tool developed by multiple European CERTs for automating the incident handling processing. It collects, filters, enriches, deduplicates multiple OSINT (open source intelligence) and non-open data feeds on IT security events. It is supported by ENISA and widely used (at the time of this writing).

- URL: http://github.com/certtools/intelmq
- Documentation: https://github.com/certtools/intelmq/tree/master/docs
- Feeds: https://github.com/certtools/intelmq/blob/master/intelmq/bots/BOTS
- Data format:
  https://github.com/certtools/intelmq/blob/master/docs/Data-Harmonization.md
  https://github.com/certtools/intelmq/blob/master/docs/Harmonization-fields.md

Note: the data format (called "Data Harmonisation Ontology" - not a particular meaningful name) between IntelMQ and Abusehelper is very similar (intentionally so).

Note2: IntelMQ can dump all of it's data into multiple output databases (or flat files) such as postgresql for further processing.

CSIRT Gadgets CIF (collective intelligence framework)
CIF is a tool which came out of the REN-ISAC community in the US. It was built for and by academic network operators / IT security groups to monitor, collect and act upon IT security data feeds ("should I block IP address x.y.z. from attacking my education university network?")

- URL : http://csirtgadgets.org/collective-intelligence-framework
- Code: https://github.com/csirtgadgets/cif-v1
- CIFv3: https://github.com/csirtgadgets/bearded-avenger
- Data format: *No documentation for the CIF format*
- Note: basically CIF is end of life for Cybergreen.

Megatron
Megatron is a tool developed by the swedish CERT to automate incident handling in batch mode. Data of IT security data feeds is collected in batch, processed and sent out ("notifications") to end users, network owners etc. It is written in Java and used by the Netherlands and Sweden.

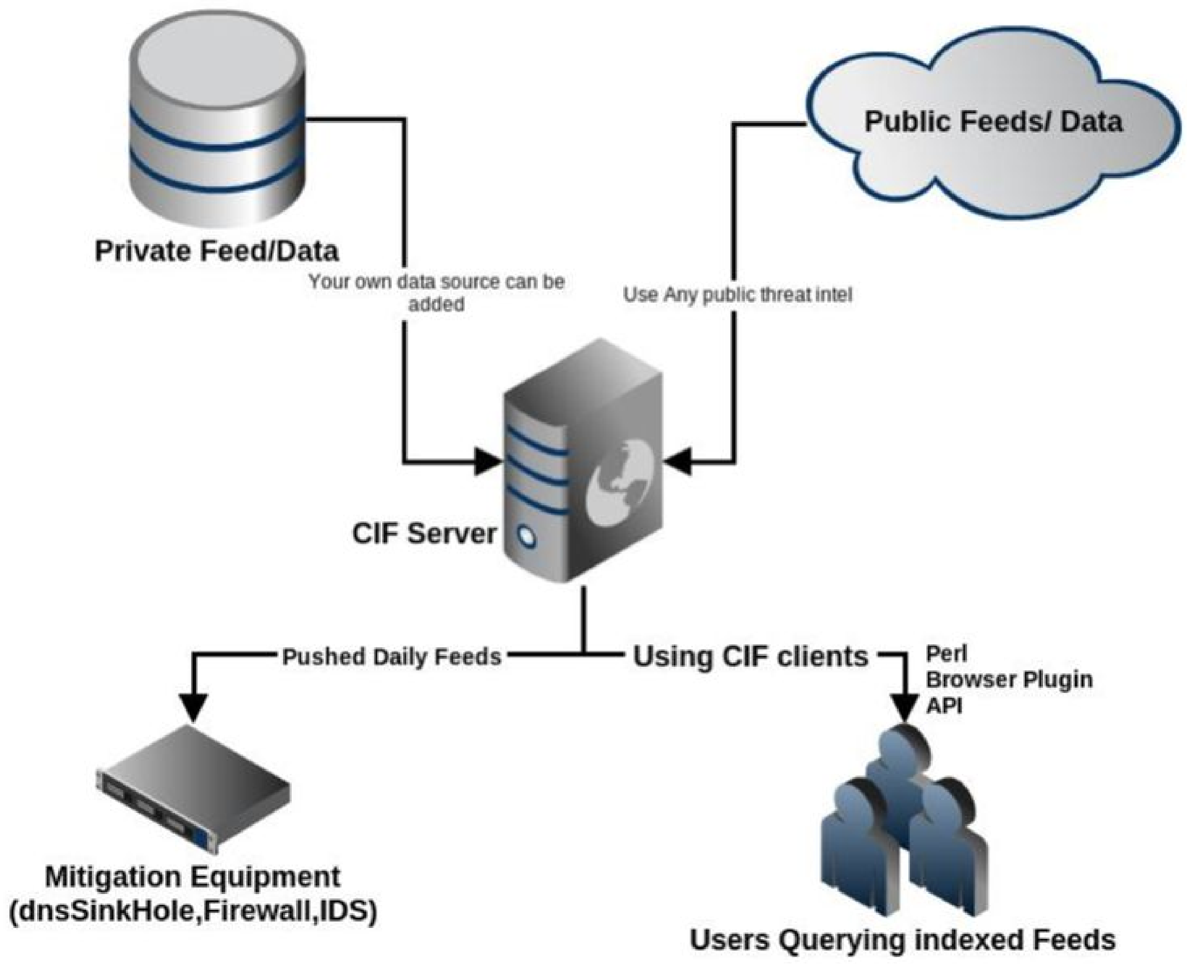
- URL: https://github.com/cert-se/megatron-java

The most common mechanism however still is...
Flat files in any type of CSV format, shared via scp via cron jobs or email.

Appendix E: CyberGreen v1

The first version of the Cybergreen project assumed, it would collect all possibly collectable data on infections or vulnerable systems on the Internet, visualise it and re-distribute it to countries, CERTs or ISPs.

At the beginning of the project, CISRT Gadgets was tasked to create a Cybergreen Portal page based on their CIF framework. The following figures show a screenshot of platform version 1 (stats frontend) as well as an initial architecture diagram.


Private Feed/Data
Public Feeds/ Data
Your own data source can be added
Use Any public threat intel
CIF Server
Pushed Daily Feeds
Using CIF clients
Perl
Browser Plugin
API
Mitigation Equipment (dnsSinkHole,Firewall,IDS)
Users Querying indexed Feeds

Platform version 1 had very ambitious goals to collect all kinds of OSINT[3] as well as private raw data, store it in Elastic Search DB, producing a “CyberGreen Index”, and finally redistributing the raw data. The “CyberGreen Index” (version 1) should be a measure of how well a country is improving. However, it was only employed on countries.

[3] Open Source Intelligence