ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
INFORMÁTICA

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

Complemento de VirusTotal para Maltego

VirusTotal plugin for Maltego

Realizado por
Alex Javier Porras Palma

Tutorizado por
Ana Nieto Jiménez
Juan Antonio Infantes Díaz

Departamento
Lenguajes y Ciencias de la Computación

Universidad de Málaga
Málaga, Septiembre de 2020

Fecha de defensa:
El Secretario del Tribunal

(This page has been intentionally left blank)

Resumen

Maltego es una herramienta OSINT (Open Source Intelligence) de análisis gráfico de enlaces para la recogida y conexión de información en tareas de investigación. Siendo una herramienta de propósito general, este trabajo persigue su aplicación a la investigación de software malicioso (malware). Maltego usa "transformadas" para definir relaciones. Con el objetivo de ampliar el contenido que se puede relacionar, se propone, con la ayuda de la librería "Maltego-TRX", definir transformadas que incluyan las relaciones diponibles en VirusTotal. VirusTotal es una herramienta online que inspecciona archivos, dominios y URLs con escáneres antivirus y servicios de *blacklisting* además de diferentes herramientas para la extracción de señales. La información disponible en VirusTotal se puede visualizar usando VirusTotal Graph que, aún siendo de gran utilidad para el cometido de análisis de malware, no cuenta con todas las funcionalidades de Maltego. Además, se persigue hacer posible la importación a Maltego de los grafos generados en VirusTotal Graph de forma transparente al usuario. Con estas mejoras, el uso de Maltego podrá extenderse al análisis de malware de forma más clara, dado que se estarán volcando los datos de VirusTotal con la interpretación específica que el análisis de malware requiere.

Palabras clave: ViruTotal, Maltego, grafos, malware, investigación digital, dominios, dirección ip, transformadas, URLs.

Abstract

Maltego is an open-source intelligence (OSINT) and graphical link analysis tool for gathering and connecting information for investigative tasks. Being a general-purpose tool, this master's thesis aims to its application to malware investigation. Maltego uses "transforms" to define relationships. With the goal of expanding the relationships offered by Maltego, we propose using the library "Maltego-TRX" to define transforms that include the relationships available in VirusTotal. VirusTotal is an online tool that inspects files, domains, and URLs with antivirus scanners and blacklisting services in addition to different tools for signal extraction. The information available at VirusTotal can be visualized using VirusTotal Graph, which is of great utility for malware analysis, however, it does not offer all of the features Maltego includes. In addition, we suggest the option of importing the graphs created in VirusTotal graph into Maltego in a transparent way for the user. With these improvements, the use of Maltego can be extended to malware analysis in a clearer way, given that VirusTotal data will be included in Maltego with the specific interpretation malware analysis requires.

Keywords: VirusTotal, Maltego, graphs, malware, digital investigation, domains, ip address, transforms, URLs.

Contents

List of Figures

List of Tables

(This page has been intentionally left blank)

Chapter 1

Introduction

We live in the information era, where the volume of digital data is increasing rapidly. Organizations have moved their business processes to digital formats, including their computing assets as fundamental parts of risk management. The number of users of social networks increases everyday and the use of Internet of Things (IoT) devices has already taken off, having increased the load of digital information with billions of devices. Specially during the times of the COVID-19 pandemic, companies need to rely on remote access to their processes, increasing the number of business related interactions over the internet. This exposes companies to more cybersecurity threats, for example, hospitals falling victim to ransomware attacks (Jercich, 2020).

Open Source Intelligence (OSINT) refers to the collection, processing, and correlation of all the information that is publicly available (Hassan, 2018), for example, social networks, forums, blogs, etc. Although OSINT also encompasses offline sources (e.g. television, radio, newspapers, etc), online sources comprises the largest segment. OSINT sources are distinguished from other forms of intelligence because they must be legally accessible by the public without breaching any copyright or privacy laws, making its gathering less expensive and less risky than traditional intelligence activities. As the volume of available data increases, organizations and business corporations are starting to rely on OSINT sources and tools. For instance, companies may be concerned about data leaks and social engineering tactics, so they keep an eye on external sources, such as social networks or forums to check on suspicious event.

Precisely, one of the main areas in which OSINT tools have been adopted is *cybersecurity* and *cyberdefense* (Pastor-Galindo, Nespoli, Gomez Marmol, & Martinez Perez, 2020). As information and communication systems are constantly being attacked by agents trying to take them down, OSINT research becomes crucial in preventing *cyberattacks*. For example, data mining techniques may be used to analyse these attacks and find common patterns, patterns that can be used to identify Indicators of Compromise (IoC) to feed Threat Intelligence

Systems. Considering OSINT as a source of information for tracebacks and investigations, forensic digital analysis can complement the digital evidence left by an incident.

This chapter introduces the context and motivation for this master's thesis, and details the main objectives to be addressed, as well as the methodology and structure chosen.

1.1 Context and motivation

There are multiple tools that can be used to work with OSINT data (see Chapter 2), one of the most powerful tools is Maltego. Maltego is an interactive data mining tool that renders directed graphs for link analysis. The tool is used in digital investigations for finding relationships between pieces of information from various sources located on the Internet (Maltego, 2020b). A few examples of Maltego uses cases are crime scene analysis (AndyF1, 2018a), cell phone analysis and geolocation (AndyF1, 2018b), and identity tracking of cybercrime actors (Esman, 2018). In particular, this master's thesis focuses on the use of Maltego for malware and cybersecurity analysis.

Maltego links information through what is called a "transform". A transform is a small piece of code that fetches related information for a given input and formats the results to be returned as entities of Maltego (Maltego, 2019d). Maltego shows relations between entities of a very diverse nature (e.g. people, phone numbers, domains, IP addresses), as well as custom entities, which may represent any kind of data. Any entity in Maltego has a set of default transforms associated with it (see Figure 3.6).

One of the biggest problems is to limit the investigation to a specific scope or purpose. This happens because predefined transforms offer their own interpretation of the values to be included in the graph and dump all the data without considering the specific purpose. This happens not only with Maltego but also with other tools using multiple OSINT sources.

The source of the transform data becomes crucial in narrowing the domain of the investigation. That is why tools like VirusTotal, not only capable of identifying links between entities, but also identifying the knowledge of expert sources, are very useful to integrate. VirusTotal inspects items with over 70 antivirus scanners and URL/domain blacklisting services, in addition to a myriad of tools to extract signals from the studied content. Any user can select a file from their computer using their browser and send it to VirusTotal (VirusTotal, 2020a) or check if the file was previously uploaded to the system using the hash of the file as a parameter in the search engine. One of the methods to consult malware samples (or just samples) is via API, which allows us to make calls from a programming language and obtain the results of the diagnosis and extracted signals.

The data available at VirusTotal can be visualized using VirusTotal Graph. It understands the relationship between files, URLs, domains, IP addresses, and other items encountered in an ongoing investigation. With it, users can pivot intelligently over any of the malware artifacts in their graph and synthesize their findings into a threat map that they can share with their colleagues (VirusTotal, 2020g). Nevertheless, it does not consider the range of entities and other functionality of requests to open sources in a general way as Maltego does.

With this in mind, the goal of this project is to extend the transforms offered by Maltego with the information available at VirusTotal (see Figure 1.1). More specifically, this project focuses on VirusTotal Graph, including all its relations into Maltego, searching for a map between its entities. Therefore, this project will empower Maltego to better contribute to the cybersecurity scope. The solution this project provides will be called "VT private transforms" from now on. In this master's thesis, it is described how they are implemented to help its application to other platforms.

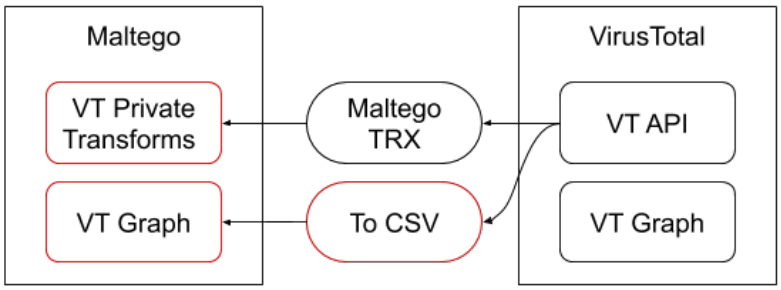


Figure 1.1: Basic components of the solution. **Red figures** illustrate the contributions of this master's thesis. Rounder figures show python libraries/scripts.

1.2 Objectives and structure

This master's thesis consists in the development of a VirusTotal add-on for Maltego. The specific goals of this project are:

O1 Introduce Maltego as an OSINT analysis tool and provide a guide of its use, exploring its current capabilities in the field of *cybersecurity* and malware research. There are a plethora of open sources that are already available in Maltego, although not all of them for free.

O2 Introduce the most relevant features of VirusTotal as a source of threat intelligence for Maltego. VirusTotal Graph already shows relations between API objects in a graph manner and it will serve as guidance to what can be implemented in Maltego. Besides, as of the date of this master's thesis,

the new capabilities of VirusTotal are not fully integrated by the current tools.

O3 Build custom transforms with VirusTotal data. Custom transform can be built using "Maltego-TRX", a python package created by Paterva. This project uses VirusTotal premium API in its newest version (i.e. v3) to explore all of what VirusTotal offers. VirusTotal basic entities have a matching representative in Maltego, but VirusTotal Graph shows additional properties in some of the nodes, such as the detection ratio in files and URLs. In cases where Maltego cannot offer a similar experience, additional transforms with custom entities are provided to depict these properties.

O4 Create a mechanism to import VirusTotal graphs into Maltego, making it possible for the user to take the work made in VirusTotal Graph without having to manually replicate it. The goal is to make it as transparent to the user as it can get.

In order to address the previous objectives, this master's thesis is structured as follows. Chapter 2 describes the state of the art, highlighting the contributions in this work compared with the current solutions. The objectives **O1** and **O2** are accomplished in Chapter 3, which provides context and the necessary previous knowledge to understand the project. The objectives **O3** and **O4** are completed in Chapter 4, which is split into sections detailing the development of the private API transforms and VT graph import. Lastly, Chapter 5 summarizes the results and shows the conclusions.

Chapter 2

State of the art

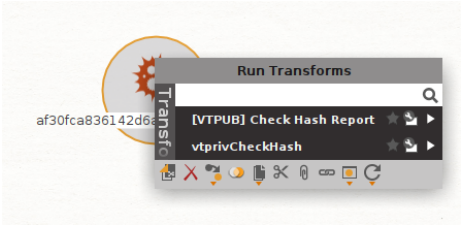
The solution proposed in this project responds to a very specific demand: an improved integration between VirusTotal and Maltego. This chapter analyses current alternatives that provide a similar service and compares them to what this project offers. These include already available VirusTotal transforms in Maltego (Section 2.1-2.2) and examples of OSINT tools that draw data from VirusTotal (Section 2.3-2.5). A summary of the analysis is provided in Section 2.6, where the main contributions of this project are highlighted.

2.1 Open source private API transform sets

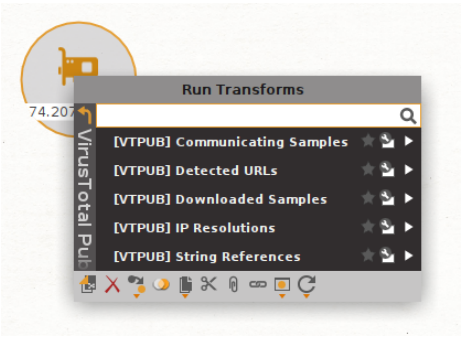
There are a bunch of custom transforms made by individuals that use the VT private API, see (tampererer, 2018) and (cmlh, 2016), for example. Others are listed in the VirusTotal support page, like (Lookingglass, 2014) and (Yip, 2015). However, these sets are drawn from the APIv2. This version does not have relationship endpoints, meaning that the transforms depend on the interpretation of the authors. On top of that, these transform sets are not being maintained and are outdated.

2.2 Malformity Labs' VirusTotal transform set

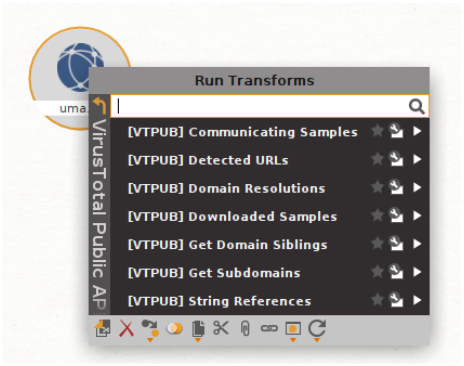
Malformity Labs offers a VirusTotal transform set that includes basic relationships that can be used with a public API key (i.e. a free account). The main advantage of this set is that it comes with Maltego and can be installed right away. However, there are only a few transforms available in this set. Classified by the type of origin node, Figure 2.1a shows hash transforms, Figure 2.1b shows IP address transforms, Figure 2.1c shows domain transforms, and Figure 2.1d shows URL transforms. It lacks the majority of transforms the private API offers and since it uses the second version of the API it is bound to Malformity Labs' interpretation of the reports. Note that some transforms, like "[VTPUB]



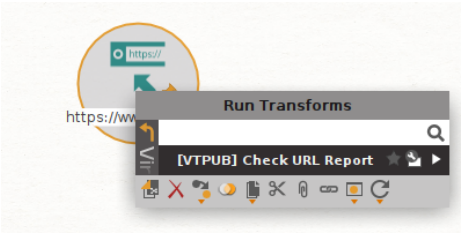
(a) file transforms.



(b) IP address transforms.



(c) domain transforms.



(d) URL transforms.

Figure 2.1: Malformity Labs' VirusTotal transforms.

Check Hash Report", do not give you any hint about what relationship they are expanding upon, so it is not very intuitive.

2.3 Sn0int

Sn0int is a semi-automatic multiplatform OSINT framework and package manager. It was built for IT security professionals and bug hunters to gather intelligence about a given target (Sn0int, 2020). It has some modules that draw data from VirusTotal, nevertheless, it does not offer a visual representation as Maltego does, and the modules it currently has only cover a small part of the VirusTotal data.

2.4 SpiderFoot

SpiderFoot is a reconnaissance tool that automatically queries over 100 public data sources (OSINT) to gather intelligence on IP addresses, domain names, e-

mail addresses, names and more (SpiderFoot, 2020). The tool is able to represent data using different kinds of charts and tables, even graphs (see Figure 2.2). Instead of "transforms", SpiderFoot uses "modules" to discover new data. It is also able to interact with VirusTotal by providing the user's API key, although it does not offer the full functionality of the API. For example, file hashes and its relationships in VirusTotal are not included, rather, it focuses on online entities like IP addresses, domain names, websites, etc.

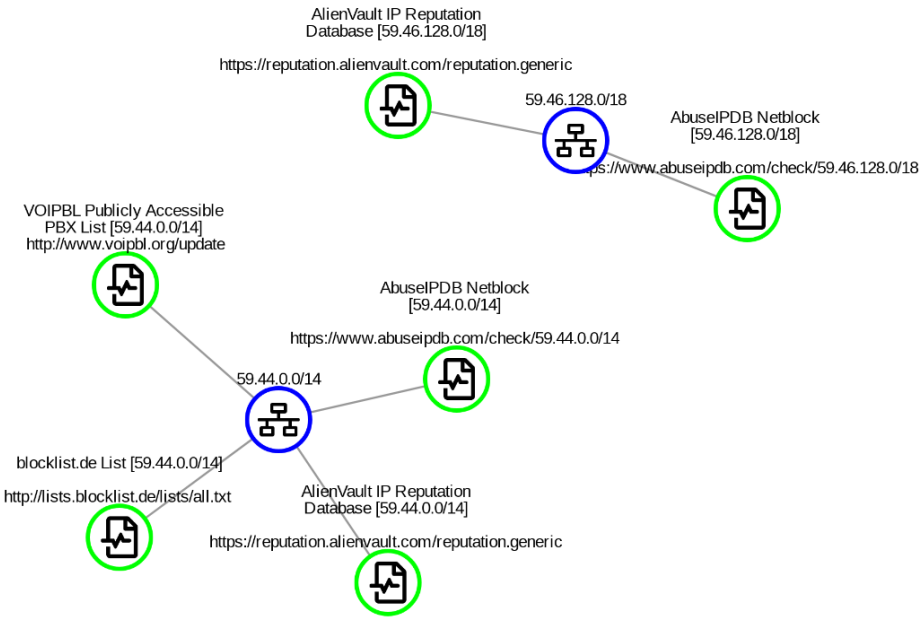


Figure 2.2: SpiderFoot example graph. Source: NixIntel

SpiderFoot can either be downloaded from Github and run a local web UI or use SpiderFoot HX, which is cloud-hosted and requires no installation.

