University of the West of England

Can Visualizing Malware Propagation Across Local Area Networks Assist in Security Decision Making?

Jacob John Williams

Cyber Security MSc

"This study was completed for the MA/MSc in Cyber Security at the University of the West of England, Bristol. The work is my own. Where the work of others is used or drawn on it is attributed".

Word Count: TBI

1 Abstract

2 Introduction

Malware is perhaps one of the greatest threats in modern day IT, it is constantly evolving both terms of who it targets and what it exploits. Due to the evolutionary nature of malware modern security has turned into a game of cat and mouse, security analysts and malware writers always vying to overtake the other by finding new ways to protect or exploit systems. Unfortunately it is common in the industry for the malware writers always to be numerous steps ahead, with those who work in security being forced to operate reactively to malware, this is why the security industry needs more who are willing to study malware in order to prevent it.

How malware propagates is key to understand how malware operates, external propagation; that being how malware bridges the air gap between networks is commonly studied and is fairly established in various forms of study, mathematically and visually. How malware propagates within a local area network however is studied rather one sidedly, leaning towards the mathematical side. These studies are extremely comprehensive and explore numerous methodologies of formulating how malware propagates, but it requires some complex knowledge and understanding of mathematical models and theories and as a result could be seen as being rather unfriendly to those wishing to involve themselves with Cyber Security from other backgrounds. Therefore the goal of this study to to attempt to visualise malware propagation using variables and characteristics from machines on local area networks and produce a more easily interpretable way of showing the nature of propagation in order to assist in security decision making.

2.1 Aims and Objectives

1. To create clear and interpretable visualisations of propagations.
    a) To inform other researchers of the details of the propagation in order to assist in the further development of anti-malware capabilities.
    b) To inform those in positions whose decisions can affect the security of a network, and where they should be deploying network security features.
    c) To aid students in the study of malware behaviour and its impact across networks.
2. To identify key characteristics of malware propagation that can be used in the response to an attack.
    a) Is malware propagation in certain malware deterministic?
    b) Does IP address locality affect the nature of propagation?
    c) How does network structure affect propagation?
    d) Can this be used to effectively respond to an active threat event?

The first objective can be considered a general committal, the effort to expand the knowledge surrounding the nature of malware. One must acknowledge that we will be studying existing malware and are not necessarily looking to find brand new characteristics, but rather create visualizations that better display the nature of malware propagation.

The second objective is the crux of this research, an attempt to create better response times to an active attack event. This paper will explore whether what is gathered from the tool can be used to assist in decision making, such as dynamically disconnecting clusters of or singular machines from the network in an attempt to protect the greater network, or perhaps in a more complex manner, assist in the dynamic creation of HoneyPots or HoneyNets in order to protect networks or observe active malware events.

3 Related Literature

As aforementioned, the vast majority of research into malware propagation is mathematical in nature, but this does not make it irrelevant to this study, as researching what we already know of malware propagation will allow us to create hypotheses and possibly support not only our own findings but others findings as well. *Yu et al (2015)* explored the prospect of malware propagation by applying epidemic theory and creating a

two-layer epidemic model. Yu et al (2015) focus on epidemic modelling because it is *"more focused on the number of compromised hosts and their distributions"* compared to another method called Control System Theory, which Yu et al (2015) claims are more inclined to *"try to detect and contain the spread of malware"*. This distinction is interesting as it categorises the two types of studies seen in the field of malware propagation, those who observe the spread of malware to define its nature are studying it Epidemiologically, and those study it to configure systems to prevent propagation are studying it under Control System Theory. With this distinction in mind, one could claim that this study will be observing Epidemiologically. Yu et als (2015) results are interesting, providing evidence that propagation in large scale networks possess three different discernible stages, *"Exponential distribution in its early stage"*, *"power law distribution with short tail in late stage"*, and *"power law distribution in its final stage"*. This aligns with how one might imagine malware propagation, with it at first infecting everything it can and once there is nothing more to infect the rate of infection quite obviously decreases exceedingly quickly. Yu et al (2015) explores the rate at which malware propagates to great depths, but like many mathematical models doesn't explore the nature of propagation, such as whether the way it happens is deterministic or if IP address locality matters.

Guillen and Rey (2018) researched what is in concept an extension of Yu et als (2015) work. Guillen and Rey (2018) acknowledge the existence of *"compartment devices"* which are devices that *"cannot be directly targeted by malware, but can be used to propagate it"*. It presents an interesting figurative, the devices that propagate without being infected do not contribute towards the infection rate themselves but they greatly increase the general infection rate. This presented an interesting line of though, that if you were to visualise malware propagation solely through signatures or system resources then these compartment devices would seemingly be unaffected but devices near to them would seemingly spontaneously become infected. If one was to attempt to study malware behaviour in networks that frequently possess these compartment devices then network trace analysis would be key. Hosseni and Azgomi (2016) formulated a model for representing malware propagation based on a rumour spreading model. Identifying five different types of machines (nodes) that perform different roles in the propagation of malware, these being *"susceptible, exposed, infectious, recovered, vaccinated"*. This study lends well to classifying the different states a machine in an active attack on a LAN may take and may be useful to consider during visualization.

Zhuo and Nadjin (2012) conducted a study with the goal of visualizing malware network traces, in doing so they created the tool MalwareVis. MalwareVis is able to pull heterogeneous attributes like protocol, IP address, and more from the network trace log. Network traces are a key part of categorizing malware since often the malware has to communicate with an outside command and control server in order to deploy, receive orders, or exfiltrate data. Analysing network traces is often a long and arduous job in malware analysis because the logs contain large volumes of data, as a result, the study focuses more on specific protocols such as TCP and DNS. Whilst the study is interesting and attempts to visualize a typically complex part of malware analysis, it still produces a visualisation that cannot be understood without in-depth study of the formulas it is built upon. To the average researcher this will prove no trouble, but perhaps to those getting into the field it will prove much more difficult and perhaps yield very little to their understanding. Gove and Deason (2018) also explore the concept of identifying malware through network activity. The target of their study is the period network activity of malware, which they acknowledge as being a difficult task due to it being *"easily drowned out by non-malicious network activity"*. Gove and Deason (2018) utilise a novel algorithm based on Discrete Fourier Transforms, and they pair the output of this with aggregation summary tables that will inform users which detection are worth investigating and which are not. This study is highly complex but also very interesting, as it presents a method of analysing data that is usually very difficult to obtain. The downside is the method in which the data is collected is similarly as complex, whilst network traces might be an excellent way to study propagation in particular it is appearing that such a method may not be conceivable within the time frame of our study.

The study of malware detection is useful to this study due to the key overlap of malware characteristics. If certain characteristics are better at identifying malware then they will most certainly be better for visualising malware, therefore much study was conducted into the detection of malware and the identification of their characteristics. Rhode et al (2018) conducted study where they used Recurrent Neural Networks to predict malware, in the way

of investigating *"whether or not an executable is malicious based on a short snapshot of behavioural data"*. The recurrent neural network works using hyper-parameters, but what these hyper-parameters work with is what is interesting. Rhode et al (2018) compile minimum and maximum values of inputs to identify the benign or malicious nature of samples. These input values are things such as *"total processes, max process ID, cpu user (%), cpu system (%)"* and more, in short they identify malicious activity when the executable exhibits resource usage past a certain threshold which is typical of other malicious samples. This study is useful as it presents the concept of using system resources and set thresholds to visualise malware, one could visualise all resource usage but have the visualisation be different for those who pass the said threshold, perhaps in a green-amber-red fashion.

