Tartarus: A honeypot based malware tracking and mitigation framework

Samuel Oswald Hunter
Dept. Computer Science
Rhodes University
Grahamstown, South Africa
Email: shunter.dot@gmail.com

Barry Irwin
Dept. Computer Science
Rhodes University
Grahamstown, South Africa
Email: b.irwin@ru.ac.za

***Abstract*—On a daily basis many of the hosts connected to the Internet experience continuous probing and attack from malicious entities. Detection and defence from these malicious entities has primarily been the concern of Intrusion Detection Systems, Intrusion Prevention Systems and Anti-Virus software. These systems rely heavily on known signatures to detect nefarious traffic. Due to the reliance on known malicious signatures, these systems have been at a serious disadvantage when it comes to detecting new, never before seen malware. This paper will introduce Tartarus which is a malware tracking and mitigation framework that makes use of honeypot technology in order to detect malicious traffic. Tartarus implements a dynamic quarantine technique to mitigate the spread of self propagating malware on a production network. In order to better understand the spread and impact of internet worms Tartarus is used to construct a detailed demographic of potentially malicious hosts on the internet. This host demographic is in turn used as a blacklist for firewall rule creation. The sources of malicious traffic is then illustrated through the use of a geolocation based visualisation.**

I. Introduction

Malicious entities are common on the Internet and most often fall into a category of Malware such as worms, viruses and Trojan horses. This malware is often the source of further malicious activity such as Distributed Denial of Service(DDOS) attacks, spam emails and the hosting of illegal content. While traditional methods of detecting malware such as Anti-Virus (AV) software and Intrusion Detection Systems (IDS) have proven useful, they on their own are only able to provide limited information[1]. A honeypot on the other hand could be designed to emulate vulnerable services which could allow for controlled interaction and the collection of malware samples[2]. Thus removing the problem faced by traditional IDS of having to discriminate between legitimate and illegitimate traffic.

The process of collecting malware samples has been simplified through the use of honeypots such as Nepethes or Dionaea, these honeypots are referred to as low interaction honeypots [2]. They emulate vulnerable services from the Microsoft Windows operating systems in order to entice and capture self propagating malware such as worms [3].

This paper will serve as an introduction to the Tartarus framework which is still in its initial stages of development. Tartarus is a malware tracking and mitigation framework which makes use of honeypots and traffic analysis from network telescopes to model, track and visualise potentially malicious hosts on the internet. The modular design made it possible to easily integrate additional modules into the Tartarus framework, these modules could increase its data collection capabilities and defensive applications.

Malware communication would be tracked and displayed through the use of a geolocation based visualisation module. This module makes use of a database that maps IP address's to geographic location and visualises this communication through the use of the Google Maps API. The Tartarus framework makes use of a Dionaea honeypot to capture malware samples and identify malicious host machines where the malware infection originated from. Furthermore, the targets of malware communication will be enumerated by an active host fingerprinting module which makes use of Nmap[1] to perform port scans and operating system identification. With the collection of geolocation and host enumeration data, Tartarus is able to provide a rich dataset of aggregated information. Using this information Tartarus would be able to construct a detailed demographic of potentially malicious hosts on the internet.

The Tartarus framework would also provide pro-active and re-active defensive techniques in order to protect a local network, this is accomplished through blacklisting and a dynamic quarantine. The Tartarus framework creates a collection (blacklist) of known bad or malicious hosts on the internet, it does this by noting the sources of malicious activity identified by a honeypot and network telescope. The threat mitigation through blacklisting is based on the principle of trust between the protected internal network and external hosts. The responsive defence is achieved through a dynamic quarantine module that isolates and effectively quarantines a host suspected of malicious behaviour internal to the network, this is achieved through the use of ARP-poisoning.

A. Document Structure

In order to properly introduce the Tartarus framework Section II will address related research in the fields of self propagating malware, defensive techniques and the analysis of

[1]http://nmap.org/ - Nmap ("Network Mapper") is a free and open source (license) utility for network exploration or security auditing

such malicious malware and its interaction. In Section III we provide an overview of Tartarus and the various components the framework relies on. The roles and interaction between the various components is described in order to properly introduce each of the core components in more detail. Section IV describes the Data Manager component and the various data sources the Tartarus framework makes use of to collect data. It is then shown in Section V how the flow of information is managed by the Information Engine. The paper then describes the last component of the Tartarus framework in Section VI; the Defence core and is responsible for the threat mitigation from both internal and external malicious hosts.