2.5 Lampyre

Lampyre is a powerful analysis assistance tool. Analytical data is processed using custom requests, which enable obtaining and enriching data from different data sources, including your own offline sources using different analytical tools (Lampyre, 2020). It is also able to show graph like visualizations in what are called "schemas". Although it includes the possibility of making requests to VirusTotal and even contain hash nodes, it works with the public API.

Lampyre is a powerful tool with an extensive set of features. It can be applied to a variety of fields apart from cyber threat intelligence, for example, financial analysis (Lampyre, 2020). It is even possible to study graphs on top of maps.

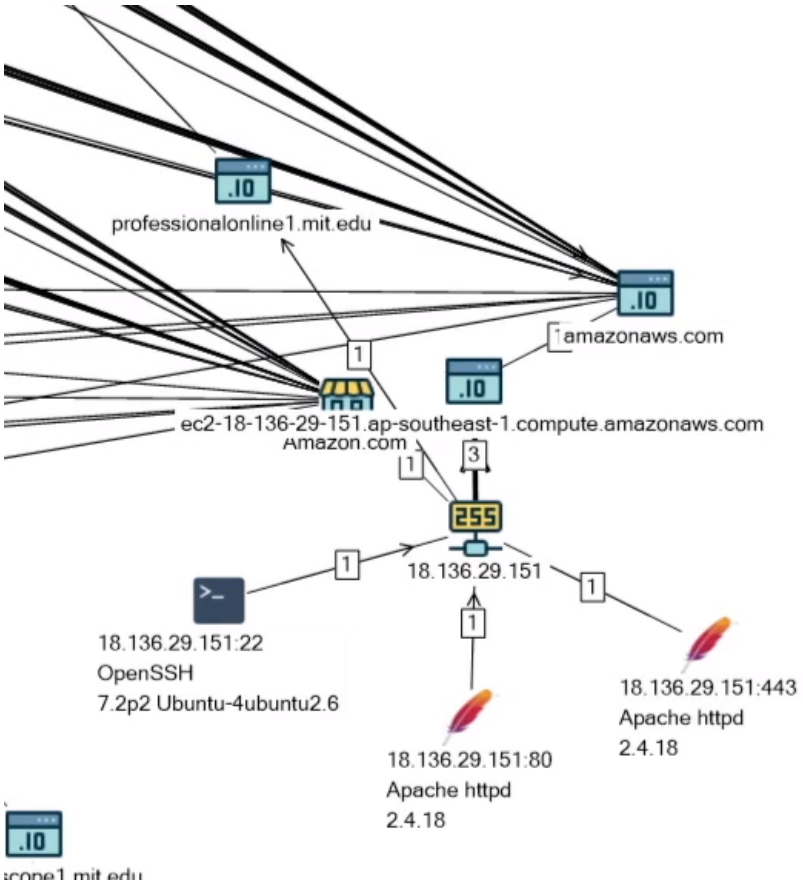


Figure 2.3: Lampyre graph example. Source: Lampyre.

2.6 Synthesis and main contributions

Table 2.1 summarises the set of characteristics more relevant to this project in order to highlight the differences between the different services shown in this chapter. These characteristics represent the degree of use of VirusTotal data the OSINT tool is able to integrate. Crucial to the purpose of this master's thesis since the main objective is to extend Maltego with all of what VirusTotal can offer.

The column for "VTGraph integration" refers to the degree in which we can reproduce what we can do in VirusTotal Graph. In order to complete this, a set of levels are defined as follows:

- Level 0. We cannot mimic VirusTotal Graph at all.
- Level 1. We can use some nodes and some relationships.

- Level 2. We can use all main nodes and some relationships.
- Level 3. We can use all main nodes and all relationships.
- Level 4. We can import public VT Graphs.

Note that Sn0int doesn't use the API properly, instead, it makes a request to the UI endpoint (kpcyrd, 2020). "VT Pivate transforms" clearly stands out over the alternatives because it achieves full integration with the newest version of the VirusTotal API, to the point where it is able to import VTGraphs into Maltego only by typing the graph ID.

Table 2.1: Comparisons between the SOTA alternatives.

	VT API		**VTGraph integration**		
	Access	**Version**		**Up to date**	**Usability**
Malformity Labs' transforms	Public	2	2	Yes	Easy
Open Source transforms	Private	2	2	No	Medium
Sn0int	Private	None	0	No	Hard
SpiderFoot	Public	2	1	Yes	Medium
Lampyre	Public	2	2	Yes	Medium/Hard
VT Private Transforms	Private	3	4	Yes	Easy

It was decided to use Maltego for this master's thesis because of its popularity, intuitiveness, and steep learning curve. Therefore, the main contributions of the solution proposed in this master's thesis (VT Private Transforms) regarding the previous works are that it uses the complete set of relationships available at VirusTotal, it is up to date with the latest release of the VirusTotal API, it is easy to install since it uses the official way to create transforms in Maltego and it is able to import VirusTotal graphs into Maltego. Note that this solution will improve the capacity of Maltego to work with VirusTotal. Moreover, the code of the transforms, entities, and import script will be available at a public Github repository, where anyone can download it and test it. In addition, it is very important to note that the development of the solution proposed has been supervised by VirusTotal members, in order to guarantee the maintenance and usability of the solution. The details about the main platforms involved in the development of VT Private Transforms (Maltego and VirusTotal) are described in Chapter 3.

(This page has been intentionally left blank)

Chapter 3

Background

In order to design the VT Private Transforms, it is critical to understand the overall use of both of the major tools present in this project: Maltego and VirusTotal Graph. The chapter starts by introducing Maltego and its different versions. Later, it goes through what entities are, including examples of the ones that are relevant to this master's thesis, followed by an introduction to transforms and its usage. Before jumping to VirusTotal, it explains how to add custom transforms to Maltego. The last sections introduce VirusTotal. Starting with a general description of the tool, it then focuses on VirusTotal Graph, ending with a comparison between the different versions of the API.

3.1 Maltego

Maltego is an OSINT application that provides a platform to not only extract data but also represent that data in a format that is easy to understand as well as analyze (Chauhan, 2015). To use Maltego, Maltego Desktop Client needs to be downloaded. It is available in three versions (as of the date of this master's thesis):

- Maltego Classic. Commercial version returning up to 10,000 results per Transform.
- Maltego XL. Commercial version for large investigations to view up to 1,000,000 pieces of information.
- Maltego CE. Community version for non-commercial use available for free, returning up to 12 entities per Transform.

This project uses the community version for demonstration purposes. Compared to other OSINT software (see Chapter 2), Maltego stands out for its GUI (see Figure 3.1), which is the main feature of the tool.

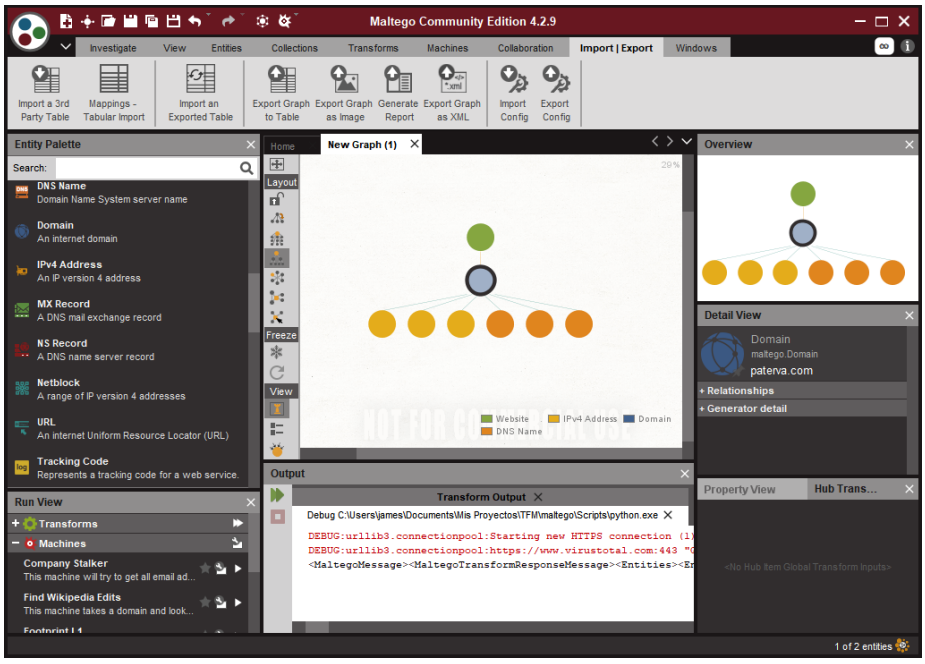


Figure 3.1: Overview of Maltego layout with an example graph.

This section provides a view of the aspects of Maltego that are relevant to this project. A detailed guide of the whole tool, including the installation process, can be found at `https://docs.maltego.com/support/home`.

3.1.1 Entities

Entities in Maltego are represented as nodes on a graph. They are the way in which Maltego expresses information. Maltego takes entities as input to obtain more entities in its analysis process. The entities in the palette are categorised into groups, with the main categories being Infrastructure and Personal (Maltego, 2019f).

There are three important aspects of an entity:

- The type, which is the information that the entity is representing.
- The value, which is the main information that is held in the node.
- The properties, which are additional information fields for the entity.

Every entity has its meaning, and since custom entities can be created, there can be an entity for every piece of information the user has an interest in. Maltego can be used for any kind of investigation, online and offline. Its main focus is

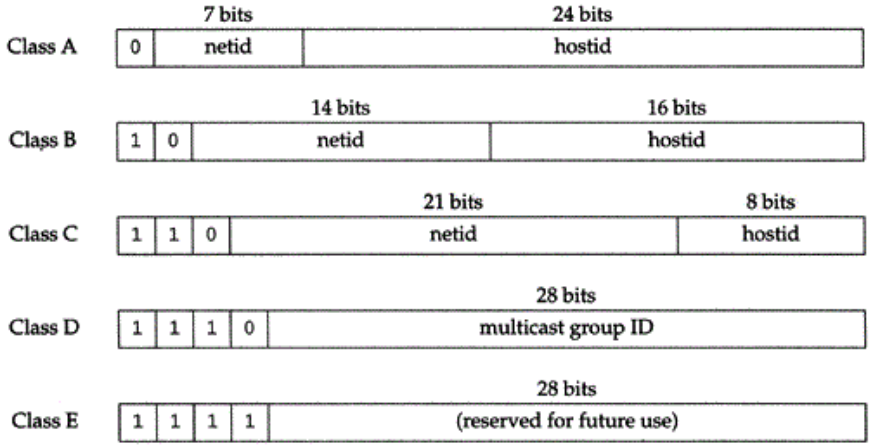


Figure 3.2: The five different classes of Internet addresses (Stevens, 1993). Source: TCP/IP Illustrated Vol. 1.

to help users organize the information they already have. As the topic of this project lies within the use of Maltego for cybersecurity, the following paragraphs introduce the default set of entities that Maltego has for such purposes.

IPv4 Address

IP addresses are often the main component included in Indicators of Compromise (IOCs), which are used for network defense, identification, response to cyber incidents, and to identify who or what is connected to your network or device (Center for Internet Security, 2020a). They could also be used for geolocation, although agents may use proxies or infrastructure hosted in different countries.

Internet Protocol version 4 refers to the 32-bit identifier of a network component. It still routes most internet traffic today (Maltego, 2019a). These 32-bit addresses are normally written as four decimal numbers, one for each byte of the address. This is called *dotted-decimal* notation (Stevens, 1993). Figure 3.2 shows the different classes of Internet addresses.

As two interfaces connected to the internet cannot share the same IP address, there exists one central authority in charge of designating addresses for networks connected to the worldwide internet. The Internet Assigned Numbers Authority (IANA) has authority over all number spaces used on the internet, including IP address space (i.e. IPv4 and IPv6) and Autonomous System (AS) numbers (RIPE NCC, 2016). Inside the IANA, RIPE NCC serves Europe, Central Asia, and the Middle East.

Due to the perpetual and high demand for internet connectivity, the IPv4 address space is on the verge of imminent exhaustion, given that the Internet has already reached enormous proportions (Beeharry & Nowbutsing, 2016). IPv4 exhaustion motivated the definition of IPv6, which uses 128 bits, instead of 32 bits. However, most internet traffic is still routed using IPv4. The IPv6 protocol is in a transition phase, having its traffic volume been increasing in the past years (Ciflikli, Gezer, & Ozsahin, 2012).

Domain

Humans work best using the *name* of a host, rather than its IP address directly. A domain name identifies a network resource that communicates over the internet (e.g. "google.com") (Maltego, 2019a). A domain must be registered by a person or an organisation.

Cybercriminals and state-sponsored actors are capitalizing on vulnerabilities that originate from splitting management responsibilities to launch social engineering attacks in an attempt to hijack an organization's entire web presence (D'Angelo, 2019). Most of these attacks are occurring by targeting critical nodes, such as domain name registrars, which can lead to loss of business-critical assets, data breaches, reputation loss, etc.

The domain name space is hierarchical, similar to the Unix file system (Stevens, 1993). Every node (see Figure 3.3) has a label of up to 63 case insensitive characters. The *domain name* of any node is built with the path starting with that node all the way to the root of the tree using dots to separate the labels. Every domain name must be unique (i.e. the concatenated list of labels), but the same label can be used in different domain names. A domain name ending with a dot is considered an *absolute domain name* or *fully qualified domain name* (e.g. `sun.tuc.noao.edu.`).

The top-level domains are divided in three categories:

- `arpa`, a special domain used for address-to-name mappings.
- three character domains, also known as organizational domains (e.g. "com", "edu", "org", etc).
- two character domains, based on the country codes found in ISO 3166 (e.g. "es" for Spain).

Each node within the domain name hierarchy is assigned to an *authority*, an organization or person responsible for the management and operation of that node (Aitchison, 2011). The authority of a node can *delegate* the authority of descendants to that node. The authority of the root node belongs to the Internet Corporation of Assigned Numbers and Names (ICANN). ICANN created

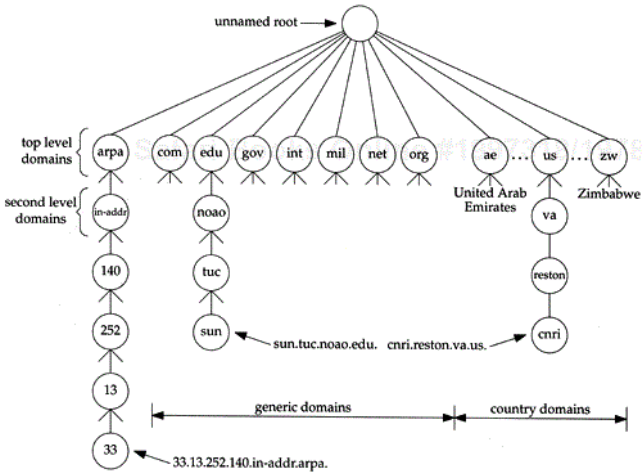


Figure 3.3: Hierarchical organization of domains (Stevens, 1993). Source: TCP/IP Illustrated Vol. 1.

the concept of *accredited registrars*, organizations with limited responsibilities for the sale and administration of parts of the domain name hierarchy.