Mills et al also used a form of machine learning in order to detect malware. Mills et al used Random Forest to create NODENS, a lightweight malware detection platform that can be deployed on cheaper hardware. After *"removing duplicate or unnecessary attributes"* a total of twenty-two features were identified for the classification model. All these characteristics were once again related to system resource usage, such as *"total processor time, user processor time, Non-paged / paged memory sizes"* and more. This reinforces the proposed method of visualising propagation, since resource usage is a recurring method of signalling what is and is not malware. Xiao et al (2019) conducted a study with the goal of detecting malware through behaviour graphs. The behaviour graphs were constructed by monitoring and gathering the order in which malware uses system calls, with this you can deem certain orders of system calls benign and others malicious. This study is interesting as well as comprehensive, and provides a good amount of detail in regards to what systems calls malware frequently uses that one could use to get started conducting a similar study. Our study will probably not utilise such a method, but it is worth noting that the system calls are usually in relation to a system resource such as CPU, memory, etc.

Bai et al (2014) used a method of mining format information from malware executables in order to identify characteristics that would make an executable appear malicious. They identified 197 different features and used them to train an algorithm that was able to achieve 99.1% accuracy in the testing environment. Whilst this method of detection is interesting and certainly seems effective, the characteristics it identifies assist in detecting malware before it is able to execute, which is not particularly useful to us when we need the malware to execute and propagate. What is useful however is that some of the characteristics captured seem to relate in some way to how the executable will go about using system resources when executed.

Patel and Tailor (2020) researched a method of not only monitoring a system for ransomware, but also counteracting it. To counteract the ransomware, the system they have produced first identifies the encryption of a protected folder it is explicitly monitoring, once it has it creates a large dummy file for the ransomware to encrypt. Whilst it is spending the time encrypting the file, the system changes its properties to read only so that the ransomware can no longer encrypt the files on the machine. Whilst what they've created is interesting and somewhat unique, one must question the usefulness of it since it doesn't particularly prevent ransomware, but merely gives you an opportunity to prevent it. One must also question the validity of the testing, since the study full well admits that *"we designed the attacks"* and one could argue due to this it doesn't necessarily reflect a real world scenario. Unlike the others, the method this study uses to identify ransomware isn't particularly useful, as the reliability of identifying ransomware based on a monitored folder may prove unreliable when extracting data since the functions used to extract said data may already be blocked by the ransomware before it is counteracted or the preventative measure of locking down the system may create a scenario where we cannot extract the data.

It was also deemed worthwhile to comprehensively research malware analysis methods, particularly in regards to sandboxes, since they will be used to observe the propagation in the first place. The first paper by Afianian et al (2019) is a comprehensive summary of knowledge regarding common methods that malware use to evade detection and analysis. The paper begins begins by acknowledging that modern day defensive and analysis techniques can be *"readily foiled by zero-day fingerprinting techniques or other evasion tactics such as stalling"*. The paper then walks through the different studied methods used to avoid analysis, dividing them into two distinct categories: detection dependent evasion, where the *"goal is to detect its environment to verify if it is a sandbox or not"*; and detection independent evasion, which are methods that do *"not rely on detecting the environment"*. This paper as mentioned previously is very comprehensive and therefore very

useful for learning how malware avoids sandboxing in a very brief and concise manner, even giving explicit examples of what malware tends to look for in regards to environmental details. Chakkaravarthy et al (2019) take a step back from individual machines and take a look at methods malware uses to avoid detection within the network, such as payload fragmentation, session splicing, and more. All of these details whilst interesting were not our key focus on this report, what was is the fact that the paper includes full details on the configuration of their virtual machines including OS, network interface, processor/core/memory, etc. Interestingly enough, this study seems to use NAT based configuration which would mean the infected virtual network still has connection to the host machine, presenting a possible danger.

Miramirkhani et al (2017) also explored the creation of sandbox machines for observing malware and took a deep dive into what small features in a virtual machine might be used by malware writers to detect a virtual environment. Miramirkhani et al (2017) identified different *"Artifacts",* naming *"Direct Artifacts"* which are artifacts created by a users direct interaction, and *"Indirect Artifacts"* which are created by general system activity. Miramirkhani et al (2017) focused only on Indirect Artifacts, stating that whilst Direct Artifacts are qualitatively stronger, they also pose an issue of privacy to investigate, since their research would require the use of participants on their own machines. Miramirkhani et al (2017) identified multiple subcategories of Artifacts, often relating to machine functionality, such as *"System, Disk, Network, Registry, and Browser".* Using these subcategories, Miramirkhani et al (2017) identified and organized their Artifacts, identifying many different metrics that differ greatly between the average user system and a Sandbox environment. Whilst it isnt in our projects aims to associate for context aware malware, it will be useful to explore at a surface level to create the most effective sandbox environment we can to consequently create accurate visualizations.

Sharafaldin et al (2017) explored the creation of a dataset for use with Intrusion Detection Systems. The dataset is completely labeled and contains more than 80 extracted network traffic features for the use in identifying benign or intrusive flows, to gather these they utilized CICFlowMeter, a publicly available software from the Canadian Institute for Cyber Security. The testing of their datasets was fairly comprehensive and consisted of the creation of two network types, the Victim and Attack networks. Sharafaldin et al (2017) further analyzed the dataset to designate different signatures to varying "Attack Profiles" which consist of attacks such as "Brute Force, Heartbleed, Botnet" and so on. This research will be useful if the tool created and utilized is going to be adapted to detect and visualize physical intrusion through an active attacker rather than just an autonomous malware, and is therefore worth keeping in mind, but such plans may be limited by the timed conditions of this project.

Creese et al (2013) undertook a project similar to our own, and attempted to visualize enterprise network attacks and their subsequent potential consequences. The visualization program utilizes traditional network diagram icons and combines them with business process modelling and notation, a disk propagation logic that connects the network and business process and task layer. Creese et als (2013) research is not only important to our project in their results, but also their methodology, since they acknowledge some key principles before and during the creation of their tool regarding exactly what their tool must possess. These design principles will surely be fitted into our own project. In terms of results, Creese et als (2013) work *"demonstrates that it is possible to determine and visualize the potential impact of a cyber attack",* using this sort of visualization technique we can create systems that allow for greater clarity in the communication of risk not only to the technicians maintaining the networks we work on, but also to the decision makers than fund them.

4 Methodology

With the objectives in mind the approach to fulfilling them will be linear in nature. There is a defined order to which the objectives must be completed, not out of preference but out of dependency. Because of this how the project will progress is fairly predictable, the following sub section will provide detail and justifications for this.

4.1 Timeline

The first stage of this project will be the creation of the tool that allows us to extract details from an infected machine to later analyse the propagation. This tool must not only extract the various system details but also potentially extract the time in which the system appears to be infected. There are numerous subsections to this stage, both pre and post tool creation. Before we can begin creating the tool we must first research malware analysis methods concerning the creation of sandbox environments in order to assure that the system we are testing on will not allow the malware to propagate outside of the virtual environment. Furthermore, the methods that malware analysts use to trick malware with a weaker context awareness algorithms into thinking it is in fact in a legitimate users system must be explored. On the other side of the aisle, it will be prudent to research which malware exploit vulnerabilities that allow them to escape virtual environments, at first to avoid them, but if time allows possibly to study them specifically.