II. Related Work

In 1988 the Morris worm infected thousands of computer over the internet by exploiting a software vulnerabilities. It was seen as the first well known self replicating program and it opened the floodgates for later nefarious pathogens. In 2001 the Code Red worm infected over 359 000 computers in less than 14 hours [4]. This was followed by SQL Slammer in 2003 which infected over 90% of vulnerable hosts within the first 10 minutes of its release[5]. With the advent of Conficker towards the end of 2008 the internet was awry with malicious pathogens, a conservative estimation of hosts infected with Conficker alone put the number around 3 million [6]. More recently the Stuxnet worm brought to light the concept of worms used in cyber terrorism. With increasing Internet coverage and bandwidth speeds and a decrease in costs across the globe [7] more hosts are gaining access to the internet. This is even more apparent in developing countries where software piracy rates are very high [8], the attack vector for these internet worms are growing at a phenomenal rate. Research[9] has shown that through the use of flash worms that seed their initial attack platform are able to infect a vulnerable population in 10s of seconds which is much faster than any human mediated defensive response could be possible[9]. Infecting thousands or even millions of hosts on the internet does not only surrender private data from the compromised hosts, but also allows for the corruption of data on a mass scale. Information theft while being a serious issue, is however not the only after effect from infected hosts, these hosts are often used in massive distributed denial of service attacks, capable of bringing down anything from a large e-commerce store to a countries internet backbone.

A. Honeypots and Malware Analysis

A honeypot is a device or service that operates in a network and is used to detect and detail any form of nefarious or malicious interaction initiated with it. Honeypots can be deployed in many different flavours, a single physical host could act as a high interaction honeypot, alternatively that same host could emulate multiple virtual honeypots [2]. Low and high refer to the level and quality of interaction an attacker or malicious entity can have with a honeypot. It should be noted that while an IDS might be similar to a honeypot they are not the same. An IDS is dependent on the vulnerability signatures it has at its disposal to identify malicious traffic. A honeypot such as Nepethes or its successor Dionaea are able to capture new malware samples that have not yet been encountered[3].

The goal of malware analysis is to understand how a specific sample of malware operates in order to build defences against it. There are two main types of malware analysis namely static[10] and dynamic[11]. Static malware analysis is concerned with the analysis of the malware binary and reverse engineering the source code, static analysis tools attempt to analyse malware without actually executing the binary[12]. Dynamic or ”live” analysis on the other hand is concerned with the study of malware’s behaviour once it has been executed. Tartarus is concerned with the origin of malware for defensive purposes and the type of malware in order to construct a malicious host demographic.

1) Defensive Techniques: There has been a great deal of research surrounding the detection and threat mitigation of self propagating worms in order to minimise their affect on networks[13][14][15][16]. An example of a pre-emptive network defence is provided by Antrosio and Fulp[13], they created a system that pro-actively seeks out vulnerable hosts on a local network and quarantines them in order to prevent them from being compromised. While their method appears effective, it comes at a cost of automation, manual configuration of exploits are required which results in a more taxing process. Tartarus provides re-active defence in a completely automated fashion, once a malicious host has been discovered on the network through a honeypot it is immediately quarantined. Researchers have also developed a method of interfering with the spread of self propagating malicious code by throttling the connections made by such malicious agents[14] [17]. This concept of connection throttling was introduced by LaBrea [2] which is used to slow down and in some cases completely stop self propagating TCP based worms. Worms however are not the only malicious denizens that could be targeted, HoneySpam [15] is another example of a honeypot that tries to disrupt malicious activity, it is used to slow down the process of email harvesting and also makes it easier to trace spammers by deploying fake open proxies and open relays.