Let us consider the following name: `www.uma.es`. In this case, the registrar delegated the ownership of `uma.es`. The leftmost part, the "www" in this case, is called a hostname and denotes a web service only by convention (Aitchison, 2011). Different examples of hostnames include "ftp" for the file transfer protocol server or "smtp" for mail.

Within the domain name hierarchy (see Figure 3.3), descendants of a node are considered its *subdomains*, for example, `uma.es` subdomains include `babel.uma.es`, `jabega.uma.es` and `webpersonal.uma.es` to which the domain owner of `uma.es` can delegate the authority. All nodes at the same level in the hierarchy that share the same parent are called *domain siblings*, in the previous example, all `uma.es` subdomains are domain siblings.

DNS Name

DNS stands for Domain Name System, a DNS name is an entry inside a domain zone file, which is a file that contains all the names associated with a domain (Maltego, 2019a), for instance, `www.google.com` is a name registered under the domain `google.com`. Every DNS name is or should be, connected to an IP address.

A malicious agent may look for an open DNS resolver to launch a distributed denial of service (DDoS) attack, either against the resolver itself or against other

systems. The target receives a deluge of DNS replies from all over the Internet. DNS replies can be spoofed, or created with false information, to redirect users from legitimate sites to malicious websites (Jones, 2019).

Names like `www.abc.com` are called "A records" or "Canonical Name (CNAME) Records" depending on whether they directly map to an IP address (A Record) or they are an alias (CNAME Record). The most common use of CNAME Records is where a host has more than one possible name (Aitchison, 2011), for example, if the name of a server is `server1.example.com` but it hosts both web and FTP service, `www.example.com` and `ftp.example.com` must resolve to the same IP address as `server1.example.com`. One option is to use various A records, but a lot of people choose to use an A record for the real name (server1) and use CNAME Records for the rest.

MX Record

The MX (mail exchange) records contain information about where mail should be sent for email addresses at the domain, for example, mail for anyone@abc.com should be sent to mx1.abc.com (Maltego, 2019a).

Apart from the name, one of the defining features of an MX Record is its preference. This value is set to indicate the relative priority of the mail server it defines. The lower the number, the more preferred the server. Setting a bigger value than others is the classic method of defining a backup mail server. In the event that the first mail server is not available, the backup mail server would be used (Aitchison, 2011).

Today, a rogue domain with an MX-record represents danger on an entirely different scale. It is a launching platform for an imminent email attack or attacks that can steal funds or trade secrets, infect business networks with malware or ransomware, or give criminals deeper access into company networks (Sachs, 2016).

NS Record

The NS records (name server) show who keeps the zone files for the domain (Maltego, 2019a). NS Records are used to list all the name servers that will respond authoritatively for the domain (Aitchison, 2011). Each zone has an NS Record defined in two places: the child zone and the parent zone, the child zone contains NS Records defining the authoritative name servers for the zone, while the parent's NS Records are called the *delegation point*, used to reference the authority of the domain zone. There should be at least two authoritative name servers for every zone.

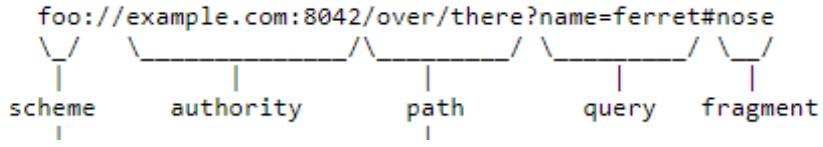


Figure 3.4: URL scheme example. Source: RFC3986.

Unauthorized changes to one NS record (most websites have at least two) allow attackers to take control of a victim's DNS server and serve results as they wish or set custom time-to-live (TTL) timing, among others (WhoisXMLAPI, 2020).

URL

A reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. A URL is a specific type of Uniform Resource Identifier (URI) (Wikipedia contributors, 2020d). A URI is a compact sequence of characters that identifies an abstract or physical resource (Berners-Lee, Fielding, & Masinter, 2005). The term "Uniform Resource Locator" refers to the subset of URIs that additionally provides a means of locating the resource by describing its primary access mechanism, for example, its network location.

The generic URI syntax consists of a hierarchical sequence of components referred to as the scheme, authority, path, query, and fragment (see Figure 3.4).

Not all URLs are harmless, malicious agents can trick users into doing something that wasn't intentional, for example, revealing accounts information using phishing scams of "typosquatting". "Typosquatting" takes advantage of typographical errors (i.e. "typos") introduced by users when URLs are typed directly into the address bar. It is common for actors to use *typosquatted* domains to display custom images or text, conduct scams, capture sensitive data, or infect users with malware (Center for Internet Security, 2020b).

Tracking Code

This entity represents a unique code that can be found in the JavaScript of some websites. Services that include such codes include Google Analytics, AdWords, PayPal Donate buttons, etc. These codes can be used to link websites together based on the owner of the tracking code (Maltego, 2019a).

Location

Geographical location, the location entity is officially made up of three fields - city, region, and country. Different transforms provide different levels of accuracy (Maltego, 2019b). Locations are important in the investigation because they may help identify the source of attacks.

Hash

A hash function is a mathematical function that maps data of arbitrary size to fixed-size values (Wikipedia contributors, 2020b). One of the main properties of hashes that are used in information security is that they are one-way functions, meaning that given a message digest n, it is very difficult to find m such that $f(m) = n$. The hash functions used in this project are "MD5" (128 bits), "SHA1" (160 bits) and "SHA256" (256 bits).

Port

A TCP/UDP network port. A port is a way to access a networked service on a computer. Each computer has 65,535 ports that can be either open or closed at any time (Shaw, 2015). Some services have ports associated with them by default, for example, HTTP has port 80 assigned to it. Ports are commonly specified by putting a colon (:) after an IP address (e.g. 27.0.0.1:22). Unidentified open ports may lead to security issues for users and organizations.

Service

Network service (port and banner combination). A network service is an application running at the network application layer and above that provides data storage, manipulation, presentation, communication, or another capability which is often implemented using a client-server or peer-to-peer architecture based on application layer network protocols (Maltego, 2019c). Often, services are the target of cyberattacks, aiming to interrupt them and thus, causing damage to organizations.

3.1.2 Transforms

Transforms are the fundamental blocks of Maltego, they are what enables linking between entities. Transforms come in packages accessible through the Transform Hub (see Figure 3.5). By default, "Paterva CTAS CE" is the only transform package installed. Running a specific transform will take the selected nodes as input and search for related output nodes. For example, Figure 3.6 shows that selecting "To DNS Name - MX (mail server)" will output nodes representing the mail server domains, as seen in Figure 3.7. This subsection explores some of the most relevant transforms available by default in Maltego.

IP address to Location

It was mentioned in previous sections that IP addresses could be used for geolocation, which may lead to identifying the geographical origin of internet traffic or even attacks. This transform takes an IP address to a location with the MaxMind free city database. There are two levels of precision to this transform, to get only the country of the IP address, or the country and the city.

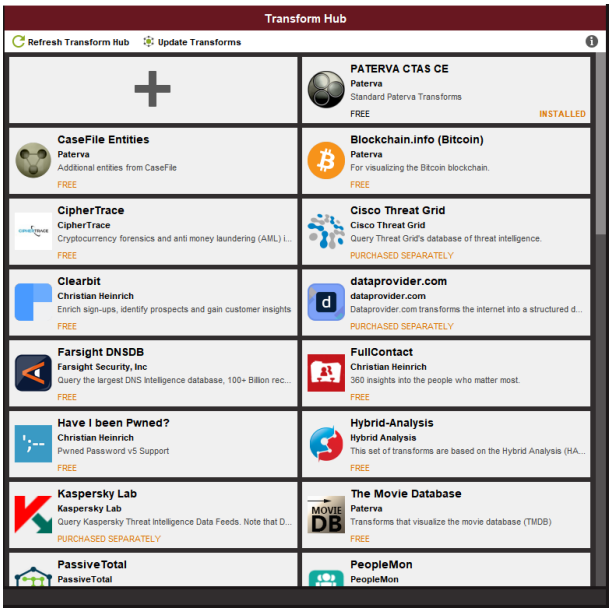


Figure 3.5: Maltego Transform Hub

IP address to DNS Name

Identifying an IP address as an IoC allows us to suspect all the DNS names it is associated with. This transform reverse resolves an IP address to its DNS Name or looks up historical DNS resolutions.

IP address to Domain

This transform returns shared domains on an MX or NS record using historical/passive DNS. Passive DNS is a system of record that stores DNS resolution data for a given location, record, and time period. This data set allows for a time-based correlation based on domain or IP overlap (riskiq, 2020).

IP address to entities from WHOIS

Obtaining the WHOIS info of a suspicious IP address may give important insights about what actors are behind it. This transform obtains WHOIS information from the IP address, then parses it to create entities using NER.

Domain to DNS Name

There are various transforms that return DNS names from domains, the main difference between them is the way they do it, for example, there is a version

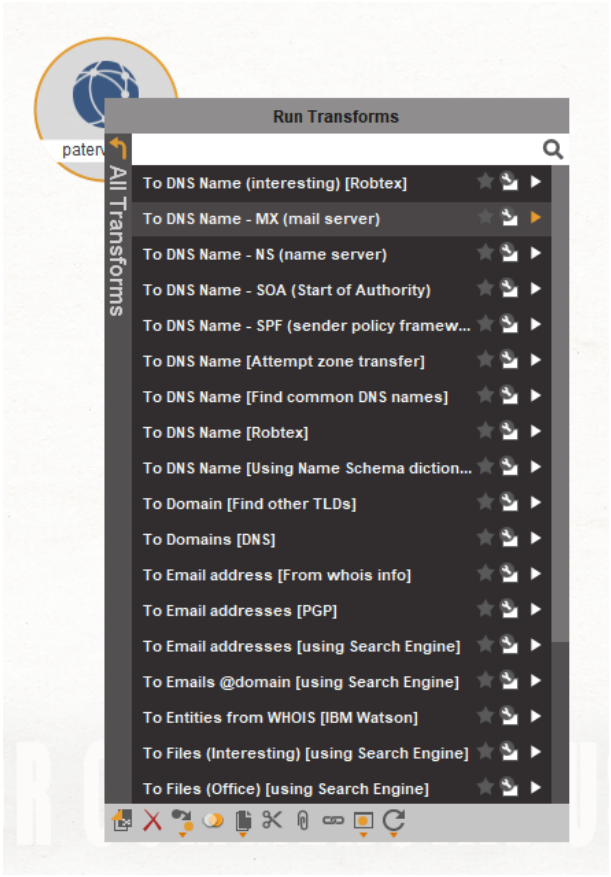


Figure 3.6: Selecting a transform from a node.

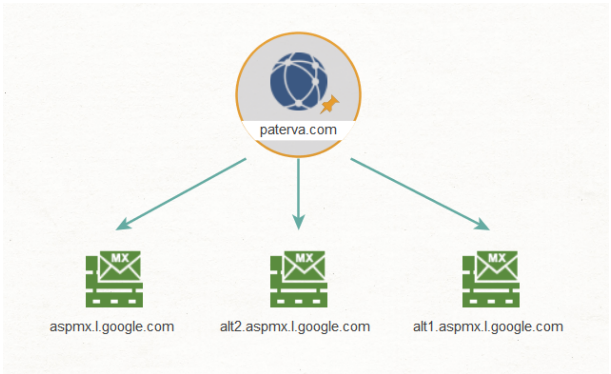


Figure 3.7: Result of running "To DNS Name - MX (mail server)".

```python
from maltego_trx.entities import Phrase
from maltego_trx.transform import DiscoverableTransform


class GreetPerson(DiscoverableTransform):
    """
    Returns a phrase greeting a person on the graph.
    """

    @classmethod
    def create_entities(cls, request, response):
        person_name = request.Value

        response.addEntity(Phrase, "Hi %s, nice to meet you!" % person_name)
```

Figure 3.8: Sample Transform. Source: `https://github.com/paterva/maltego-trx`

that obtains the DNS names from Robtex[1] database, another one extracts information of the domain from the SOA (Start of Authority) record, there is also one version that tries to guess common DNS names in the domain.

3.1.3 Maltego TRX

Maltego-trx is the Maltego transform library for python, installation instructions can be found at `https://github.com/paterva/maltego-trx`. It provides a simple API for creating custom transforms.

Adding a new transform

Adding a new transform is as simple as creating a new class in the "transforms" folder of the project directory. Any file in the folder where the class name matches the filename and the class inherits from "DiscoverableTransform", will automatically be discovered and added to the server. A simple transform would look like the one shown in Figure 3.8.

Importing a local transform in Maltego

You can add the local transform to Maltego by opening Maltego desktop client, clicking on the "Transforms" tab in the ribbon bar and clicking "New Local Transform" (see Figure 3.9) (Maltego, 2019e).

[1] `https://www.robtex.com/`

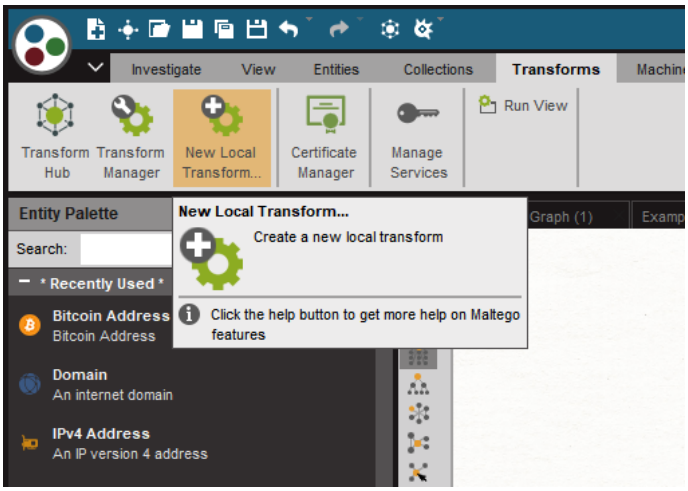


Figure 3.9: New local transform. Source: Maltego.

The first page (see Figure 3.10) allows us to describe the new transform. The most important details are the "Transform ID" (which must be unique) and Input Entity Type. The next page allows us to specify the settings used to execute the transform (see Figure 3.11). The "command" field should be the absolute path of the interpreter that will execute the transform.

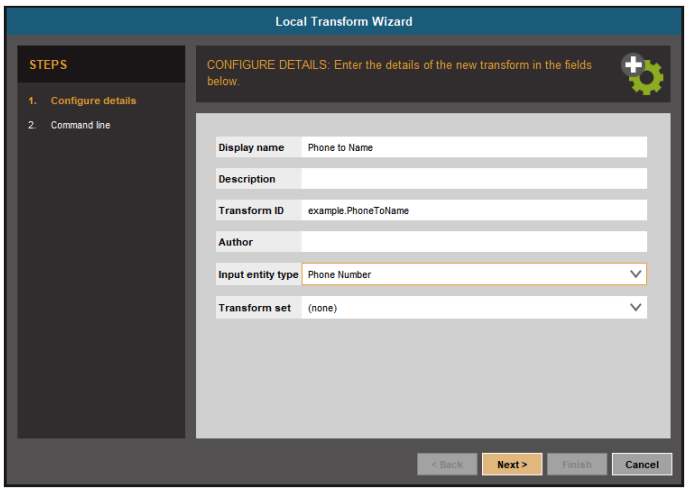


Figure 3.10: Local transform wizard. Source: Maltego.