The creation of the tool will follow, and then be followed itself by the testing phases. At first the tool will be tested on systems that are not under active attack, then proceed to active attack testing once the extraction functionality is deemed acceptable. This section aligns with the first objective.

The next stage will involve the deployment of the tool to extract data from machines that are experiencing an active attack. Development will technically be continuing during this stage as different malware tend to exhibit different characteristics and therefore the tool may have to be adapted to suit best what the current active attack is. This is in alignment with objective three, the for the sake of transparency this project will include all of the tested characteristics, whereas the ones visualised may only be the ones with significant results.

The final stage will be to analyse, visualise, and draw conclusions from the results, keeping to the requirements outline within objective two. Stage two and three will occur multiple times, each run analysing a different malware. How many times this runs is determined by the amount of time surplus.

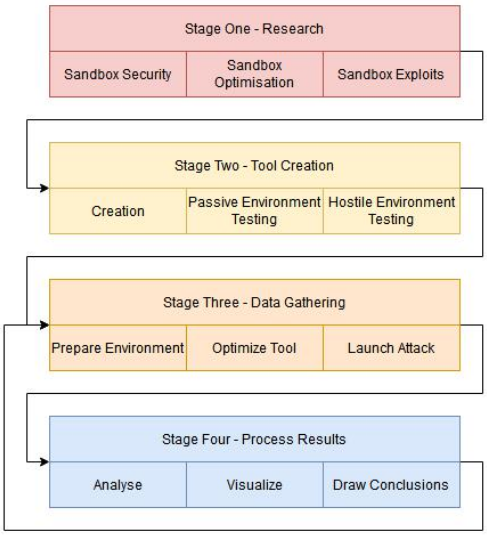


Figure 1 - Project Timeline

4.2 Ethics

There are two acknowledged ethical concerns around this project, one which acts as a requirement and another that must be declared. The requirement being that due to working with legitimate malware strains, the sandbox environment must be as secure as possible because anything less may allow the malware to propagate through the sandbox and into the home/work/school network, there is could pose a threat to any device on said network. Therefore one can consider it an ethical requirement to not only optimise the sandbox environment but also research the malware strains thoroughly for any potential context based exploits that may endanger the external network.

The declaration is that at no point in this project will malware be created, only malware that has already circulated, been studied, and prevented will be studied.

4.3 Testing Setup

In order to gather the appropriate data to visualize malware activity and propagation, we will be using a three tiered approach. The first two methods will be handled by a tool created especially for this project, which will gather metrics from all of the running virtual machines every five seconds and take screenshots of the current view every 30 seconds. This way we can view various possible changes occurring in the background via resource usage and we can also view obvious situation changes in the foreground of the runtime. For example, when running a WannaCry sample we would expect to see a sharp spike in resource usage at first and then the "ransom note" appear after it has finished its encryption operation, using these two methods we can observe both changes. The third method of gathering data will be utilizing WireShark to capture the communications between the machines on the testing network, WireShark will run on a separate node on the network which will be running in promiscuous mode and observing the primary network adaptor. This way we can view any and all activity occurring between the nodes.

In this paper the network setup will be fairly simplistic, being a LAN consisting of four vulnerable Windows 7 virtual machines. The machines will be configured to be as vulnerable as possible since our goal is to observe the way in which the malware propagates, to achieve this each virtual machine will have: an open, password-less, limited encryption fileshare on the network; all ports open; and disabled user access control. As mentioned prior, there will be a fifth node on the network running Ubuntu 20.04 LTS configured to network in promiscuous mode and it will only run WireShark. This network will be running entirely on Oracle Virtualbox's internal network setup in order to prevent propagation from the virtual network setup into the testing environments host machine. The only machine that will have the ability to access the external host will be the Ubuntu machine, which will have a Virtualbox fileshare in order to extract the network activity csv. This fileshare will be disabled and ejected during active testing, and will only be mounted and reactivated after testing has concluded, the infected machines have been reverted to their clean snapshots, and the Ubuntu machine has been checked for signs of infection.

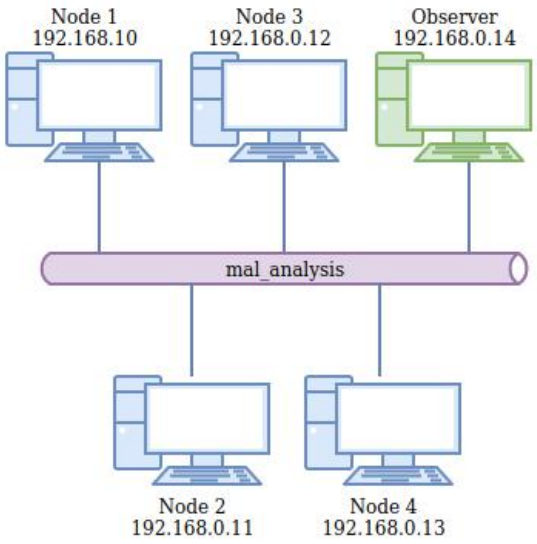


Figure 2 - Network Architecture

4.4 Testing Plan

The testing conducted will generally be quantitative in nature, the goal is to attempt to observe and visualize the activity and propagation of as many samples as possible in order to get a broad overview of the malware landscape and demonstrate the capabilities of our setup as much we can. This is not to say that the finding will be lacking detail however, as each sample will have the three different methods to analyse, and the network log will have many different aspects to be analyzed alone, such as the different protocols.The various samples being used will be acquired from the online malware archive TheZoo.

Each sample will be subject to the same amount of testing, twenty minutes in total: ten minutes of general activity, ten minutes of activity with the sample active starting on Node 1.

Further time constraints will be setup and tear down time, which would take about ten minutes each, making the total amount of time per test around forty minutes given that there are no complications. This amount of time was decided upon to greater allow the use of a quantitative method, but also to standardize the amount of time we can analyse. It is entirely possible that some samples may not activate within the testing time, but this will be explored and noted if it has been determined to be typical or atypical

5 Testing and Visualization

Within this section we will conduct tests using various malware samples and utilise our aforementioned setup to gather appropriate metrics. The first subsection will contain a table briefly detailing every test conducted, the subsections following it will take a deeper look at the tests that appropriately showed propagation resulting in either a fully infected or partially infected network.