III. Tartarus Framework

At the time of writing Tartarus consisted of partially completed components in order to illustrate a proof of concept malware tracking and mitigation framework that would make use of honeypots and network telescopes to detect malicious traffic. The framework is currently targeted towards use in small business networks but is designed to be scalable so that it could eventually be deployed at Internet Service Provider (ISP) level. While the construction of a malicious host demographic is more geared towards academic research, the results could be equally valuable if applied towards real-word defensive

[2]http://labrea.sourceforge.net/labrea-info.html - LaBrea ”sticky” honeypot and intrusion detection system.

applications. A geolocation based visualisation has also been created to illustrate the sources of malicious traffic along with the frequency at which they have been observed. The visualisation was created so that the data could be easily observed, understood and navigated. In order to demonstrate the real-word application of malicious host demographic data, Tartarus would generate and maintain a list of potentially malicious hosts that it has observed on the Internet. This list would then be used to create firewall rule sets to deny access to and from these hosts. In order to fight threats internal to a network Tartarus would also able to mitigate malicious activity from hosts through a dynamic quarantine procedure. The purpose of this quarantine is to isolate malicious hosts on the network before they can harm other hosts, giving system administrators time to restore a compromised host to a safe state.

Functions and tasks were separated in the Tartatus framework according to core components and modules, this was done to insure a modular design and was implemented by making use of event based messaging between the various components and modules. The framework consists of three core components:

Data Manager
: The data manager component is responsible for collecting data from any source it has been configured to, currently the Tartarus framework collects data from a honeypot and a network telescope. It provides access to this information and initiates the retrieval of additional information through the Information Engine whenever new data is captured.

Information Engine
: The delegation of tasks according to events received from the Data Manager is handled by the Information Engine. It is also responsible for the retrieval of additional information such as host enumeration data and information from third party sources over the Internet such as projecthoneypot.org. Host enumeration data is obtained through a script that interfaces with Nmap and is outlined in Section V-B. Complementing information from online sources are retrieved through the use of scripts that request web based data and interpret the response.

Defence Core
: The Defence Core is responsible for network based defensive measures implemented through the Tartarus framework. This includes blacklist generation and the dynamic quarantine of potentially malicious machines on an internal network, this is achieved through an ARP-poisoning module. The defensive techniques along with their limitations are discussed in more detail in Section VI.

The frameworks design through abstraction of function allows for easier integration of addition functions or data sources in the future. For instance additional honeypot data could be gathered though the Data Manager, all that would be required is for the data to be supplied in a suitably serialised manner. The basic flow of data through the Tartarus framework is illustrated in Figure 1. An event would be registered through the Data Manager component and passed to the Information Engine for analysis. The Information Engine would then decide what should be done with the event based on its characteristics. If the traffic source is from the local network it would pass the event to the Defence Core along with suitable instructions. If the traffic source is external to the network and new (it has not been seen before) the event is marked as new and further analysis is done. If the source is known, the number of observations is incremented and the last seen field updated. The data event consists of the following:

- Event ID
- Timestamp
- Source IP
- Host Name
- Source Port
- Destination Port
- Type

During analysis additional information is added to the event before it is inserted into the Tartarus database.

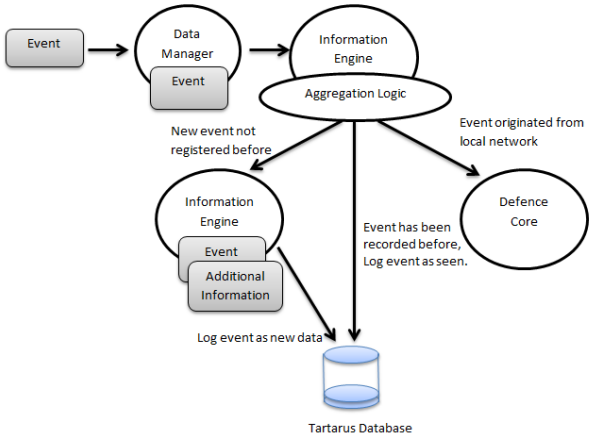

Fig. 1. An event such as a malicious host connecting to a honeypot occurs and is registered by the Data Manager. The data is then passed on to the Information Engine for further analysis.