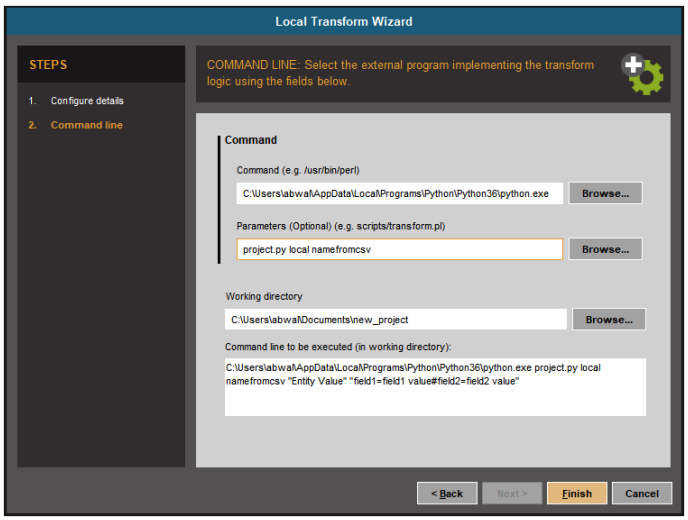


Figure 3.11: Transform configuration. Source: Maltego.

The "parameters" field will need to include "project.py" to tell Python to execute the project file. It also needs to include the parameter "local" to tell our project is being run locally, and the transform name to run. Finally, Figure 3.13 shows the new transform in action.

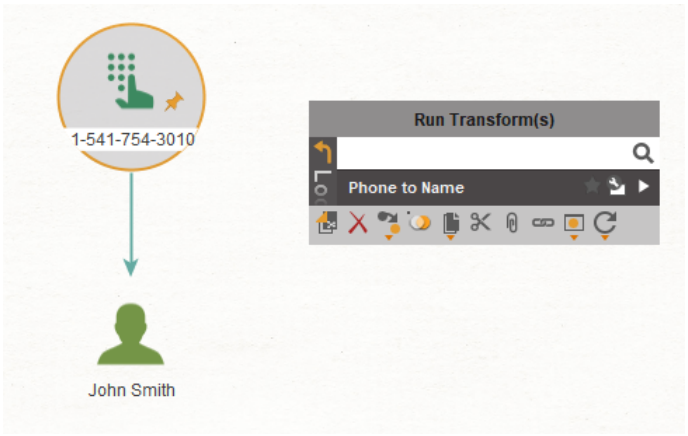


Figure 3.12: Run new local transform. Source: Maltego.

3.2 VirusTotal

This section defines the key concepts of VirusTotal that are within the scope of this project. As antivirus engines are the core of the features available at VirusTotal, this section starts with a description of them, followed by a detailed

exploration of the current VirusTotal APIs (i.e. v2 and v3), everything comes together in the VirusTotal Graph subsection, which is an important part of the project, after all, the mission is to implement its features in Maltego. The end of this section looks at the VirusTotal transform set already available in Maltego and compare it to what this project accomplishes.

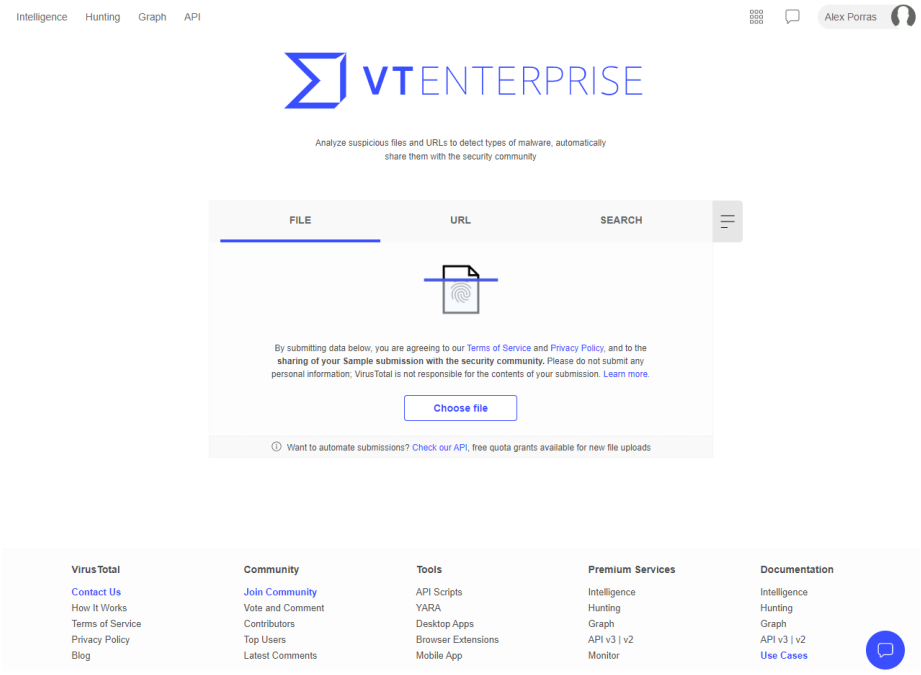


Figure 3.13: VirusTotal main page.

3.2.1 Antivirus engines

One of the most known features of VirusTotal is the analysis of files and URLs by its myriad of partnered antivirus engines. It currently has 72 engines. As an example, Figure 3.14 shows the results of searching for a malicious sample. Note that new malware variants may take a while for antivirus engines to identify, so there is not a hard threshold by which it can be said that it has “a lot of” or “few” detections.

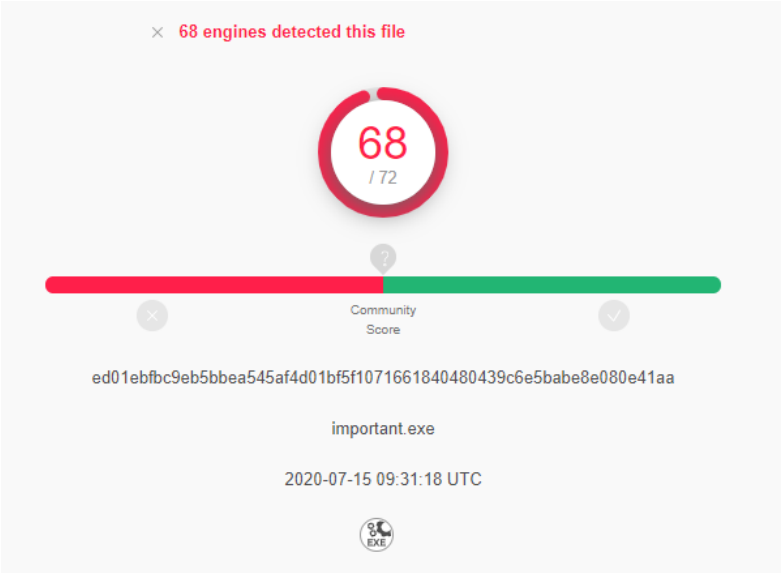


Figure 3.14: Example detection ratio of VirusTotal antivirus engines.

Antivirus engines are software designed and used to prevent, detect, and remove malware (Wikipedia contributors, 2020a). Antivirus software not only detects computer viruses, with the proliferation of other kinds of malware, but it also alerts us of, for example, browser hijackers, ransomware, keyloggers, spam, phishing, etc[2].

VirusTotal not only inspects the samples sent to it but it also catalogs, indexes, and saves them so others can download and search them. Making VirusTotal one of the biggest malware databases in the world.

3.2.2 VirusTotal Graph

VirusTotal Graph visually explores the VirusTotal dataset, understanding the relationship between files, URLs, domains, IP addresses, and other items encountered in an ongoing investigation (VirusTotal, 2020h). VirusTotal Graph is backed by the VirusTotal API version 3, which revolves around three concepts (VirusTotal, 2020b):

- Objects. Any item that can be retrieved or manipulated using the API (e.g. files, URLs, domain names).
- Collections. A set of objects, usually of the same type, except for a few cases.

[2]Explaining the different kinds of malicious activity is out of the scope of this project.

- Relationships. Links between objects, for example, a URL can be related to a file because the file was downloaded from the URL.

Nodes and relationships

Each node in the graph represents an entity. There are 5 basic entity types (VirusTotal, 2020f): Files, domains, URLs, IP addresses and relationship nodes (see Figure 3.15).

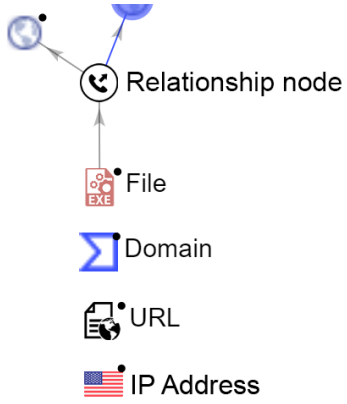


Figure 3.15: Basic entity types.

Relationships are the way in which the VirusTotal API expresses links or dependencies between objects (see Figure 3.16). Relationships can be one-to-one or one-to-many, depending on whether the object is related to a single object or to multiple objects (VirusTotal, 2020d).

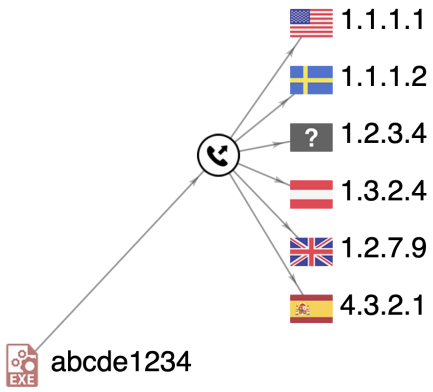


Figure 3.16: Hash abcde1234 connects with contacted IP addresses. (Source: VirusTotal).

Each entity has a set of relationships that can be explored, which can be separated into public and private, the private ones can only be accessed by users with a VirusTotal private API key. Depending on the access level of the account, VirusTotal Graph shows the available relationships of the selected node. Some relationships may output lots of nodes, to prevent this from cluttering the screen, VirusTotal Graph returns relationship outcomes in clusters of 20 nodes each time. VirusTotal Graph also allows users to share the graphs they have created publicly or within their organization.

3.2.3 VirusTotal API

The API is the way to access VirusTotal features in a programmatic way, allowing us to automate certain tasks or integrate with other platforms. The newest version of the API is "version 3". The VirusTotal API version 3 will be replacing the API version 2, which will be deprecated. The newer API was designed to be easier and more consistent, inspired by the `http://jsonapi.org/` specification (VirusTotal, 2020c).

The API lets users upload and scan files or URLs, access finished scan reports, and make automatic comments, as well as access the available objects and their characteristics. This project will not focus on uploading and scanning, but accessing what is already present.

The APIv2 is a more basic version, with a set of API endpoints that instead of separating the information into objects, collections and relationships, returns everything that is known about a resource (e.g. a file) into the same API response, for example, `/file/report` returns information about files.

HTTP	Description
204	Request rate limit exceeded. You are making more requests than allowed. You have exceeded one of your quotas (minute, daily or monthly). Daily quotas are reset every day at 00:00 UTC.
400	Bad request. Your request was somehow incorrect. This can be caused by missing arguments or arguments with wrong values.
403	Forbidden. You don't have enough privileges to make the request. You may be doing a request without providing an API key or you may be making a request to a Private API without having the appropriate privileges.

Table 3.1: APIv2 errors.

The APIv3 revolves around three key concepts: objects, collections and relationships (VirusTotal, 2020b). An object is any item that can be retrieved using the API, for example, files, URLs, and domain names are API objects. Objects can be grouped into collections, which are usually of the same type. Relationships are links between objects, for instance, a URL can be related to a file because the file was downloaded from the URL.

API responses

Both versions of the API return JSON objects as a response in most cases. The main difference here is that the newer version has a wider variety of error codes, thus making the responses more meaningful. Table 3.1 shows the error response of the APIv2 while Table 3.2 shows error responses of the newest version.

Endpoints

APIv3 has an extended set of endpoints for the main API objects (files, URLs, domains, and IP addresses) and most notoriously adds graphs as a new type of API object. The endpoints this project is most concerned with are the "relationship" endpoints, which are not available in the version 2 of the API[3]. These endpoints allow us to consult related objects more easily and directly.

Note: The APIv2 does not organize the components into objects, collections, and relationships, therefore the following are exclusive to the APIv3.

[3]See `https://developers.virustotal.com/reference`

Error code	HTTP	Description
AlreadyExistsError	409	The resource already exists.
AuthenticationRequiredError	401	The operation requires an authenticated user. Verify that you have provided your API key.
BadRequestError	400	The API request is invalid or malformed. The message usually provides details about why the request is not valid.
ForbiddenError	403	You are not allowed to perform the requested operation.
InvalidArgumentError	400	Some of the provided arguments is incorrect.
NotFoundError	404	The requested resource was not found.
QuotaExceededError	429	You have exceeded one of your quotas (minute, daily or monthly). Daily quotas are reset every day at 00:00 UTC.
TooManyRequestsError	429	Too many requests.
UserNotActiveError	401	The user account is not active. Make sure you properly activated your account by following the link sent to your email.
WrongCredentialsError	401	The provided API key is incorrect.
TransientError	503	Transient server error. Retry might work.

Table 3.2: APIv3 errors.

Objects

Each APIv3 object is uniquely identified by the type and id of the object. Each object has an associated URL with the following structure: `https://www.virustotal.com/api/v3/<collection_name>/<object_id>`

In addition to and id an type, objects have a set of attributes that are always present, even if empty, which can contain any type supported by JSON, including lists and dictionaries. Optionally, users can also include relationships into the object request, however, there are endpoints to ask for specific relationships of objects.

Collections

Collections are returned in a similar fashion to objects, but, in the data section, they have a list with the objects descriptors instead. Many collections are *iterable*, instead of getting the entire collection at once, it is received in chunks and explored using a cursor.

Relationships

When retrieving an object with a GET request, users may also want to retrieve the relationships it has with other objects, to do so, their request should be like

the following:

```
https://www.virustotal.com/api/v3/<collection_name>/
<object_id>?relationships=<relationship1>,<relationship
_2>
```

The response of such requests will include a relationships dictionary, keyed by their relationship name with values that may be an object descriptor if the relationship is one-to-one or a **collection** if the relationship is one-to-many.

one-to-many relationships, usually have their own URL that can be used to iterate over the related objects by sending GET requests like this:

```
https://www.virustotal.com/api/v3/<collection_name>/
      <object_id>/relationships/<relationship>
```

```
https://www.virustotal.com/api/v3/<collection_name>/
             <object_id>/<relationship>
```

The first one will only return the id and the type of the related objects, while the second one returns the entire object descriptors.

Chapter 4

VirusTotal Private Transforms

In this chapter, the design and development of the VirusTotal private transforms (VT private transforms) as well as a way of importing VirusTotal graphs into Maltego are explained. It is worthy to note that to accomplish the objectives of this master's thesis (i.e. Section 1.2), a total of 39 new transforms were implemented. These are "local" transforms, but, buying a Maltego server subscription would allow us to make them public.

Each section introduces the design of the solution followed by a subsection with the implementation. The design explains the ideas and the overall architecture of the solution. It includes various Unified Modeling Language (UML) diagrams, such as class and action diagrams. It also contains flow charts that explain the logic behind the more complex parts. This section is split in two subsections, one for the VirusTotal transforms and the other for the script that translates VirusTotal graphs into a format Maltego can read. All the code is written in Python. During this project, a VirusTotal API key with private access is used, meaning that the complete VirusTotal API functionality can be used.

4.1 Developing new transforms

As introduced in the background chapter, Maltego-TRX allows us to create new transforms. Transforms must extend from the class 'DiscoverableTransform'. The class diagram of the transforms is shown in Figure 4.1. To increase readability, the classes are named according to the entity they come from (i.e. the VirusTotal API object) followed by the relationship they request. This is because there can be more than one API object type with the same relationship name.