5.1 Testing Overview

Test no.	Sample	MD5	Result	Expected(?)
1	WannaCry (Wormless Variant)	84c82835a5d21bbcf75a61706d8ab549	Infection of Node 1, No propagation.	Yes, general metric testing.
2	WannaCry	db349b97c37d22f5ea1d1841e3c89eb4	Propagated, network fully infected.	Yes.
3	CryptoLocker (November 2013)	7f9c454a2e016e533e181d53eba113bc	Infection of Node 1, no encryption, no ransom, no propagation.	Mixed. Excepted no propagation. Didn't expected lack of encryption and ransom. Did infect machine and create startup processes, Cuckoo verified (Appendix 2).
4	CryptoLocker (September 2013)	04fb36199787f2e3e2135611a38321eb	See (3).	See (3).
5	CryptoLocker (January 2014)	829dde7015c32d7d77d8128665390dab / 0246bb54723bd4a49444aa4ca254845a	See (3).	See (3).
6	CryptoWall	47363b94cee907e2b8926c1be61150c7	No visible infection, no visible process, no encryption.	No. Behaved as one might expect of a Crypto-ransomware after being ran, but then effectively did nothing.
7	DirCrypt	N/A	N/A	N/A. TheZoo lists this as being Ransomware yet

				it doesnt appear to be a malware sample, but rather a program for decrypting encrypted files?
8	TeslaCrypt	6d3d62a4cff19b4f2cc7ce9027c33be8	Full infection, files encrypted. No propagation.	Yes. TeslaCrypt is not known to possess a propagation component.
9	TeslaCrypt	6e080aa085293bb9fbdcc9015337d309	Full infection, files encrypted. No propagation.	Yes.
10	TeslaCrypt	209a288c68207d57e0ce6e60ebf60729	Full infection, files encrypted. No propagation.	Yes.
11	Radamant	6152709e741c4d5a5d793d35817b4c3d	Debatable infection, no visible encryption but computer becomes softlocked in WMI error.	No.
12	Radamant	b0625408735468e40f4af9472afcb35a	See (11).	N/A
13	Vipasana	2aea3b217e6a3d08ef684594192cafc8	Full infection, files encrypted, ransom note displayed. No propagation.	Yes. Vipasana is not known to possess a propagation component.
14	Vipasana	a890e2f924dea3cb3e46a95431ffae39	See (13).	See (13).
15	Vipasana	adb5c262ca4f95fee36ae4b9b5d41d45	See (13)	See (13).
16	Locky	b06d9dd17c69ed2ae75d9e40b2631b42	Execution, self deletion, no encryption, no propagation.	Partially. Locky is known to be context aware and therefore may not have executed fully.
17	Petya	af2379cc4d607a45ac44d62135fb7015	Execution, forced restart, encryption, ransomnote. No propagation.	Yes. Original Petya variant did not possess the EternalBlue exploit.
18	Petya	a92f13f3a1b3b39833d3cc336301b713	See (17).	See (17).
19	NotPetya / PetrWrap	e5cc289b0b2b74b8e02f5a7f07867705		
20	NotPetya / PetrWrap	e5cc289b0b2b74b8e02f5a7f0786		

		7705		
21	NotPetya / PetrWrap	e5cc289b0b2b74b8e02f5a7f07867705		

5.2 WannaCry

WannaCry (or WannaCryptor) was the first sample we set out to observe and visualize. WannaCry caused much damage in 2017 before its killswitch was discovered by Marcus Hutchins, with the National Health Service UK (2018) reporting that “80 out of 236 of its trusts were compromised”, “603 primary care and other NHS organisations were infected”, and “1220 pieces of diagnostic devices were disconnected to prevent infection”. Its characteristics made it the perfect choice for being the first test, as it is renown for its ability to quickly and easily propagate using the SMB vulnerability EternalBlue and being generally very loud whilst doing so. The reason behind this is that after it has finished infecting the machine it was launched upon, it then begins probing the network for open SMB ports (445) across the network and if it finds one it will begin taking advantage of the EternalBlue exploit to propagate onto the machine. It will do this for every machine it can reach on a network, and there is even possibilities where the infected machine will propagate again onto already infected machines.

The first metric we decided to analyse was the CPU usage across all nodes in the network. The hypothesis being that since WannaCry is an encrypting ransomware, one should see an obvious increase in CPU usage on a single node, followed by subsequent spikes in CPU usage across various nodes as it propagates. Furthermore, due to the “ransomnote” running and WannaCry constantly scanning the network after infection, one would expect to see a general increase to CPU usage across all machines after infection. As one can see in figure 3, these hypotheses are mostly correct. One can observe the first 10 minutes of dormancy in the test with some erroneous activity from Node 1, possibly being caused startup programs attempting to update, or the node querying its own connection status.Then half way you can observe the very noticable spike in activity by node 1 (blue), this would be the launching of the WannaCry sample. After Node 1’s (blue) activity subsides, you can then observe a spike in activity on Node 3 (green), followed immediately by smaller spikes in activity on Node 2 (red) and Node 4 (orange). These first four spikes are the immediate infection and encryption points, all occuring in about the space of two minutes. Following these, you see a general increase in CPU usage across the board with numerous spikes in activity, which may be caused by WannaCry’s re-infection phenomenon mentioned earlier. Towards the end, activity returns to a stable, yet increased state of usage.

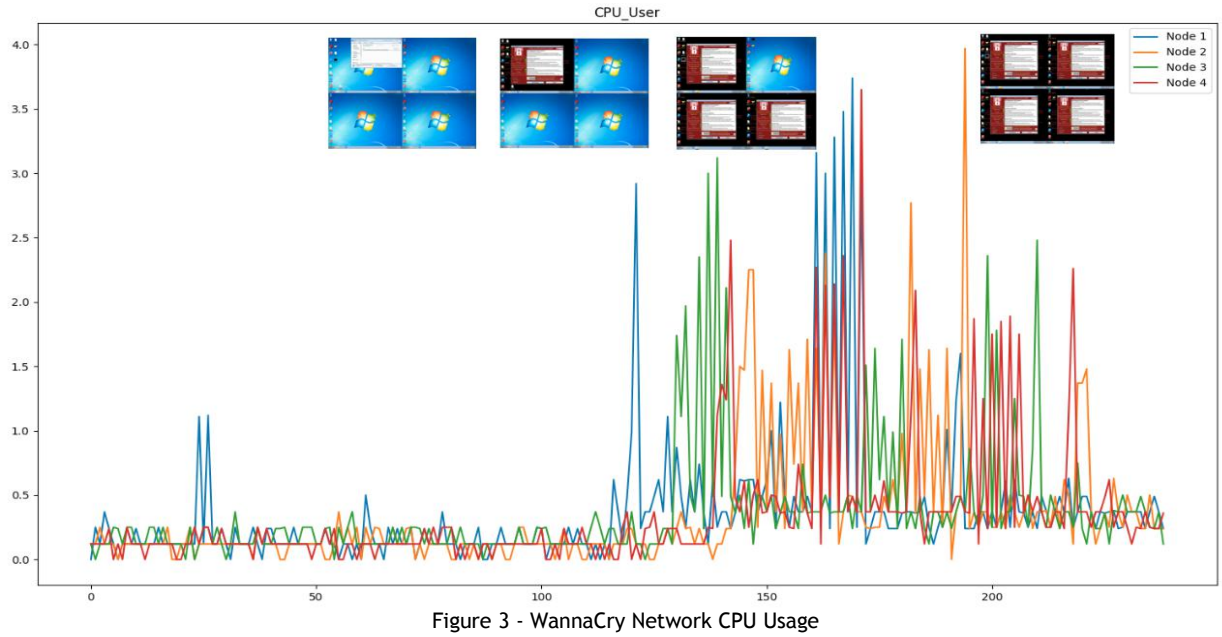


Figure 3 - WannaCry Network CPU Usage

Following this we can look at the network activity of the nodes, which is where things get fairly interesting. The first and most obvious thing to observe when dealing with WannaCry is of course activity involving the SMB protocol, in theory one should expect to see a fairly staggered increase in SMB communication across the nodes.