IV. Data Manager and Data Sources

The Data Manager component maintains all the data sources that are used by the Tartarus framework, the data provided by these sources are different from the Information Engine, as it consists of original events that have been detected as opposed to aggregated information. Network telescopes and honeypots are not normal hosts that one would expect to find on a network. They should not receive any legitimate traffic and as such we can assume that any traffic destined for them is un-solicited and potentially malicious. This makes them ideal for the collection of nefarious traffic. As a result of not expecting any legitimate traffic it greatly reduces the amount of filtering that would have been required.

A. Dionaea Honeypot

The Tartarus framework makes use of a Dionaea honeypot as its primary data source, while Tartarus is intended to use multiple honeypots and network telescopes, at the time of writing only one honeypot was used. Dionaea is a low interaction honeypot that exposes multiple vulnerable services in an effort to capture malware samples or interaction from malicious entities. It supports the following network protocols smb, http, ftp, tftp, msql and sip. Dionaea stores all connection information in a local sqlite database but could also be configured to expose data over XMPP. Currently the sqlite database is queried for new connection information at short periodic intervals.

B. Network Telescope

A network telescope in its most basic form would be a host on a network that traffic from often multiple addresses or an entire address block is routed. If the network telescope is passive it would simple log all incoming traffic. Active network telescope such as those used by the Internet Motion Sensor(IMS) make use of a lightweight responder to complete the 3-way TCP handshake in order to receive even more packet data [18]. A network telescopes ability to capture traffic destined for unused address space results in it receiving little or no legitimate traffic. As a result a network telescope is highly suited for providing information regarding DDoS attacks and the automated propagation of internet based worms and viruses. As the vast majority of denial of service attacks detected by a network telescope is as a result of reflected traffic from backscatter, it is not able to identify malicious sources and as such is not used by the Tartarus framework. Traffic generated by self propagating worms on the other hand is used by the Tartarus framework to identifify potentially malicious hosts. The use of a network telescope as a data source greatly increases the speed at which Tartarus is able to construct a malicious host demographic. At the time of writing the Tartarus framework made use historic data captured by a network telescope, while this data was not highly effective for the creation of a host demographic due to its age it was effective at showing that heterogeneous data sources can be used by the Tartarus framework.

V. Information Engine

The core responsibility of the Information Engine was to manage the flow of information. It would also initiate the process of obtaining additional information that would complement already owned data and aggregate this information. When an event is registered in the Data Manager and passed to the Information Engine it undergoes a set of heuristics to determine what would need to happen with the event data. These heuristics reside in the Aggregation Logic module along with other functionality such as aggregating data from multiple sources and inserting information into a database. The remainder of this section will outline the modules contained within the Information Engine and describe their functions.

A. Aggregation Logic

The Aggregation Logic module would be responsible for managing data flow decisions, aggregating data and storing all this information in a database. The heuristic steps involved during the event delegation are shown in Figure 2. A large proportion of the decisions are based on the source IP address of the event, if it is local then the event would be passed to the Defence Core and logged. When an event does not originate from the local network it would be cross referenced with events that have been observed in the past. If the current event was to match a past event the frequency (occurrence) of the event would be incremented and a new last seen value would be saved to the database. If however the event was new, then the Aggregation Logic module will pass the event details to the Host Enumeration module as well as any other information retrieval modules that have been implemented. If heterogeneous data sources exist the module would attempt to cross reference shared attributes between the event and data sources in order to extract correlating information. An example of this could be the source IP address detected by a honeypot as the origin of a malware sample when the same IP address is found to be scanning addresses by a network telescope data source. This correlation could be validated by performing host enumeration (as discussed below) during both the occurrence of the port scanning and the detection of the malware to determine if the source IP might have belonged to the same host.

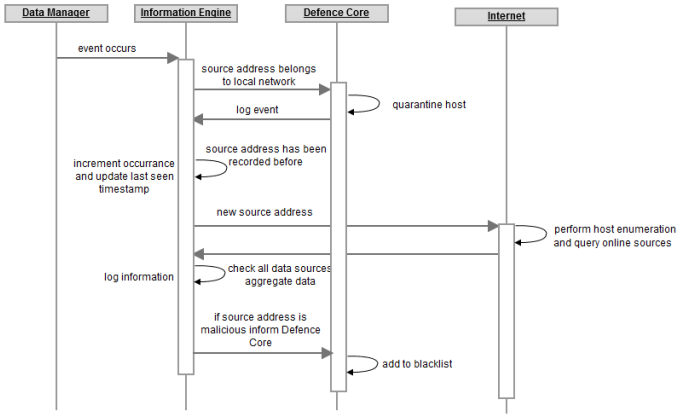

Fig. 2. Sequence diagram illustrating the communication between the core components of the Tartarus framework.