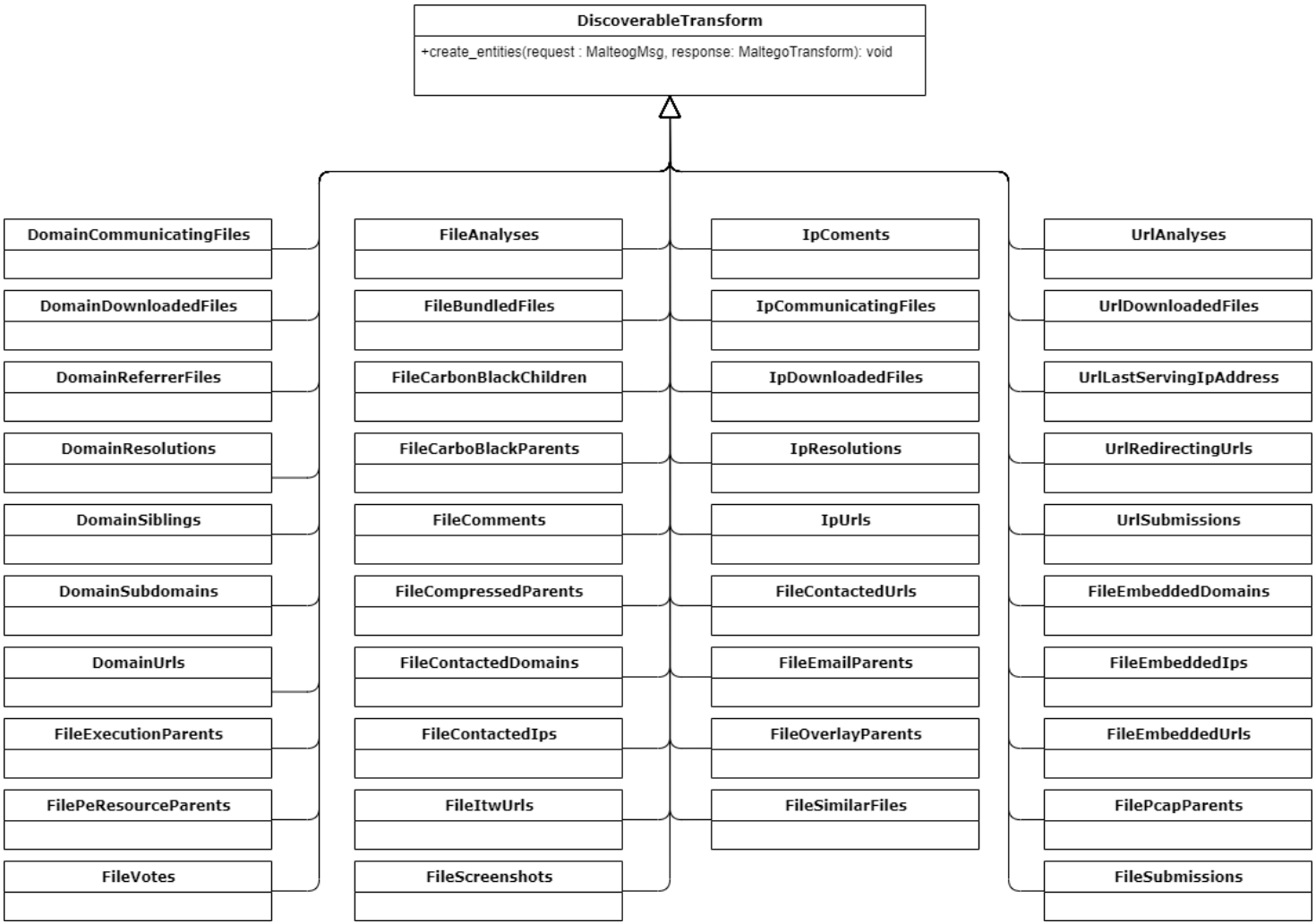


Figure 4.1: VirusTotal private transforms class diagram.

This project includes a total of 39 new transforms (see Figure 4.1). Each transform implements a single class method called 'create_entities()'. This method receives the entity it was called from in the request parameter, and outputs the transform entities in its response parameter, the method itself returns void. The request parameter is an instance of 'MaltegoMsg' and the response is an instance of 'MaltegoTransform'. MaltegoMsg contains a serialized version of the origin node, not a reference to the node itself. MaltegoTransform is a class holding the entities that are going to be returned by the transform, the entities are of type 'MaltegoEntity'. To make it easier for transforms that return the same type of node, the same type of nodes extend from MaltegoEntity, as seen in Figure 4.2.

The classes themselves are not created solely based on the Maltego entity they return, but by the concept that the output represents, this way, different concepts are not attached to entity types, for example, votes and comments return a 'Phrase' entity, but their concepts are different, so the characteristics of the VoteEntity (e.g. its icon) can be changed without altering the CommentsEntity. Besides, separating the entities in different independent classes makes it possible to modify them in the future more comfortably, for example, if the VirusTotal API changes.

The way the transforms operate is illustrated in Figure 4.3. First, an HTTP request to the VirusTotal API is made using the user's private API key. Then, the JSON response is read, if it is empty, no new nodes appear, if there has been an error, Maltego shows the content of the error in an alert window, otherwise, the transform defines the new nodes and adds them to the response to later spawn in Maltego.

The rest of the section deals with the implementation of the VirusTotal entities first, followed by descriptions of the different transforms that this project includes.

4.1.1 Entities

As explained before, VirusTotal nodes will be expressed as subclasses of MaltegoEntity. Each of them works independently and can be customized.

FileEntity

File nodes only need the hash of the file it represents as value, so to create the hash entity, it is enough with the VirusTotal file id (see Figure 4.4).

AnalysisEntity

In VirusTotal Graph, the file nodes show the analyses by changing the icon of the node, red ones represent files with at least one antivirus detection while

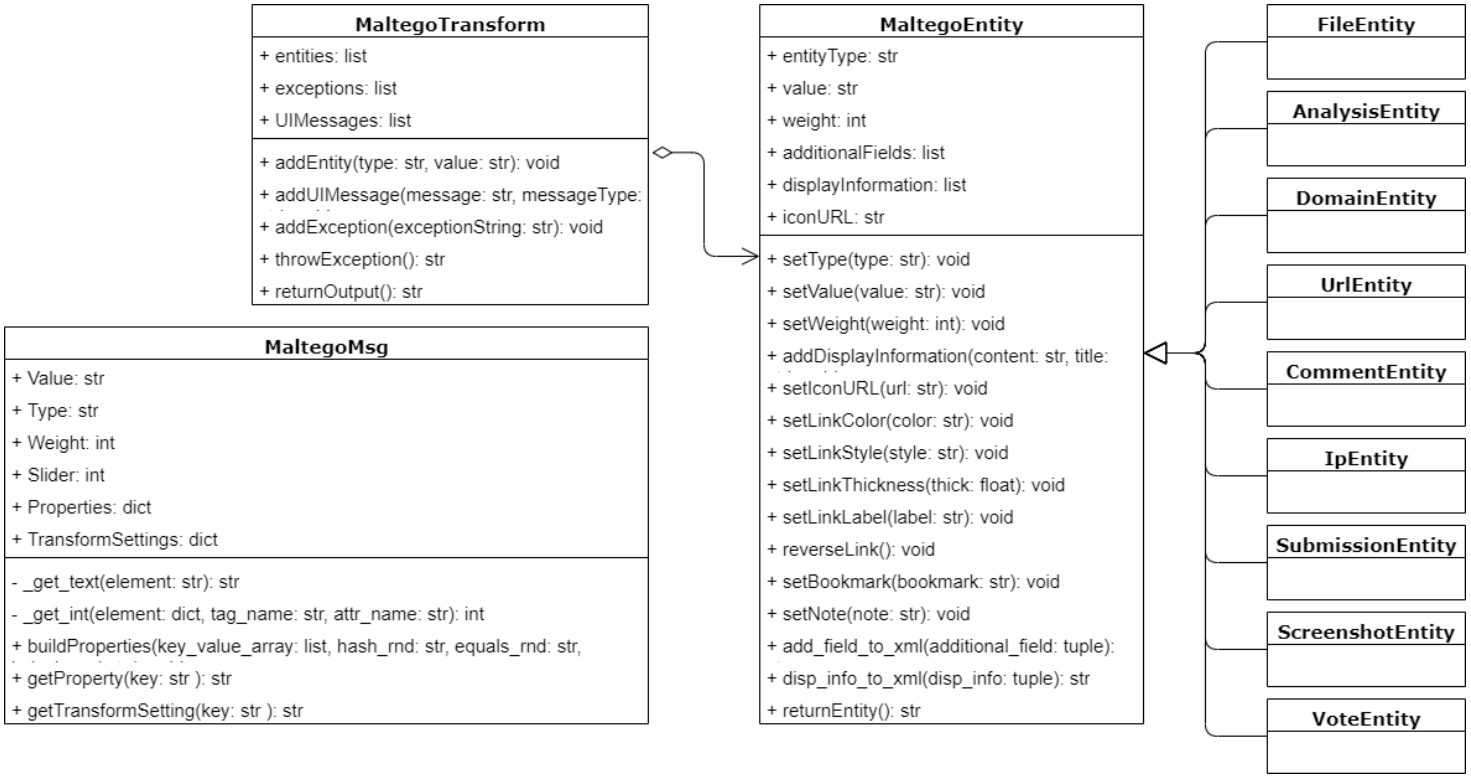


Figure 4.2: Maltego class diagram with the addition of VirusTotal entities.

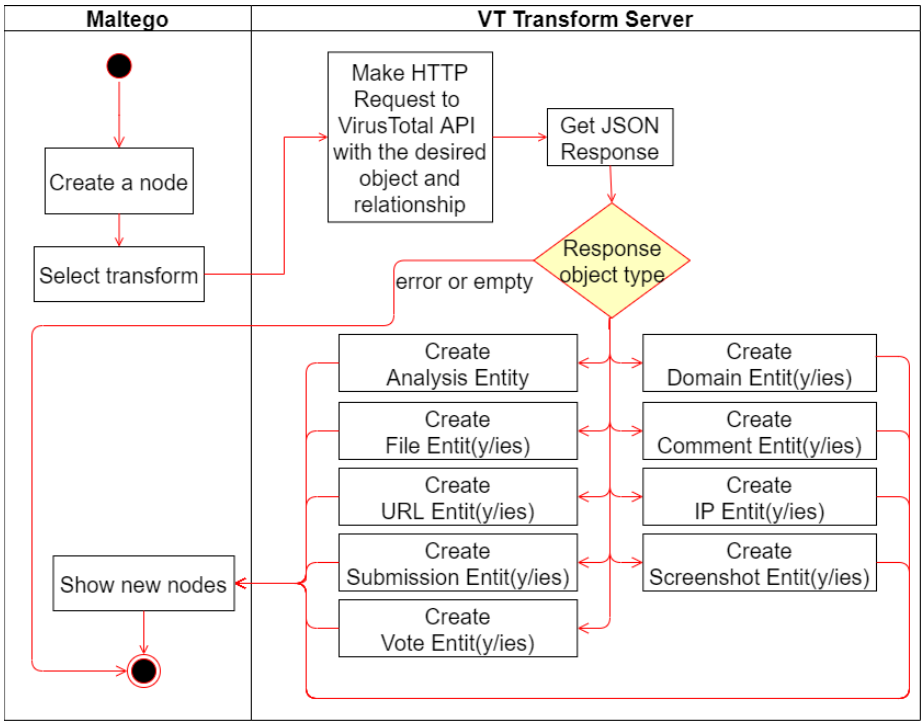


Figure 4.3: Transforms action diagram.

```
class FileEntity(maltego.MaltegoEntity):
    """
    Describes a VirusTotal file node in Maltego.
    """
    def __init__(self, vt_file=None):
        super().__init__('maltego.Hash', value=vt_file['id'])
```

Figure 4.4: File entity node implementation

```python
class AnalysisEntity(maltego.MaltegoEntity):
    """
    Describes the ratio of antivirus detections in VirusTotal
    as a node in Maltego.
    """
    def __init__(self, analysis=None):
        malware_count = (analysis['stats']['malicious'] +
        analysis['stats']['suspicious'])
        goodware_count = (analysis['stats']['harmless'] +
        analysis['stats']['undetected'])
        super().__init__('maltego.Banner',
            '{} / {}'.format(malware_count, malware_count + goodware_count))
        analysis_date = datetime \
            .utcfromtimestamp(analysis['date']).strftime('%Y-%m-%dT%H:%M:%SZ')
        self.addProperty(
            fieldName='date',
            displayName='Date',
            value=analysis_date
        )
        self.setLinkLabel(analysis_date)
        if malware_count == 0:
            self.setIconURL(icons.zero_analyses)
        else:
            self.setIconURL(icons.one_plus_analyses)

        for key in analysis['results']:
            self.addProperty(
                fieldName=key,
                displayName=key,
                value=analysis['results'][key]['result']
            )
```

Figure 4.5: Analysis entity implementation.

black ones show that there hasn't been any detection. To portray a similar behavior in Maltego, an analysis node was created. The analysis node shows the ratio between the number of antivirus detections and the total number of antivirus that gave an answer (see Figure 4.5).

Depending on whether the file had zero or more detections, different icons are shown (see Figure 4.6). The link to the analysis node has a label indicating the date in the UTC of the analysis. Lastly, all the individual antivirus verdicts are included as properties of the node.

DomainEntity

Domains are one of the main VirusTotal Graph node types. Like files, domain nodes only need the id of the domain (i.e. its name) to create the node (see

Figure 4.6: To the left, the icon that is showed when there is at least one positive. To the right, the icon that is showed when there are no positives.

Figure 4.7).

```
class DomainEntity(maltego.MaltegoEntity):
    """
    Describes a VirusTotal domain node in Maltego.
    """
    def __init__(self, domain=None):
        super().__init__('maltego.Domain', value=domain['id'])
```

Figure 4.7: Domain entity implementation.

UrlEntity

VirusTotal Graph shows the URL and whether it is malicious or harmless in the same icon. In Maltego, the URL will be separate from the analysis results. On top of adding the URL as the main value of the node, it is also included as the URL property of the node, this is in case someone wants to use an alias for a shorter title in the node (see Figure 4.8).

```
class UrlEntity(maltego.MaltegoEntity):
    """
    Describes a VirusTotal URL node in Maltego.
    """
    def __init__(self, url=None):
        super().__init__('maltego.URL', url['attributes']['url'])
        self.addProperty(
            fieldName='url',
            displayName='URL',
            value=url['attributes']['url'])
```

Figure 4.8: URL entity implementation.

CommentEntity

It is possible to post comments in VirusTotal for certain API objects. In Maltego, they are going to be phrase nodes. The value of the entity is the comment itself. Note that the name of the user that posted the comment is not disclosed.

```
class CommentEntity(maltego.MaltegoEntity):
    """
    Describes a VirusTotal comment as a node in Maltego.
    """
    def __init__(self, comment=None):
        super().__init__('maltego.Phrase', comment['attributes']['text'])
```

Figure 4.9: Comment entity implementation.

IpEntity

VirusTotal graph shows IP address nodes in conjunction with the country that hosts the IP address. However, Maltego only shows the IP address of the node, although it has a transform that returns the country of the IP address. The last analysis results are also included as part of the properties of the node.

```
class IpEntity(maltego.MaltegoEntity):
    """
    Describes a VirusTotal IP Address node in Maltego.
    """
    def __init__(self, ip_address=None):
        super().__init__('maltego.IPv4Address', ip_address['id'])
        self.addProperty(
            fieldName='last_analysis_stats',
            displayName='last_analysis_stats',
            value=str(ip_address['attributes']['last_analysis_stats']))
```

Figure 4.10: IP entity implementation.

SubmissionEntity

Submissions indicate the origin of the API objects that are uploaded to VirusTotal. This project includes them in Maltego as location entities, the value of the node is the city followed by the full name of the country, that is retrieved from the Alpha2 country code since VirusTotal expresses the country in that format. The submission time in the UTC is included in the properties of the node and the label of the link (see Figure 4.11).

```python
class SubmissionEntitity(maltego.MaltegoEntity):
    """
    Describes a VirusTotal Submission (Location) node in Maltego.
    """
    def __init__(self, submission=None):
        country_code = submission['attributes']['country']
        country_name = pycountry.countries.get(alpha_2=country_code).name
        city = submission['attributes']['city']
        super().__init__(
            'maltego.Location','{}, {}'.format(city, country_name))
        self.addProperty(
            fieldName='city',displayName='City', value=city
        )
        self.addProperty(
            fieldName='country', displayName='Country', value=country_name
        )
        self.addProperty(
            fieldName='countrycode',
            displayName='Country Code',
            value=country_code
        )

        submission_date = datetime \
                .utcfromtimestamp(submission['attributes']['date']) \
                    .strftime('%Y-%m-%dT%H:%M:%SZ')
        self.addProperty(
            fieldName='date', displayName='Date', value=submission_date
        )
        self.setLinkLabel(submission_date)
```

Figure 4.11: Submission entity implementation.