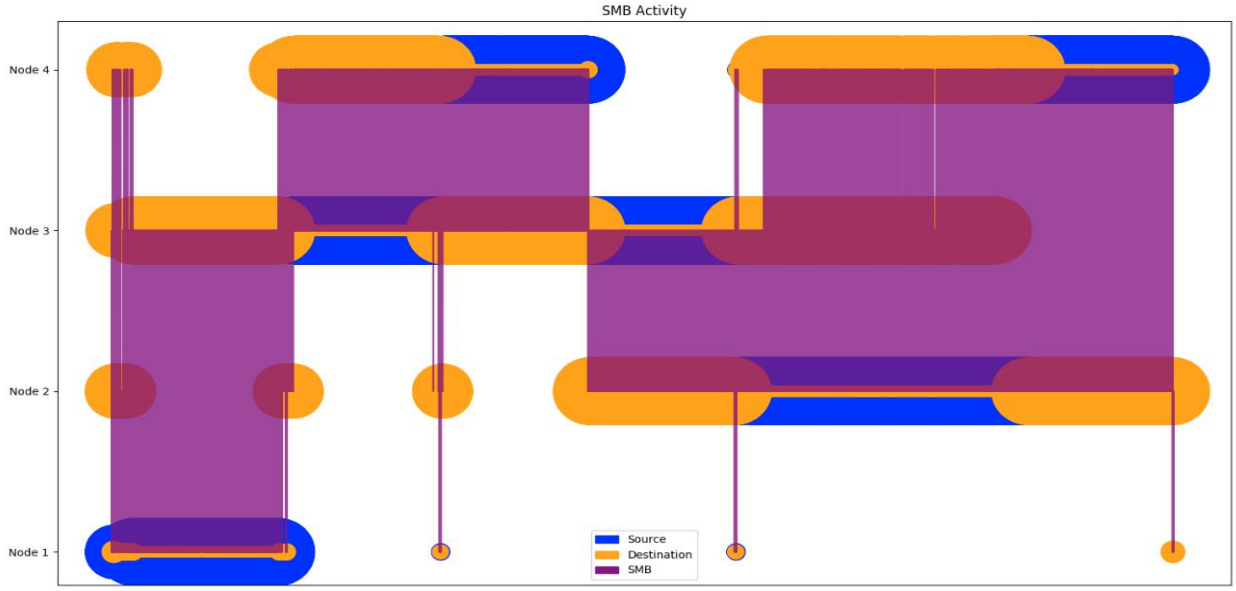


Figure 4 - WannaCry SMB Activity

The four rows each represent an individual node, with the orange colour representing when a node is the source of the SMB activity and the blue the destination of the same such activity, the purple lines represent where the SMB activity is going. The SMB activity of WannaCry is loud and constant, hence why this visualization in particular is also loud and perhaps confusing at first glance. What is happening is the initial infection occurs upon node 1 and propagates to Node 3, WannaCry then propagates from Node 3 to Node 4 and then to Node 2 (second row). All of the later SMB activity can be attributed to the infected machines attempting to reinfect the already infected machines on the network. One interesting thing to take away from this graph is the strange activity of Node 1, which communicates back and forth visibly with Node 3 for a time, and then ceases most activity after aside from the three points where it has a back and forth with Node 3 and Node 2. Why Node 1 doesn't display the continuous SMB activity akin to the other nodes is hard to say, could the initial infection of WannaCry only propagate in such a way once, and then leave the subsequent propagations to infect the rest of the network in order to project its entry point? One would believe this to be true if it were not for the fact that WannaCry even outside of its propagation is loud due to its nature as a ransomware.

The next protocol worth looking at is TCP, figure 5. At a glance one might notice that the latter half of the graph looks similar to that of the SMB graph distribution.

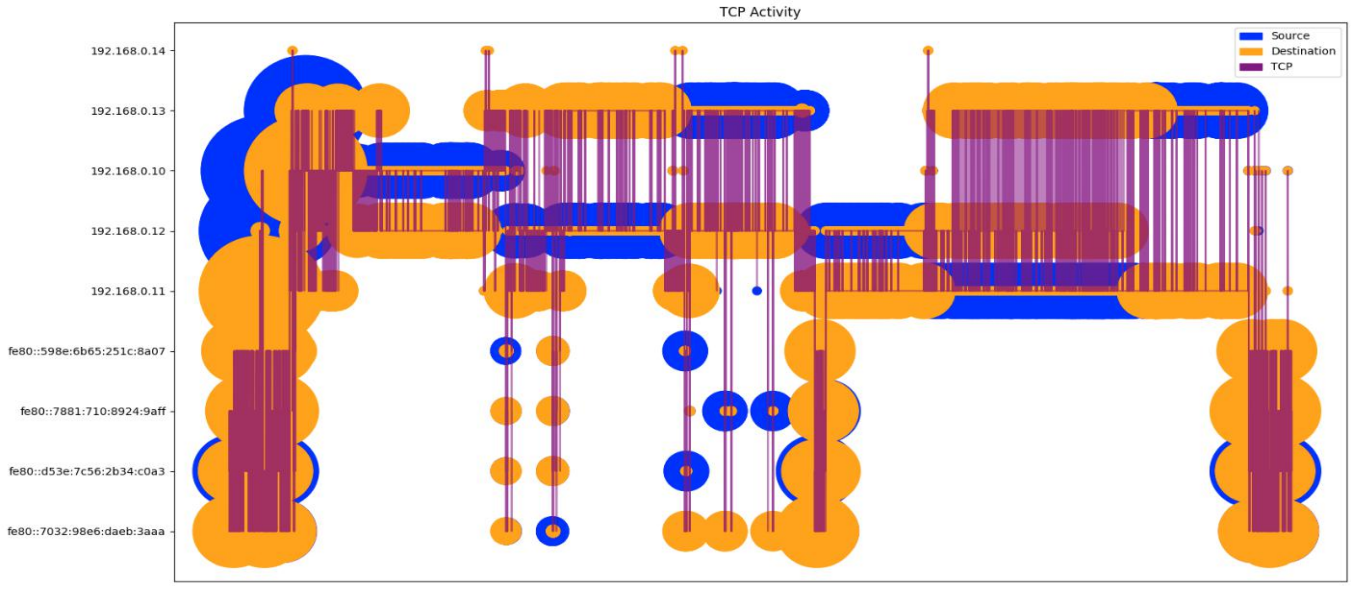


Figure 5 - WannaCry TCP Activity

This is to be expected, since the SMB protocol runs atop the TCP protocol on port 445, which this version of WannaCry is exploiting. Therefore the former half of the graph will be showing simply just TCP activity in general, and the latter half will be showing the general activity and the activity with the SMB protocol. However, the question therein lies if one can distinguish the difference between the two here, the answer being a rather resounding negative. The only obvious way of noticing the WannaCry activity is to have prior knowledge of the SMB activity, therefore this visualization on its own is not a reliable method of determining if propagation has occurred or not.