B. Host enumeration

The Host Enumeration module was implemented through a script that interfaced with Nmap; a popular network scanner and operating system fingerprinting tool. The module takes an IP address as input and performs a port and Operating System (OS) identification scan on the supplied IP address. It does this using the -O -NP flags for nmap, the script needs to run with root privileges in order to perform the OS identification. The results from the scan could provide valuable information regarding the source of an event. This information includes the

open ports on that host, which in turn allows us to infer the services running on that host. An example of information that could be inferred from the results of a port scan follows. One of the hosts that was observed had ports 6667-6669 open, this tells us that the host is being used as an IRC[3] server which is a well known method for controlling botnets.
Host enumeration is also the main source of information with regards to the construction of a malicious host demographic.

C. Geolocation Lookup and Visualisation

The Information Engine includes a module that looks up the geographic location of a given IP address, this is done by querying a MaxMind[4] GeoLite City database. The database is in optimized binary format for fast query responses and is exposed through a Python API. The Information Engine makes use of the Geolocation Lookup module to retrieve the latitude and longitude for IP addresses which is in turn used by a geolocation based visualisation for traffic sources. The google maps API engine is used to display this information.

The visualisation checks for new entries in a sqlite database every five seconds, this database is populated by the Information Engine whenever a new event is registered. If new entries are found in the sqlite database they are added to the map through the use of a marker, each marker has an info window that contains information regarding the host it represents. Depending on the source of the event and what information gathering modules have been used the info windows could show which ports are open on the host, the number of times the host has been observed, the first and last time the host was observed.

Markers are colour coded to indicate age, the last 25 hosts to have been added to the map have green markers, the next 150 have yellow coloured markers and the oldest 300 hosts are represented with grey markers. The use of markers for host locations are illustrated in Figure 3. Only the info windows of the last five hosts are open, as a new host is added the fifth marker's info window would be closed, this is done so that the visualisation does not become over encumbered with open info windows. Any markers info window can however be opened again by simply clicking on the marker. Figure 4 shows how the geolocation based visualisation looks while Figure 5 shows an info window for a malicious host in Australia.

D. On-line information retrieval

There are valuable sources of information on the internet that can be used in collaboration with our own data. The combination of data results in a rich dataset of information that can be used to draw more assertive conclusions from. An on-line source such as www.projecthoneypot.org provides information obtained from many distributed honeypots regarding potential malicious hosts on the internet and those hosts

[3]http://www.irc.org/ - Internet Relay Chat is a form of real-time text messaging.

[4]http://www.maxmind.com/app/geolitecity - Provides worldwide coverage with 99.5% accuracy on a country level and 79% on a city level within a 25 mile radius

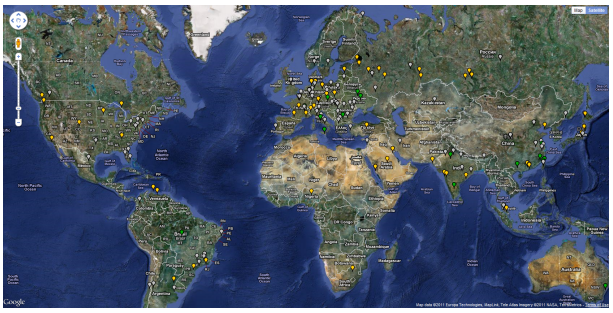

Fig. 3. Marker placement for malicious source addresses.

Fig. 4. Geolocation based host enumeration from network telescope data with open info windows.