ScreenshotEntity

VirusTotal sandboxes often take screenshots of the virtual machines to visually show the behavior of the samples. This is shown in Maltego as image nodes by setting the URL property to that of the screenshot. The screenshot ID is used as the value of the entity (see Figure 4.12).

```python
class ScreenshotEntity(maltego.MaltegoEntity):
    """
    Describes a VirusTotal sandbox screenshot as a node in Maltego.
    """
    def __init__(self, screenshot=None):
        super().__init__(screenshot['id'])
        self.addProperty(
                fieldName='url',
                displayName='URL',
                value=screenshot['attributes']['link']
            )
        self.addProperty(
                fieldName='sandbox_name',
                displayName='Sandbox name',
                value=screenshot['attributes']['sandbox_name']
            )
```

Figure 4.12: Screenshot entity implementation.

VoteEntity

Votes in the VirusTotal community are used to show whether they think an API object is malicious or harmless. For the lack of a better node, Phrase nodes are used to represent the votes in Maltego. The value of the node is a composition of the verdict, the value, and the id of the vote (see Figure 4.14). Depending on whether the vote verdict is "harmless" or "malicious" different icons are shown (see Figure 4.13).

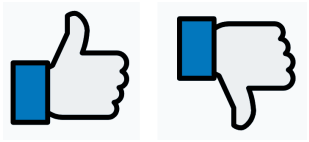


Figure 4.13: Vote icons: "malicious" to the right and "harmless" to the left. Courtesy of "Pixel Perfect" at `flaticon.com`.

4.1.2 File transforms

This section shows all the transforms that take a file as input. These transforms come from APIv3 file relationships. For example, the hashes in Table 4.1 can be used to try the transforms.

```
class VoteEntity(maltego.MaltegoEntity):
    """
    Descibes a VirusTotal vote as a node in Maltego.
    """
    def __init__(self, vote=None):
        super().__init__(
                'maltego.Phrase',
                '{}, {}, {}'.format(
                    vote['attributes']['verdict'],
                    vote['attributes']['value'],
                    vote['id']
                )
            )
        if vote['attributes'].get('verdict', 'harmless') == 'harmless':
            self.setIconURL(
                [redacted]
            )
        else:
            self.setIconURL(
                [redacted]
            )
        vote_date = datetime \
            .utcfromtimestamp(vote['attributes']['date']) \
            .strftime('%Y-%m-%dT%H:%M:%SZ')
        self.addProperty(
            fieldName='date', displayName='Date', value=vote_date
        )
        self.setLinkLabel(vote_date)
```

Figure 4.14: Vote entity implementation.

Table 4.1: Sample hashes.

Sample Name	**SHA1**
WannaCry	5ff465afaabcbf0150d1a3ab2c2e74f3a4426467
KMSpico_v10.2.0.zip	902bbdcea6feb2875b9302f32720e4de4fef60bb
Suspcicios DNS	5d36cdcf827d08fe8975db62cf536a9b3cdb965b

Analyses

This relationship returns the antivirus analyses of a given file in the form of an AnalysisEntity. Only the last analysis is shown, note that this does not analyze the file again, but rather takes the historical analyses and returns the last one. Figure 4.15 shows the detections of the "Wannacry" sample.

While investigating malware, it is beneficial to know the entities that are associated with an infectious file. For example, we could relate a person or an organization with a malicious file, thus, suspecting files that the person/organization releases in the future.

Figure 4.15: New transform showing the antivirus detection ratio of WannaCry.

Bundled files

Sometimes, files consist of packages containing other files, such as zip or RAR files, for example. This relationship outputs all the files that are packaged inside the input. You can try this using the file called KMSpico_v10.2.0.zip (see Table 4.1). This transform can be used in situations where, for instance, a bunch of packaged files contain malware, providing a nice lead on the rest of the files.

Carbon black children/parents

These transforms deal with files derived from other files according to "carbon black" they return instances of FileEntity. "Carbon black" refers to VMWare Carbon Black, a company that offers endpoint security services (VMWare Carbon Black, 2020). Carbon Black acts like a surveillance camera for end-user PCs, recording downloaded files, spawned processes, files written to disk, etc. These relationships store the information of files written to disk by the sample under consideration (VirusTotal, 2020d).

Comments

This relationship returns comments (i.e. instances of CommentEntity) made by VirusTotal users regarding the selected file (see Figure 4.17).

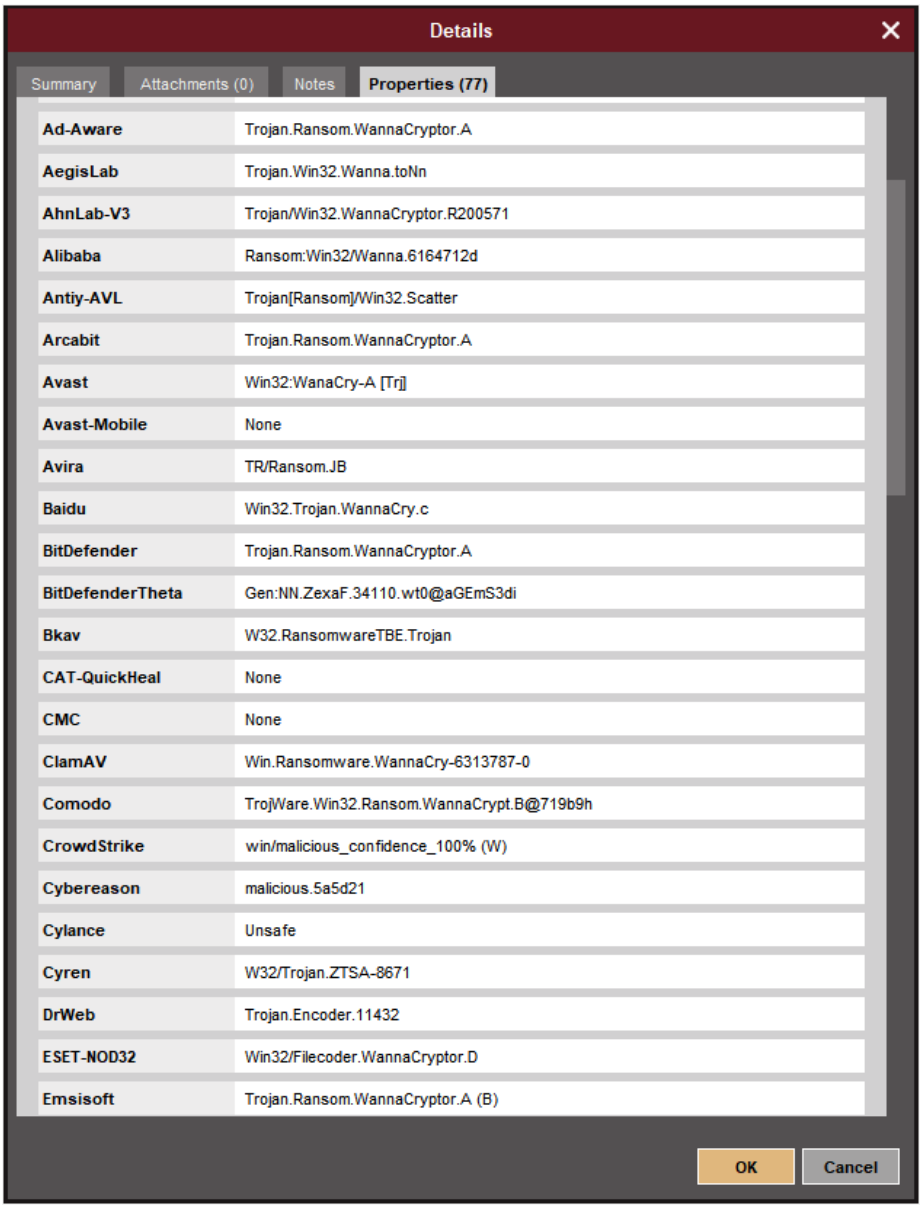


Figure 4.16: Example of properties of an Analysis entity (WannaCry verdicts).

Figure 4.17: New transform showing comments of the WannaCry sample.

Compressed parents

This is the opposite of the bundled files transform. It returns compressed files (i.e. instances of FileEntity) that contained the selected file (see Figure 4.18). A single malicious sample may be bundled in different files that have different origins. This may be due to a malware sample that is becoming popular among malicious agents.

Contacted domains/IPs/URLs

These transforms return the internet domains, IP addresses, or URLs contacted by the input file as instances of DomainEntity, IpEntity, and UrlEntity respectively. For demonstration purposes, a file that seems to contain a suspicious DNS was picked (see Table 4.1). The result of the transform can be seen in Figure 4.19. It doesn't only help identify domains, IP addresses, or URLs that may be hosting malicious samples, but also a way to find similarities between the contacted domains for different samples.

Email parents

This transform returns all email files containing the input file as instances of FileEntity. There plenty of malware campaigns that infect organizations and users by sending emails with infected files attached to them. This transform may help identify malware samples that were used by those campaigns.

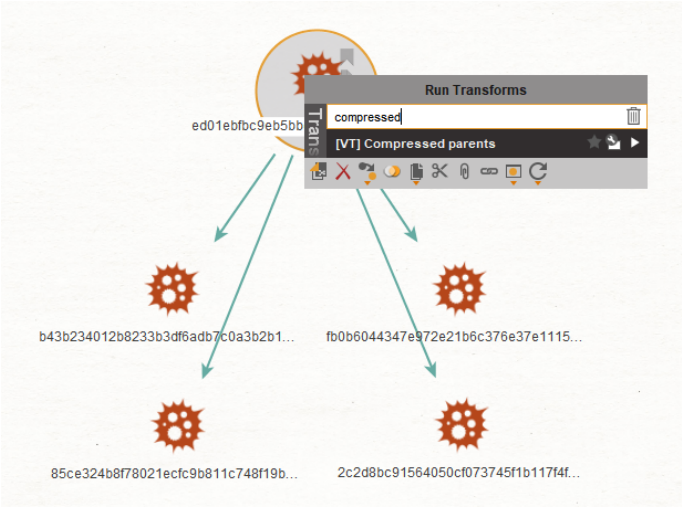


Figure 4.18: New transform showing compressed files that contained the WannaCry sample.

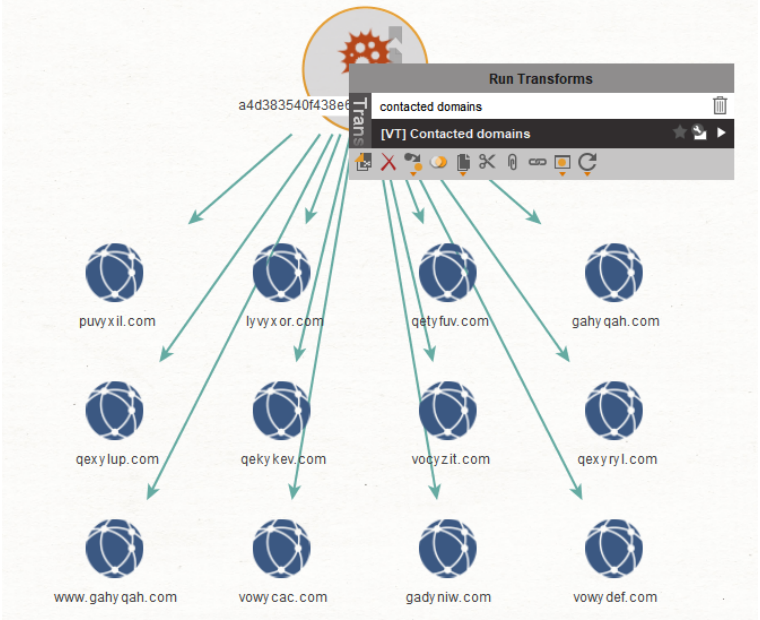


Figure 4.19: New transform showing contacted domains of the sample containing the suspicious DNS.

Embedded domains/IPs/URLs

While doing malware static analysis, one of the things to look for are strings embedded within the file. It is not necessary to execute the sample to know these strings and they provide strong indicators of compromise (IoC). IoCs are forensic artifacts that are used as signs that a system has been compromised by an attack or that it has been infected with a particular malicious software (Catakoglu, Balduzzi, & Balzarotti, 2016). These transforms show domain names, IP addresses, and URLs that are embedded within the selected file.

In the wild URLs

This transform returns URLs from which the file was downloaded as instances of UrlEntity. This could be useful to find out more about the returned URLs. Maybe those URLs have common domains or are hosted by the same IP address, for example.

Overlay parents

This transform returns files that contain the selected file as an overlay. Overlays are self-contained blocks of code that are part of a bigger program that does not fit in memory (Wikipedia contributors, 2020c).

PCAP parents

This transform outputs Packet Capture (PCAP) files that contain the input file as instances of FileEntity. PCAP files are produced by network monitoring software (e.g. WireShark). If this transform produces any output, it means that the input file was captured in network traffic.

Execution parents

One of the tactics to hide malware from antivirus engines is to send an executable file that is not malicious in itself, but it is used as a proxy to install and execute other files. This transform returns files that executed the input file.

PE resource parents

"Portable Executable" (PE) refers to the fact that the format is not architecture-specific (Microsoft, 2020). PE files index resources in a table within the file. This transform shows PE files containing the selected file as a resource.

Screenshots

It was previously mentioned that VirusTotal uses sandboxes to analyse the behavior of the samples, some of these sandboxes take screenshots. This transform links screenshots related to the sandbox execution of the input file as instances of ScreenshotEntity (see Figure 4.20).

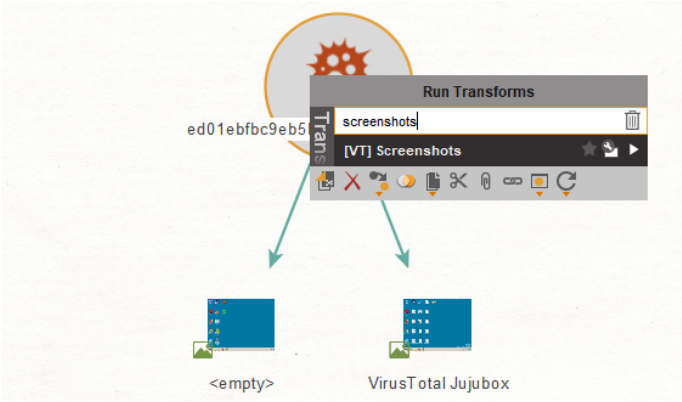


Figure 4.20: New transform showing WannaCry sandbox screenshots.

Similar files

This transform links files that are similar to the selected file. The association is made by using VirusTotal's in-house "vhash", a type of fuzzy hash. Fuzzy hashes hold a certain tolerance for changes and can tell how different two files are by comparing the similarity of their outputs (Mimecast, 2019). Knowing about similar files can help identify and cluster files by malware family.

Submissions

If we know the places a sample has been submitted from, we can get an idea of how spread that sample is, plus, if we track the time of the submission, it is possible to create a map of the trajectory the sample followed since it was first discovered. This transform returns the locations and date from which the file has been submitted as instances of SubmissionEntity (see Figure 4.21).

Votes

This transform shows votes for the selected file made by the VirusTotal community (see Figure 4.22). Some malware analysts follow the opinions of other malware analysts, this voting system is a good way of knowing what others think.

4.1.3 Domain transforms

The background chapter covered the basics of domain names and their characteristics. This section is about the transforms that are built from the relationships coming from domain API objects in VirusTotal.

Figure 4.21: New transform showing WannaCry submissions to VirusTotal.