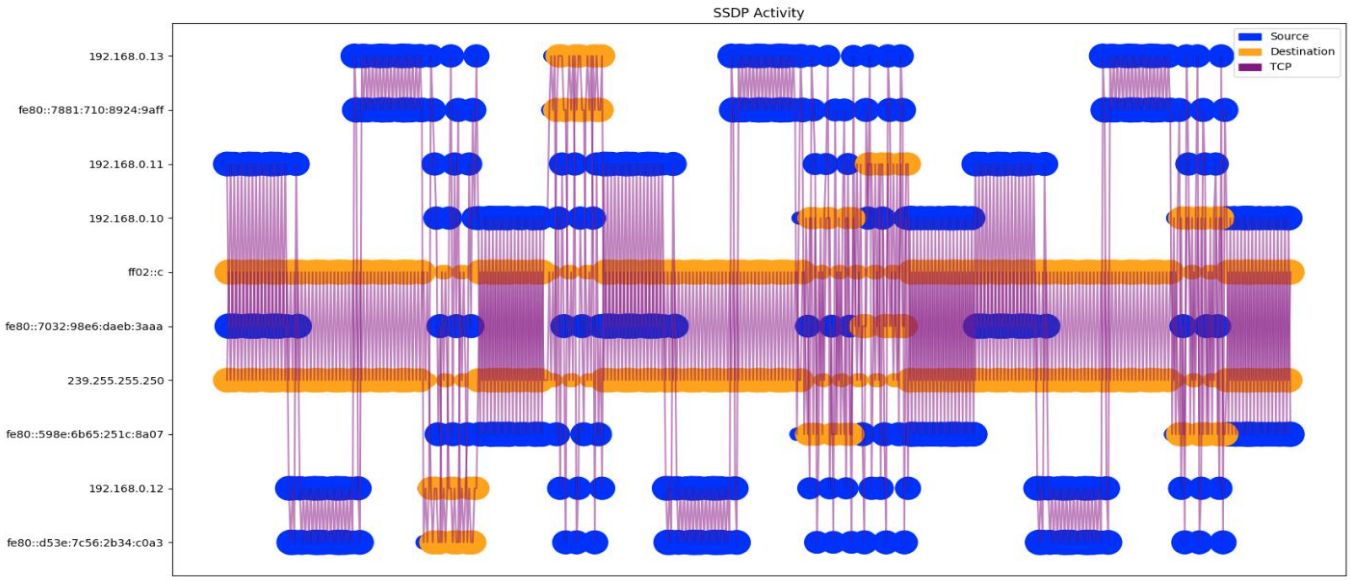


Figure 6 - WannaCry SSDP Activity

Nothing of particular note appears in the visualization of SSDP in a network infected with WannaCry, other than that it appears to consistently work as normal with very little deviations. One cannot pinpoint the exact point where propagation may be occurring or not. This is to be expected, since WannaCry is not known to utilize SSDP to any extent.