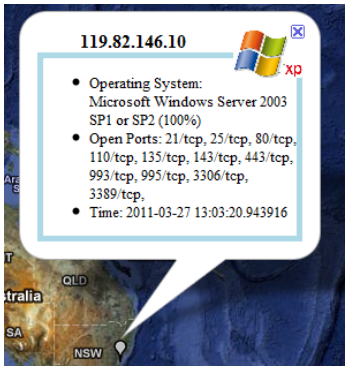

Fig. 5. Info window showing the host enumeration results from a malicious source in Australia.

activities, this could be used to complement our own honeypot data. The Tartarus framework would make use of a retrieval module that could extract data from webpages in order to harvest information from those sites.

E. Constructing a Malicious Host Demographic

The construction of a malicious host demographic is achieved by firstly identifying the malicious hosts and then retrieving as much information as possible about those hosts. This is currently achieved through a host enumeration module but would also make use of on-line information retrieval and other analysis tools. Through the aggregation of this information Tartarus is able to construct a detailed demographic of potentially malicious hosts on the internet. This demographic

would then be used for further analysis, for example the host demographic could allow us to identified high densities of malicious activity according to geographical location. Another example of information that could be learned from the host demographic is the time of day that most attacks happen.
The host demographic would also assists in the creation of firewall rules or adjusting levels of trust based on attributes such as geographical location, ISP, time of day and targeted port. Additionally data such as most active host, most malicious address space and most targeted ports could all be learned from the host demographic. The host demographic that would be generated by the Tartatus framework could thus help us to better understand at least a portion of the malicious hosts connected to internet.

VI. Defence Core

The Defence core is responsible for threat mitigation techniques in the Tartarus framework. At the time of writing two threat mitigation functions have been implemented; blacklisting and dynamic quarantine. Section II highlighted the seriousness and threat posed by self propagating worms on the internet, while the proposed mitigation techniques are not a perfect solution against all threats we do believe that they hold substantial value in securing a network.

A. Blacklist

The defence core maintains a collection of bad or potentially malicious hosts on the internet, those hosts of been flagged by the Information Engine as potentially malicious and stored in a local database. By using this database the defence core is able to generate various forms of blacklists for application in network defence. Examples of this include firewall rules or a DNS blacklist. In a future implementation of Tartarus these blacklists will also be exposed as RSS feeds.

B. Trust Engine

IP addresses change and are not always static, as a result the Trust Engine ”checks on” blacklisted hosts to make sure that their IP address still belongs to the correct malicious host. This is done by doing a nmap scan of the host again, hashing it and comparing it to a stored hash of the host when it was first detected. The problem with this method is that if the malicious host opened or closed a port during the interim of last being checked, the hashes would be different. A proposed solution to this problem is to compare the number of differences (if any) between the the current and last scan of that host and determine an acceptable margin of error. The scan data could of course be complemented through additional information such as that provided by a whois or traceroute which would provide a more detailed identity for a host.

C. Malicious Host Quarantine through ARP Poisoning

ARP poisoning is a technique that is well known for its use in attacking wired and wireless networks during man in the middle attacks. It is used to sniff packets between two hosts on a network that are within the same subnet as the attacker. These packets could also be modified or even dropped in the case of a denial of service attack. The Tartarus framework makes use of ARP poisoning to successfully quarantine traffic from a potentially malicious internal host through a denial of service, this is done until the given host can be serviced and returned to a safe and clean state.
ARP poisoning works by associating the attackers MAC address with a the target of the victims communication. This is achieved by sending our own ARP responses to the victim and thereby placing a fake entry into its ARP cache. Figure 6 shows an example of ARP poisoning, whereby the attacker has assumed the IP address of two hosts by poisoning their ARP tables. There are other legitimate uses for ARP poisoning, often network registration tools would make use of it to redirect unregistered hosts on a network to a registration page or its used as a method of implementing redundancy in the case of a malfunctioning server being taken over by another.

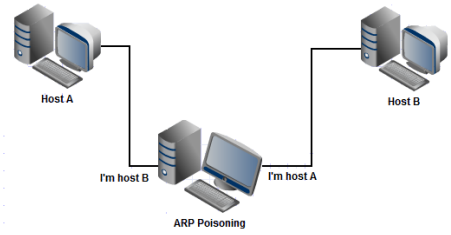

Fig. 6. An example of how ARP-poisoning works between hosts.