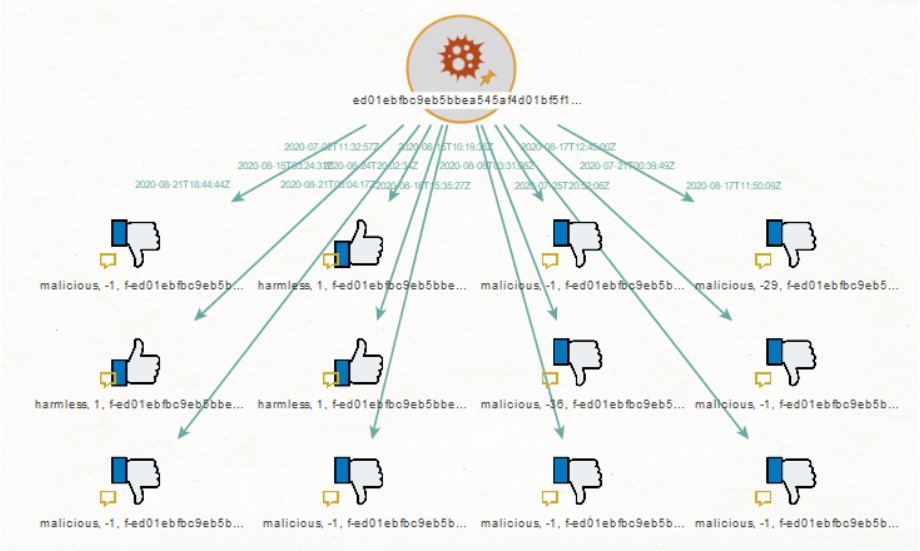


Figure 4.22: New transform showing WannaCry votes in VirusTotal.

Communicating files

This transform returns files that communicate with the selected internet domain. This is the opposite side of the transform called "Contacted domains". If we find a suspicious domain, it is plausible to suspect all the files that contact this domain.

Downloaded files

This transform returns files that were downloaded from the selected internet domain. Note that domain names don't refer to websites, but a place on the internet with a specific alias. There can be lots resources under the same domain name depending on how big the organization behind it is, for example, `dropbox .com` is a domain name, but the organization offers file hosting to billions of users, each user has its own set of files that are independent of other users' files.

Referrer files

This transform is the opposite end of the transform called "Embedded domains", it outputs files that contain the selected internet domain. If a domain name is known to be an indicator of compromise, this transform can be used to get samples that have that domain embedded.

Resolutions

This transform obtains DNS resolutions of the selected domain and returns the result as IP addresses (see Figure 4.23). As discussed in the background chapter, domain names aren't the ones that translate to IP addresses, DNS names are. Of course, there are plenty of DNS names that are equal to domain names, but they are not the same thing. Using this transform, we can find all the IP addresses under the authority of the input domain name.

Siblings and subdomains

These transforms show different levels in the hierarchy tree of the given domain name. "Siblings" shows domains at the same level while "subdomains" shows domains at lower levels. Note that all subdomains are under the authority of the input domain, but domain siblings aren't.

URLs

From the background chapter, we know that URLs are used to point at a specific asset inside a computer. A single DNS name can host numerous services depending on the port, but also, there can be any number of paths inside a single combination of DNS name and port, resulting in boundless URLs, which are controlled by the author of the service. This transform links URLs hosted by services under the authority of the selected internet domain, they are shown in Maltego as instances of UrlEntity (see Figure 4.24).

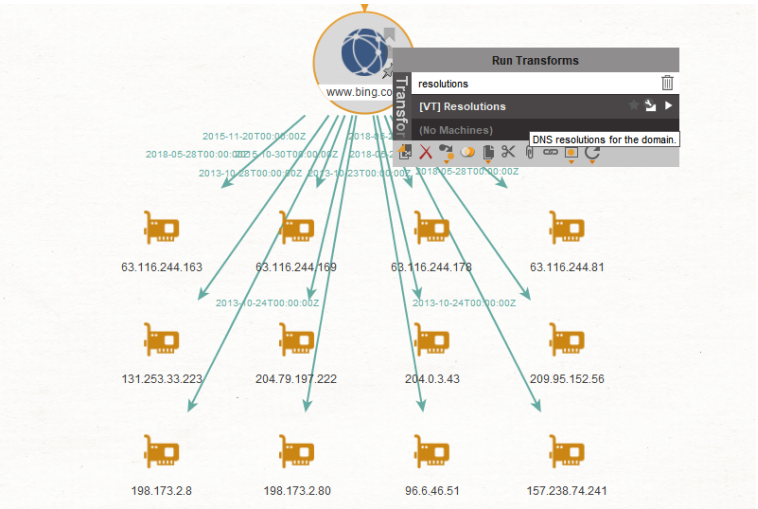


Figure 4.23: New transform showing `www.bing.com` DNS resolutions.

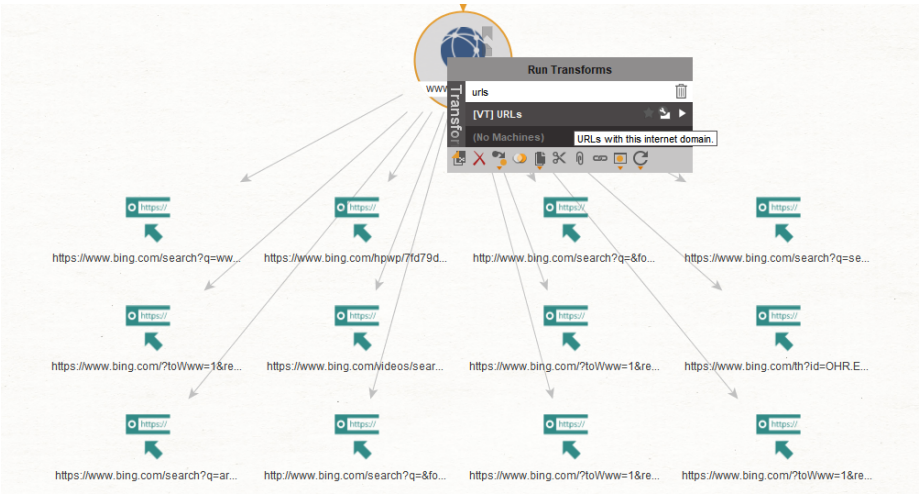


Figure 4.24: New transform showing URLs with `www.bing.com` as internet domain.

4.1.4 IP address transforms

This section covers all the transforms coming from IP address API objects in VirusTotal. Previous chapters have already discussed the different types of IP addresses. This section deals with IPv4. Since IPv4 are so limited in number, not every single person can have a different IP address. That is why Network Address Translation (NAT) is used. NAT allows us to use a single IPv4 address to serve a group of internet users. Sometimes, a single IPv4 address is used in a large area, this means that packages sent from the same IP address can have very different sources.

Comments

This transform shows comments (instances of CommentEntity) by the VirusTotal community regarding the input IP address. There may be some insights coming from what other malware analysts have to say.

Communicating files

This transform is the opposite end of the file transform called "Contacted IPs", it shows files that communicate with the selected IP address. Similar to previous transforms, if we find a suspicious IP address, we can suspect files that communicate with the said IP address.

Downloaded files

This transform shows files that were downloaded from the selected IP address. Although it is possible to directly receive files from an IP address, it is most common to download these files from services hosted with this IP address, for example, an FTP server.

Resolutions

Given an IP address, this transform shows all the domain names it is associated with. The label shows the date of the DNS resolution. Technically speaking, IP addresses resolve to DNS names, which are under the authority of the domain name.

URLs

This transform shows URLs related to the selected IP address. URLs can include the IP address instead of a DNS name. Running a website at "http://localhost:80/" is the same as running it at "http://127.0.0.1:80/". The same IP address can appear in any number of URLs.

4.1.5 URL transforms

This section covers transforms coming from VirusTotal API objects of type "url". Because URLs can be in any shape and size, the ID of URL objects is its SHA256 digest. Unlike the rest of the nodes, before asking VirusTotal about URL relationships, its hash needs to be calculated.

Analyses

VirusTotal also has partnered with antivirus engines that check URLs. This transform shows the last analysis made by those antivirus engines as an instance of AnalysisEntity because the behaviour is the same as the file analyses.

Downloaded files

This transform shows files that were downloaded from the selected URL. Unlike the previous cases, where the files were indirectly downloaded from the input node. URLs can point directly to files of any kind, making the relationship direct.

Last serving IP address

The IP addresses serving URLs may change over time. This transform links the last serving IP address of the selected URL. The same IP address may even change countries, so focusing on a single IP address in cases where it has historically been found suspicious may not be a good idea in the long run.

Redirecting URLs

Users make typos in the URLs, the most common usually are anticipated by the host and are redirected to the correct one, but, there are cases where a typo in an URL redirects to a completely different one on purpose guided by malicious agents. A chain of redirection that tries to avoid antivirus detection is also possible. This transform shows URLs that redirect to the selected URL.

Submissions

This transform shows the date and location of the submissions to VirusTotal of the selected URL as instances of SubmissionEntity. We need to be careful though, because it is possible to manipulate the origin of the submission by, for instance, using a VPN.

4.2 Importing VirusTotal graphs into Maltego

While VirusTotal graphs provide valuable insights, it is bound to the VirusTotal knowledge base. Maltego is able to exploit any source of data, private and public. Being able to transfer graphs from VirusTotal to Maltego with minimum

effort enriches investigations by building on top of the VirusTotal graph.

Ideally, there would a button in the import menu in Maltego that gets VirusTotal graphs, but, that would require changing the code of Maltego, which is not possible in the meantime. The import "tool" is a python script that operates with the format Maltego and VirusTotal serializes the graphs (i.e. VirusTotal graphs as JSON objects and Maltego graphs as CSV tables). Basically, it takes the VirusTotal graph and puts it in a CSV that Maltego can read.

Most of the node types available in VirusTotal are also present in Maltego, there are only a few cases in which they are not (e.g. wallet). The main difference between VirusTotal graphs and Maltego graphs are relationship nodes. These do not exist in Maltego, so this project removes the relationship nodes and puts the relationship name as a link label. Although this is not as useful as having relationship nodes, it will help keep a more organized graph.

Maltego CSV graphs have two sections: the nodes and the links. The file starts with all the different nodes in the graph, each in its own "paragraph". Each node "paragraph" contains the ID and the properties of the node. "Paragraphs" because each node is its own CSV (i.e. the columns do not take all the file) and they are separated by blank lines (see Figure 4.25).

```
maltego.Domain,Domain Name,WHOIS Info
EntityID,fqdn,whois-info
2r6nmrn3n9hvz,paterva.com,

maltego.Hash,Hash,Hash Type,Owner,Before,After,Included Media Types,Excluded Media Types
EntityID,properties.hash,type,owner,before,after,includeMediaType,excludeMediaType
2r6nmrn3n9hvx,af30fca836142d6a0b8672f1e8f53acf," ",,,,,

maltego.IPv4Address,IP Address,Internal,owner,Before,After,Include Media Type,Exclude Media Type
EntityID,ipv4-address,ipaddress.internal,owner,Before,After,includeMediaType,excludeMediaType
2r6nmrn3n9hvy,74.207.243.85,false,,,,,
```

Figure 4.25: CSV node section containing a domain, a hash and an IP address.

The links are written at the end of the file. They relate two entities by their IDs. Each link also has its own ID and the link properties, such as the label, the thickness of the line, etc (see Figure 4.26). Figure 4.27 shows the flowchart of writing the CSV file from VirusTotal JSON graphs.

```
Maltego Link,Source Entity ID,Target Entity ID,Label,Show Label,Color,Style,Thickness,Description
LinkID,SourceEntityID,TargetEntityID,maltego.link.manual.type,maltego.link.show-label,maltego.link.color,m
2r6nmrn3n9hwc,2r6nmrn3n9hvz,2r6nmrn3n9hw0,itw_domains,1,,-1,-1,
```

Figure 4.26: Example of a Maltego CSV link, cut to fit in the page.

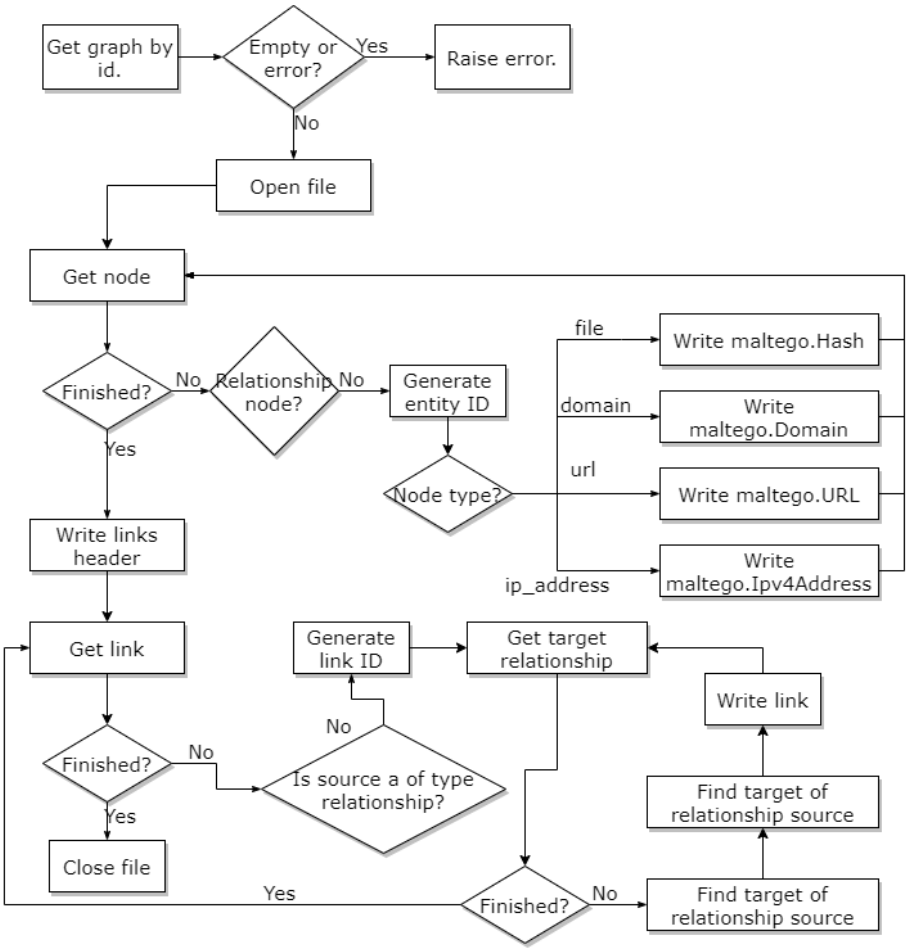


Figure 4.27: VirusTotal Graph to Maltego CSV flow chart.

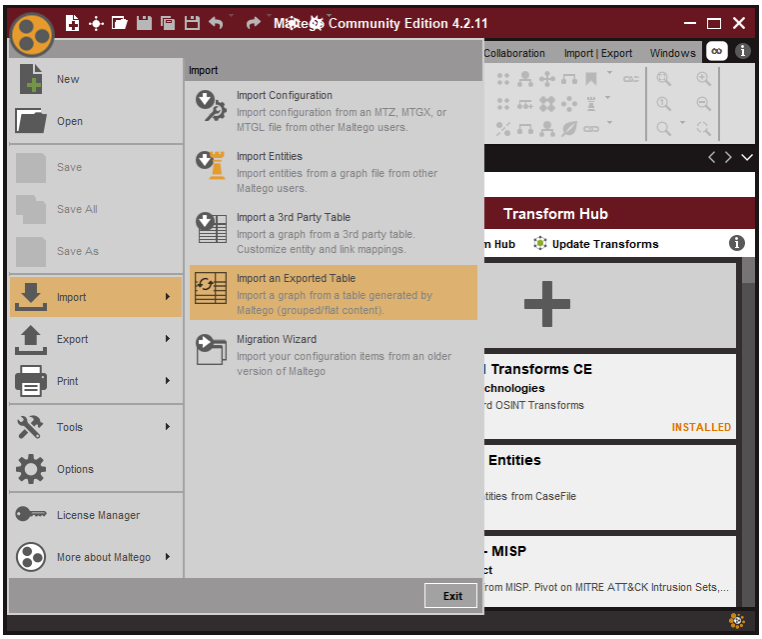


Figure 4.28: Selecting to import a graph from a CSV file.