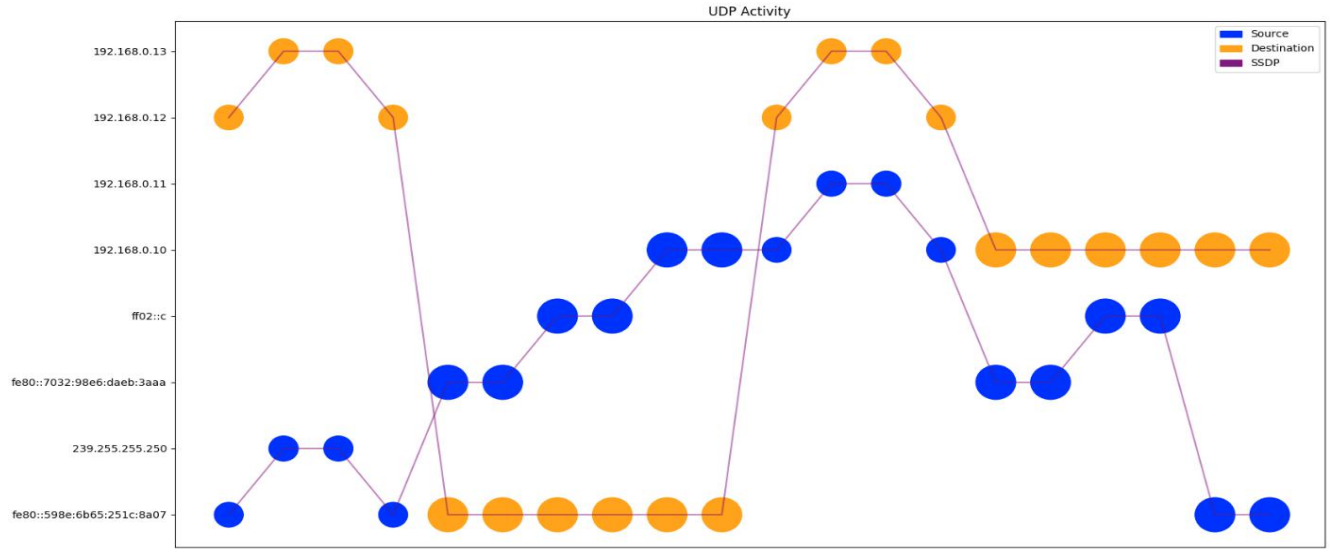


Figure 7 - WannaCry UDP Activity

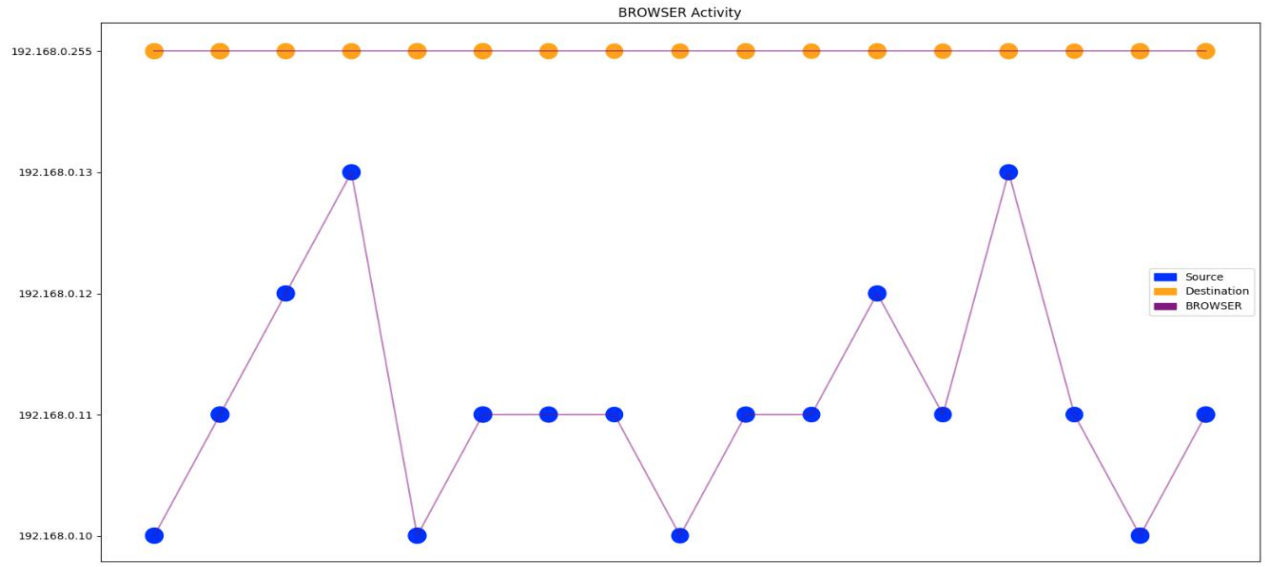


Figure 8 - WannaCry BROWSER Activity

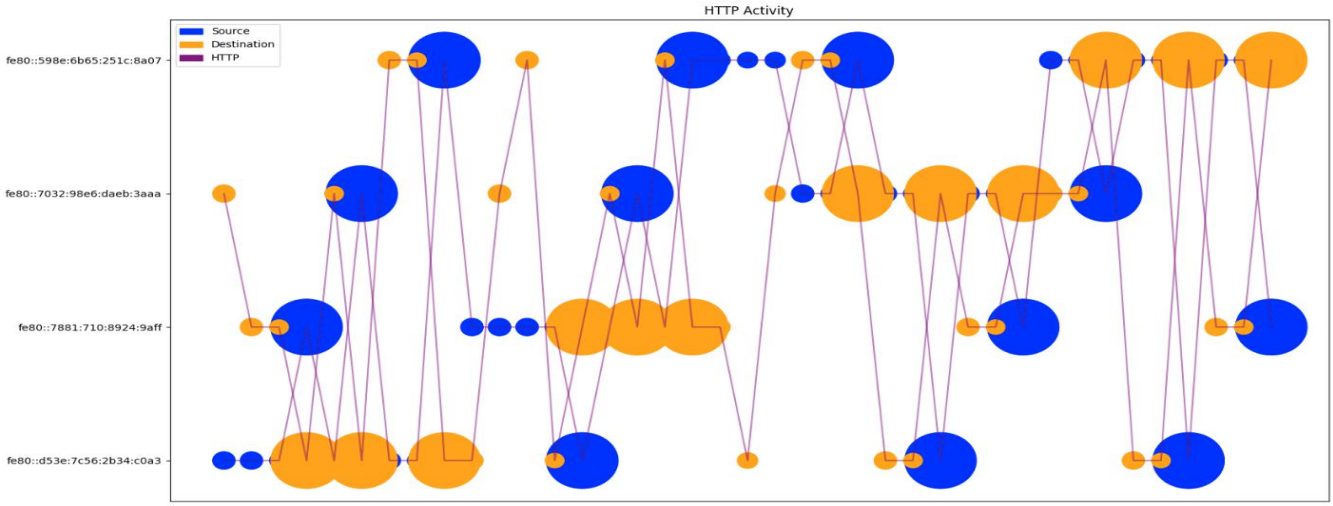


Figure 9 - WannaCry HTTP Activity

Just as with SSDP, there is no particularly obvious activity within UDP, BROWSER, or HTTP that allows us to observe propagation concerning WannaCry. All of these protocols appear to continue acting consistently even after infection.

Then finally we come to the screenshot gathering of our setup. The interesting thing here is that we can roughly estimate the time it takes to go from a seemingly clean network setup to being fully infected, happening in roughly two minutes and thirty seconds.

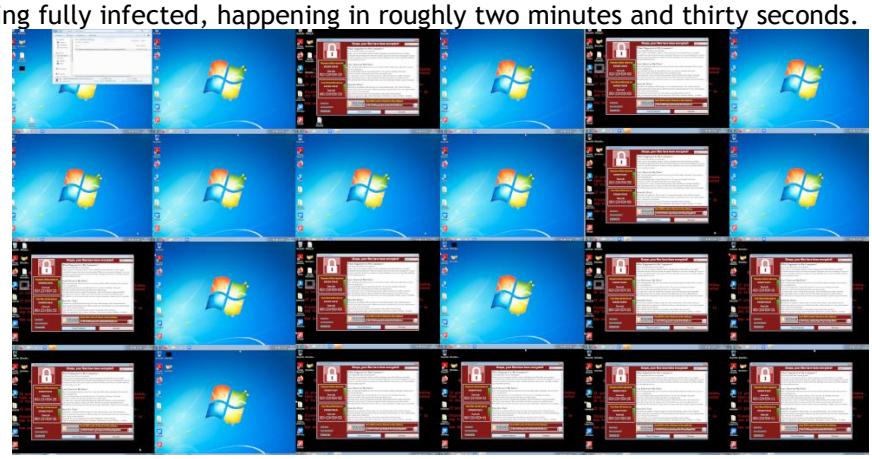


Figure 10 - WannaCry Network Clean to Infected

The final interesting thing that the screenshot functionality reveals to us about the nature of WannaCry propagation is something that has been mentioned at various points in this section, that being the tendency for WannaCry to reinfect machines it has already infected. This is proved when you look at the last screenshot taken in this test.

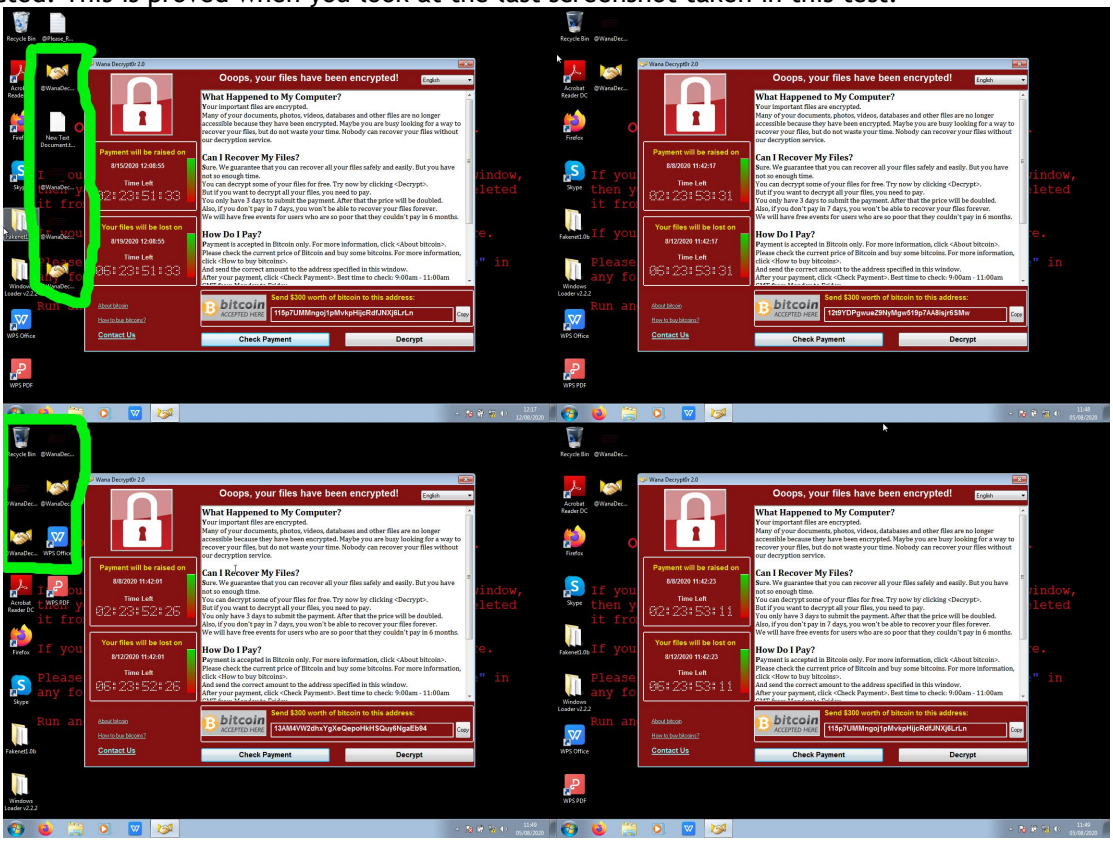


Figure 11 - WannaCry possible reinfection

Both of the nodes that were infected first have a second copy of the WannaCry executable on its desktop, aswell as an extra background which accompanies the executable. Full screen shots of each of these machines can be viewed in Appendix 1.