The ARP poisoning functionality in the Tartarus framework is provided by a module that interfaces with ettercap[5] A list of hosts that are currently under quarantine is maintained in a sqlite database, this list is maintained by the module. It was shown in Section V-A how events are delegated to different parts of the Tartarus framework based on their attributes, when it is determined that a host should be quarantined by the Information Engine an alert is also sent to a system administrator. This alert is sent via email and contains information on the host that is being quarantined, once a systems administrator has seen too the quarantined host they can lift the quarantine through the dynamic quarantine module. Figure 7 shows a class diagram exposing the basic functions of the dynamic quarantine class.

This method of dynamic quarantine is used to lesson the impact of self propagating malware on a local network by isolating malicious sources and giving network administrators time to address the problem. It is however not without its drawbacks, unfortunately due to the nature of the Address Resolution Protocol this would only work on a relatively small network that would not require packets to be routed across different subnets. A possible solution to this is discussed in Section VII through interaction with a managed switch.

[5]http://ettercap.sourceforge.net/ - Ettercap is a suite for man in the middle attacks on LAN. It features sniffing of live connections and on the fly content filtering. It supports active and passive dissection of many protocols and includes many feature for network and host analysis.

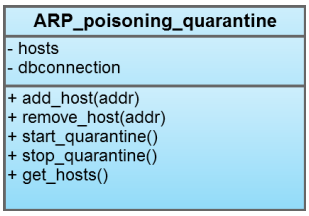

Fig. 7. A Class diagram that represents the dynamic quarantine module that interfaces with ettercap.

Another problem with this method of quarantine is that you are effectively performing a denial of service attack on a given host, this approach might be considered over zealous as it would be more efficient to simple block traffic to a certain port in which case an appropriate plugin to interact with the firewall or switch could be developed. If a worm such as Conficker is attempting to exploit the ms08-067 [19] vulnerability over port 445, it would be sufficient to simple drop all traffic targeting port 445. This can be done by creating filters for Ettercap, these filters match traffic patterns and are most commonly used during man in the middle attacks to inject or modify sniffed packets. In the case of the Tartarus framework, Ettercap filters can be created to drop intercepted packets destined for a given port from quarantined hosts, thus allowing the hosts to still operate productively on a network while halting the malware's ability to spread. This would in essence have the same effect as a Labrae tarpit, without being limited to only TCP based worms.

VII. Future Work

While the current implementation of Tartarus is limited by its infancy, there are multiple extensions planned to increase its situational awareness and defensive capabilities. Tartarus is dependent on data sources to interpret and understand the environment in which it has been placed, for this reason additional data sources will be integrated into the Tartatus framework in the future. The objective of additional data sources will be to assist Tartarus to model and monitor its surroundings more effectively. This would include a network mapping module that could perform scans of the local network in order to construct and monitor the hosts connected to the network. This could be used to generate alerts when an unauthorised host is connected to the network or immediately quarantine such hosts. This module would also be combined with the host enumeration module to monitor the services running on hosts connected to a network, this would be useful in order to detect and flag potentially malicious services associated with ports such as those that are used by back-door software.

The current method of quarantine through ARP-poisoning while effective for a small network would not work on a substantially large network as the Address Resolution Protocol is not route-able across subnets. As a result of this limitation a more extensive solution would be to use a form of dynamic route injection across managed edge point switches. When malicious behaviour is detected the appropriate switch would be queried to return the port associated with the malicious host. At which point the switch would be instructed to bind that port to an unused VLAN.

Lastly a ticket and progress management module is proposed, this module would issue alerts to system administration whenever a potentially malicious host is detected and allow administrators to update the progress of attending to the host. With the implementation of these proposed extensions, the Tartarus framework could become a core component for monitoring and defence of a network.