4.2.1 How to use the import script

The import script is a single python executable that receives two parameters. It is required to have Python 3 installed to run it, as well as an OS environment variable called "VIRUSTOTAL_API_KEY" with a private API key as value. Once it is done, the script can be run like this

python vt_to_maltego.py <graph_id> <output_csv_file>

where "graph_id" is the ID of the graph to import, and "output_csv_file" is the resulting CSV file that Maltego can read. Note that the script can only be used with public graphs.

After successfully writing the CSV file, it needs to be imported in Maltego. To do so, select "Import" in the main menu (i.e. the one displayed clicking the Maltego icon) and select "Import from Exported" table (see Figure 4.28). The last step is to find the CSV file the script generated. This works because the CSV file is written the same way Maltego does, so it is not considered a 3rd party table. This way, it is not necessary to specify the format in the import wizard.

(This page has been intentionally left blank)

Chapter 5

Conclusions and future work

This chapter summarizes the results obtained with this project and goes through an overview of the whole process. It starts by remembering the available VirusTotal transforms in Maltego, and the difference we have made by including the private API relationships in conjunction with the ability to import VirusTotal graphs. The chapter ends with suggestions for improvements for future iterations.

5.1 Conclusion

Chapter 1 introduced the concept of OSINT and its relevance today, as well as the objectives for this master's thesis. In order to fully understand the potential of this work, Chapter 2 showed the current transforms that were available in Maltego as of the date of this project. They were either limited in the information the obtained because they only used the public API or limited by the interpretation of the reports because they used the APIv2. This project overcame those constraints by including private API relationships and using the APIv3. Chapter 3 described in detail the use of Maltego and VirusTotal Graph, covering Objectives **O1** and **O2**. Later, Chapter 4 went through the process of creating and adding custom transforms in Maltego. In total, this project includes 39 new transforms. These transforms match the relationships available at VirusTotal Graph, fulfilling Objective **O3**. Being compatible with VirusTotal Graph means that it is possible to reproduce its graphs in Maltego. On top of that, this project also has made it possible to import complete VirusTotal graphs into Maltego simply by executing a script that converts VirusTotal graphs into CSV files that Maltego can import (see Objective **O4**), something that would have required to be copied by hand. The code of the transforms, entities, and import script will be available at a public Github repository, where anyone can download it and test it.

In conclusion, the integration of VirusTotal with Maltego that this project

achieves goes a step further to what was available before. It fulfills the objectives proposed and enriches malware investigations that are carried on using Maltego by suitably introducing the information available at VirusTotal. As a personal reflection, working on this master's thesis has helped me deepen my understanding of malware investigations and its components.

5.1.1 Conclusión en español

En el Capítulo 1 se introdujo el concepto de OSINT y su relvancia hoy en día, así como los objectivos de este TFM. Para poder entender el potencial de este trabajo, en el Capitulo 2 se mostró las transformadas que están actualmente disponibles en Maltego hasta la fecha de este trabajo. Estas transformadas estaban limitadas porque sólo tenían acceso a la API pública o por la interpretación de los reportes de la APIv2. Este trabajo sobrepasó estas limitaciones incluyendo las relaciones privadas de la APIv3. En el Capítulo 3, se descibió en detalle el uso de Maltego y VirusTotal Graph, cubriendo los objetivos **O1** y **O2**. A continuación, en el Capítulo 4 se ha navegado sobre el proceso de creación de nuevas transformadas y cómo añadirlas a Maltego. En total, se han añadido 39 transformadas nuevas. Éstas concuerdan con las relaciones que ofrece VirusTotal Graph, cumpliendo el Objetivo **O3**. Ser compatible con VirusTotal Graph significa que es posbile reproducir sus grafos en Maltego. Sobre esto, este trabajo ha hecho posible la importación completa de los grafos de VirusTotal en Maltego simplemente ejecutado un programa que convierte los grafos de VirusTotal en archivos CSV que Maltego puede importar (ver Objetivo **O4**), algo que se hubiera requerido hacer a mano. El código de las transformadas, entidades e importación de grafos estará disponible en un repositorio público de Github, donde cualquiera podrá descargarlo y probarlo.

En conclusión, la integración de VirusTotal con Maltego que se realiza en este trabajo va un paso adelante de lo que había disponible. Se cumplen los objetivos propuestos y enriquece las investigaciones de malware que se realizan con Maltego gracias a la incorporación de la información disponible en VirusTotal. Como reflexión personal, realizar este trabajo fin de máster me ha ayudado a profundizar mi entendimiento de las investigaciones malware y sus componentes.

5.2 Future work

This section deals with possible additions to the project to be made in future iterations. Two of the most notorious include publishing the transforms and adding the possibility to import VirusTotal graphs with custom nodes in Maltego.

The current version of the script that allows us to import VirusTotal graphs in Maltego does not support the use of custom nodes, for example, actor, victim, wallet, SSL cert, etc. In a future iteration, it would be nice to include all these

types of nodes.

One major improvement would be to publish the transforms in a Maltego public server so that anyone can access the transforms from the transform hub. Maltego offers paid subscriptions to different kinds of servers (Maltego, 2020a). More specifically, this project fits to be used with an iTDS. The iTDS is Maltego's internal Transform Distribution Server. It allows custom Transforms to be developed, managed, shared, and distributed from a central point within an organization.

VirusTotal graph includes a feature to find common patterns in a selection of nodes of the graph (VirusTotal, 2020e). If in the future it becomes possible to access Maltego code we could take advantage of this feature as well.

(This page has been intentionally left blank)

Bibliography

Aitchison, R. (2011). *Pro dns and bind 10* (1st ed. 2011. ed.). Berkeley, CA: Apress.

AndyF1. (2018a). *Maltego and osint for crime scene analysis.* https://medium.com/@andrewfnam/maltego-and-osint-for-crime-scene-analysis-942343790199. ([Online; accessed 2-April-2020])

AndyF1. (2018b). *Using maltego for cell phone analysis and geolocation with osint.* https://medium.com/@andrewfnam/using-maltego-for-cell-phone-analysis-and-geolocation-with-osint-19091a8de206. ([Online; accessed 2-April-2020])

Beeharry, J., & Nowbutsing, B. (2016). Forecasting ipv4 exhaustion and ipv6 migration. In *2016 ieee international conference on emerging technologies and innovative business practices for the transformation of societies (emergitech)* (p. 336-340).

Berners-Lee, T., Fielding, R., & Masinter, L. (2005). *Uniform Resource Identifier (URI): Generic Syntax.* doi: 10.17487/RFC3986

Catakoglu, O., Balduzzi, M., & Balzarotti, D. (2016). Automatic extraction of indicators of compromise for web applications. In *Proceedings of the 25th international conference on world wide web* (p. 333–343). Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee. Retrieved from https://doi.org/10.1145/2872427.2883056 doi: 10.1145/2872427.2883056

Center for Internet Security. (2020a). *Cybersecurity spotlight – internet protocol.* https://www.cisecurity.org/spotlight/cybersecurity-spotlight-internet-protocol/. ([Online; accessed 2-September-2020])

Center for Internet Security. (2020b). *Ei-isac cybersecurity spotlight – typosquatting.* https://www.cisecurity.org/spotlight/ei-isac-cybersecurity-spotlight-typosquatting/. ([Online; accessed 2-September-2020])

Chauhan, S. (2015). *Hacking web intelligence : open source intelligence and web reconnaissance concepts and techniques.* Amsterdam, [Netherlands: Syngress.

Ciflikli, C., Gezer, A., & Ozsahin, A. (2012). Packet traffic features of ipv6

and ipv4 protocol traffic. *Turkish Journal Of Electrical Engineering And Computer Sciences*, *20*(5), 727-749.

cmlh. (2016). *Maltego virustotal.* https://github.com/cmlh/Maltego-VirusTotal. ([Online; accessed 30-August-2020])

D'Angelo, V. (2019). *Dns, domain names, and certificates: The missing links in most cyber security risk postures.* https://cybertechaccord.org/dns-domain-names-and-certificates-the-missing-links-in-most-cyber-security-risk-postures/. ([Online; accessed 2-September-2020])

Esman, G. (2018). *Connecting the dots: Tracking identity of ddos-for-bitcoins criminal service operator with maltego, splunk and domaintools.* http://www.mensk.com/tracking-identity-of-ddos-for-bitcoins-criminal-service-operator-with-maltego-splunk-and-domaintools/. ([Online; accessed 2-April-2020])

Hassan, N. A. (2018). *Open source intelligence methods and tools a practical guide to online intelligence* (1st ed. 2018. ed.). Berkeley, CA: Apress.

Jercich, K. (2020). *Ransomware attack leaves 5 years of patient records inaccessible at colo. hospital.* https://www.healthcareitnews.com/news/ransomware-attack-leaves-5-years-patient-records-inaccessible-co-hospital. ([Online; accessed 31-August-2020])

Jones, C. (2019). *Dns 101 - how dns helps and hurts your network security!* https://www.titanhq.com/blog/dns-101-how-dns-helps-and-hurts-your-network-security/. ([Online; accessed 2-September-2020])

kpcyrd. (2020). *kpcyrd/virustotal-subdomains.* https://sn0int.com/r/kpcyrd/virustotal-subdomains. ([Online; accessed 31-August-2020])

Lampyre. (2020). *Intro to lampyre.* https://lampyre.io/documentation/documentation/about.html. ([Online; accessed 30-August-2020])

Lampyre. (2020). *Tutorial 4 - financial analytics.* https://lampyre.io/documentation/documentation/tutorials/financial-analytics.html. ([Online; accessed 2-September-2020])

Lookingglass. (2014). *Maltego.* https://github.com/Lookingglass/Maltego/blob/master/VTtransforms.py. ([Online; accessed 30-August-2020])

Maltego. (2019a). *Infrastructure.* https://docs.maltego.com/support/solutions/articles/15000018989-infrastructure. ([Online; accessed 16-April-2020])

Maltego. (2019b). *Infrastructure.* https://docs.maltego.com/support/solutions/articles/15000018992-locations. ([Online; accessed 16-April-2020])

Maltego. (2019c). *Penetration Testing.* https://docs.maltego.com/support/solutions/articles/15000018995-penetration-testing. ([Online; accessed 13-July-2020])

Maltego. (2019d). *What is a transform?* https://docs.maltego.com/support/solutions/articles/15000015758-what-is-a-transform-. ([Online; accessed 3-April-2020])

Maltego. (2019e). *Writing local transforms in Python.* https://docs.maltego.com/support/solutions/articles/15000017605-writing-local-transforms-in-python. ([Online; accessed 04-May-2020])

Maltego. (2019f). *Your first graph.* https://docs.maltego.com/support/solutions/articles/15000008832-your-first-graph. ([Online; accessed 8-April-2020])

Maltego. (2020a). *Maltego servers.* https://www.maltego.com/maltego-servers/. ([Online; accessed 31-August-2020])

Maltego. (2020b). *What is maltego?* https://docs.maltego.com/support/solutions/articles/15000019166-what-is-maltego-. ([Online; accessed 1-April-2020])

Microsoft. (2020). *PE Format - Win32 apps.* https://docs.microsoft.com/en-us/windows/win32/debug/pe-format. ([Online; accessed 23-August-2020])

Mimecast. (2019). *Content examination definitions: Using fuzzy hashing.* https://community.mimecast.com/s/article/Content-Examination-Definitions-Using-Fuzzy-Hashing-809049828. ([Online; accessed 3-September-2020])

Pastor-Galindo, J., Nespoli, P., Gomez Marmol, F., & Martinez Perez, G. (2020). The not yet exploited goldmine of osint: Opportunities, open challenges and future trends. *IEEE Access*, *8*, 10282-10304.

RIPE NCC. (2016). *The internet registry system.* https://www.ripe.net/participate/internet-governance/internet-technical-community/the-rir-system. ([Online; accessed 10-April-2020])

riskiq. (2020). *Passive dns.* https://www.riskiq.com/platform/architecture/internet-data-sets/passive-dns/. ([Online; accessed 14-July-2020])

Sachs, D. (2016). *Why mx records matter in the fight against bec and spear phishing.* https://www.infosecurity-magazine.com/opinions/mx-records-matter-fight-bec-spear/. ([Online; accessed 2-September-2020])

Shaw, D. (2015). *Nmap essentials : harness the power of nmap, the most versatile network port scanner on the planet, to secure large scale networks.* Birmingham, England ;: Packt Publishing.

Sn0int. (2020). *sn0int registry.* https://sn0int.com/. ([Online; accessed 30-August-2020])

SpiderFoot. (2020). *Documentation.* https://www.spiderfoot.net/documentation/. ([Online; accessed 30-August-2020])

Stevens, W. R. (1993). *Tcp/ip illustrated: Vol. 1: The protocols.* Place of publication not identified: Addison Wesley Professional.

tampererer. (2018). *Virustotal private api maltego.* https://github.com/tampererer/VirusTotal-Private-API-Maltego. ([Online; ac-

cessed 30-August-2020])

VirusTotal. (2020a). *How it works - virustotal.* https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works. ([Online; accessed 1-April-2020])

VirusTotal. (2020b). *Key Concepts.* https://developers.virustotal.com/v3.0/reference#key-concepts. ([Online; accessed 13-April-2020])

VirusTotal. (2020c). *Overview.* https://developers.virustotal.com/v3.0/reference#overview. ([Online; accessed 26-May-2020])

VirusTotal. (2020d). *Relationships.* https://developers.virustotal.com/v3.0/reference#relationships. ([Online; accessed 16-April-2020])

VirusTotal. (2020e). *Virustotal graph commonalities.* https://support.virustotal.com/hc/en-us/articles/360004843838-VirusTotal-Graph-Commonalities-and-Hunting. ([Online; accessed 31-August-2020])

VirusTotal. (2020f). *VirusTotal Graph nodes.* https://support.virustotal.com/hc/en-us/articles/360004687837-VirusTotal-Graph-nodes. ([Online; accessed 1-April-2020])

VirusTotal. (2020g). *Virustotal graph overview - virustotal.* https://support.virustotal.com/hc/en-us/articles/360004679937-VirusTotal-Graph-overview. ([Online; accessed 1-April-2020])

VirusTotal. (2020h). *VT Graph.* https://www.virustotal.com/gui/graph-overview. ([Online; accessed 13-April-2020])

VMWare Carbon Black. (2020). *About us.* https://www.carbonblack.com/about-us/. ([Online; accessed 23-August-2020])

WhoisXMLAPI. (2020). *Understanding and securing your dns records with a dns history lookup resource.* https://reverse-ip.whoisxmlapi.com/blog/understanding-and-securing-your-dns-records-with-a-dns-history-lookup-resource. ([Online; accessed 2-September-2020])

Wikipedia contributors. (2020a). *Antivirus software — Wikipedia, the free encyclopedia.* https://en.wikipedia.org/w/index.php?title=Antivirus_software&oldid=955259720. ([Online; accessed 25-May-2020])

Wikipedia contributors. (2020b). *Hash function — Wikipedia, the free encyclopedia.* https://en.wikipedia.org/w/index.php?title=Hash_function&oldid=955276927. ([Online; accessed 23-May-2020])

Wikipedia contributors. (2020c). *Overlay (programming) — Wikipedia, the free encyclopedia.* https://en.wikipedia.org/w/index.php?title=Overlay_(programming)&oldid=940169883. ([Online; accessed 23-August-2020])

Wikipedia contributors. (2020d). *Url — Wikipedia, the free encyclopedia.* https://en.wikipedia.org/w/index.php?title=URL&oldid=951945382. ([Online; accessed 22-April-2020])

Yip, M. (2015). *Maltegovt.* https://github.com/michael-yip/MaltegoVT. ([Online; accessed 30-August-2020])