VIII. Conclusion

Honeypot technology has proven to be an excellent means for detecting and identifying known and never before seen malware. As such it has been a perfect source of information for the Tartarus framework and has allowed the framework to track, visualise and defend against malware. The Tartarus framework has effectively made use of a Dionaea honeypot as a data source in its Data Manager component. It made use of the information that was provided by the honeypot to investigate and defend against potentially malicious hosts both on the internet and local network. Through the use of a Host Enumeration module Tartarus was able to infer the services and operating system of hosts from which malware originated. This information has proven useful in the construction of a malicious host demographic which would in turn help us to better understand the state of malicious activity on the internet. Through the use of the dynamic quarantine module the Tartarus framework used ARP-poisoning to isolate a malicious host on the network and in so doing provide system administrators a chance to restore the network to a safe state. While the Tartarus framework is still in its initial stages of development, it has the potential to become an effective tool for the detection, visualisation and defence against malware.

Acknowledgement

The authors would like to thank the CSIR DPSS for funding and support of this research.

References

[1] N. Ierace, C. Urrutia, and R. Bassett, "Intrusion prevention systems," *Ubiquity*, vol. 2005, pp. 2–2, July 2005. [Online]. Available: http://doi.acm.org/10.1145/1071925.1071927

[2] N. Provos and T. Holz, *Virtual honeypots: from botnet tracking to intrusion detection*, 1st ed. Addison-Wesley Professional, 2007.

[3] Dionaea catches bugs. Onilne. [Online]. Available: http://dionaea.carnivore.it/; Last accessed: 27/04/2011

[4] D. Moore, C. Shannon, and k claffy, "Code-red: a case study on the spread and victims of an internet worm," 2002, pp. 273–284.

[5] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," vol. 1. Piscataway, NJ, USA: IEEE Educational Activities Department, July 2003, pp. 33–39. [Online]. Available: http://portal.acm.org/citation.cfm?id=939830.939954

[6] T. conficker working group, "Conficker," Internet, http://www.confickerworkinggroup.org/wiki/pmwiki.php/ANY/Introduction.

[7] C. C. Zou, W. Gong, and D. Towsley, "Worm propagation modeling and analysis under dynamic quarantine defense," 2003.

[8] R. D. Gopal and G. L. Sanders, "Global software piracy: you can't get blood out of a turnip," *Commun. ACM*, vol. 43, pp. 82–89, September 2000. [Online]. Available: http://doi.acm.org/10.1145/348941.349002

[9] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the internet in your spare time," 2002.

[10] M. Christodorescu and S. Jha, "Static analysis of executables to detect malicious patterns," in *In Proceedings of the 12th USENIX Security Symposium*, 2003, pp. 169–186.

[11] M. Egele, C. Kruegel, E. Kirda, H. Yin, and D. Song, "Dynamic spyware analysis," in *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2007, pp. 18:1–18:14. [Online]. Available: http://portal.acm.org/citation.cfm?id=1364385.1364403

[12] S. Harris, A. Harper, C. Eagle, and J. Ness, *Gray Hat Hacking, Second Edition*, 2nd ed. New York, NY, USA: McGraw-Hill, Inc., 2008.

[13] J. V. Antrosio and E. W. Fulp, "Malware defense using network security authentication," *Innovative Architecture for Future Generation High-Performance Processors and Systems, International Workshop on*, vol. 0, pp. 43–54, 2005.

[14] M. M. Williamson and M. M. Williamson, "Throttling viruses: Restricting propagation to defeat malicious mobile code," 2002.

[15] M. Andreolini, A. Bulgarelli, M. Colajanni, and F. Mazzoni, "Honeyspam: honeypots fighting spam at the source," in *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*. Berkeley, CA, USA: USENIX Association, 2005, pp. 11–11. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251282.1251293

[16] E. Cooke, F. Jahanian, and D. Mcpherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," 2005, pp. 39–44.

[17] J. McClurg, "Stop port scans with labrea," http://www.sans.org/reading_room/whitepapers/tools/stop-port-scans-labrea_405.

[18] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson, "The internet motion sensor: A distributed blackhole monitoring system," in *In Proceedings of Network and Distributed System Security Symposium (NDSS 05*, 2005, pp. 167–179.

[19] M. S. Bulletin. (2008, October) Microsoft security bulletin ms08-067 critical. Online. Microsoft. [Online]. Available: http://www.microsoft.com/technet/security/bulletin/ms08-067.mspx; Last accessed: 22/04/2011