For further analysis of characteristics, a Cuckoo analysis was also performed on our WannaCry sample in order to reveal characteristics that our setup was incapable of detecting. Considering the sample ran and infected all four machines, what was not expected of Cuckoo to return was the possibility that WannaCry was actually infact contextually aware, this is evidenced by the fact that at some point in execution Cuckoo observed WannaCry specifically looking for the Cuckoo agent.py in the startup directory.

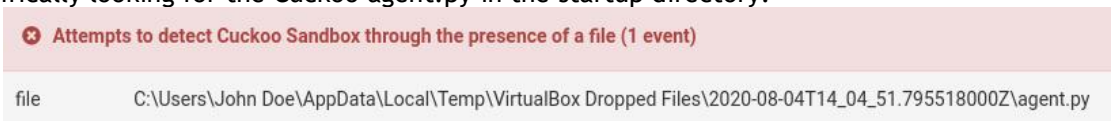


Figure 12 - WannaCry looking for Cuckoo

Despite this, the sample still infects and propagates across our LAN setup, and even more surprisingly, it still infects and encrypts the Cuckoo setup. This tells us that even though the WannaCry sample was developed to detect being in a possible sandbox environment, it doesn't act on it once detected. Another possible conclusion is that it does infact act on it, but in such a way that it encrypts the agent.py first and then moves on to the rest of the files in order to limit the amount of analysis and possibly stop the sandbox environment from detecting the EternalBlue SMB exploit.

5.3 CryptoLocker

Within this study we tested three variants of CryptoLocker, appearing in September 2013, November 2013, and January 2014. Each of the tests ended inconclusively with the payload of the CryptoLocker ransomwares not deploying within the twenty minute time frames. Following further research a unique reason for such behaviour became apparent, after being launched the CryptoLocker ransomware creates a process visible in task manager;

inserts itself into the startup programs; and then deletes the original executable that it used to create said processes. This was all observed, and the process of the ransomwares can be observed in task manager *(Figures 13 & 14)*.

However this was as far as the ransomware proceeded, no propagation was expected because CryptoLocker is not known to propagate internally in local area networks, but what was not expected was the lack of encryption, which is where our unique situation unfolds. After deleting the injection method, CryptoLocker then has to communicate with its Command and Control server, which is now known as the GameOver ZeuS server. When it communicates with the server, as well as registering it as part of its botnet, the server also generates the 2048bit RSA keys which it sends back to the clientside CryptoLocker in order to allow it to encrypt all of the files.

This presents two such interesting situations, the first and most obvious being that the GameOver ZeuS server was taken down in Operation Tovar in 2014, meaning that any communication with the server is now impossible. But if hypothetically the server was still available, the ransomware still would not be able to run because our testing setup works on an internal network structure, meaning it has no ability to communicate with the server even if it was available. Overall, this presents a previously unforeseen difficulty, that malware which requires external influence in order to perform its entire function will not be able to function fully in our setup. In essence, a flaw of the system created may be that it is only useful for testing dropper type malwares and not downloader types or botnets.

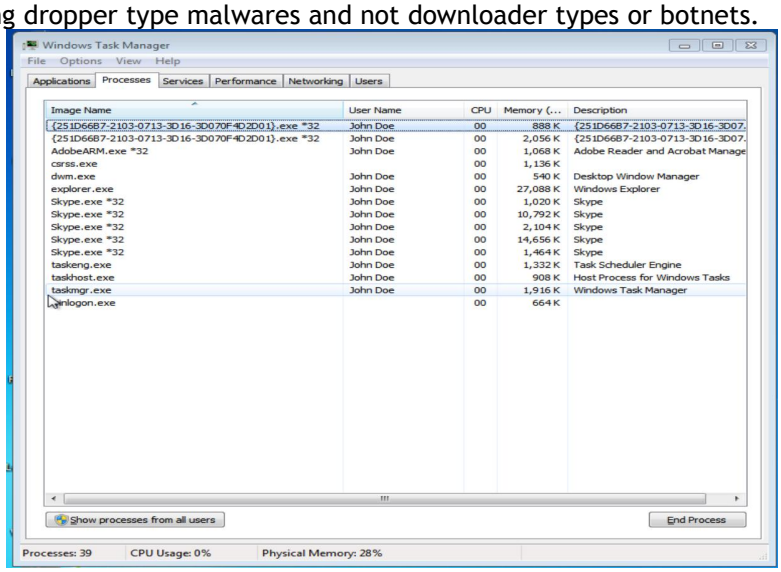


Figure 13 - September Variant Process

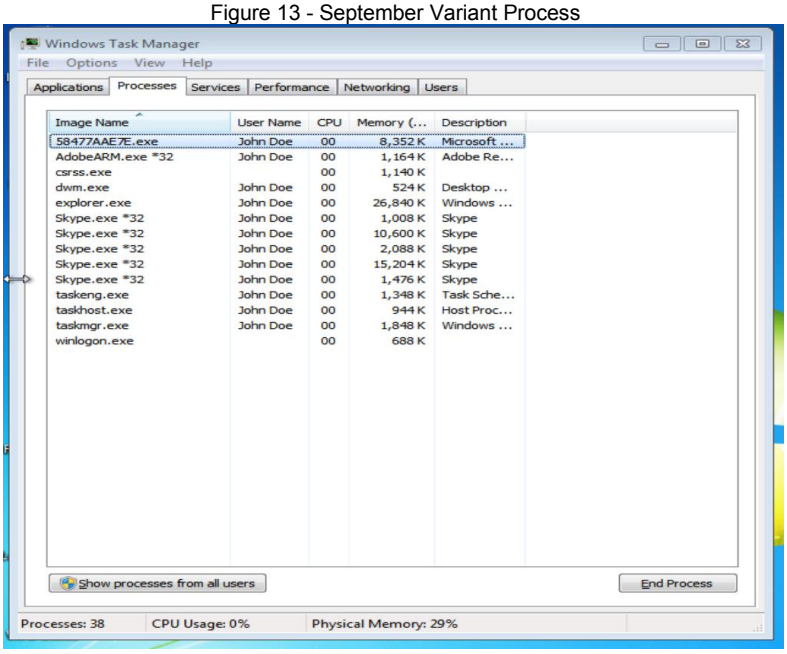


Figure 14 - CryptoLocker January 2014 Variant Process

5.4 Locky

Unlike CryptoLocker which presented an unforeseen difficulty in the testing of malware, Locky presented a predicted one. In essence, Locky behaves similarly to CryptoLocker in that after it is executed it performs self deletion. Where it differs is that Locky is known to be context aware, meaning it has multiple methods of identifying whether or not it is in fact running in a virtual environment or not. In our test, the sample performed observable self deletion and then nothing else happened, all of the reported infection identifiers were explored in registry and system files with none to be found. Therefore it is believed that this is the first case during testing where a sample was able to identify it was within a virtual machine and decided not to execute. There was a previous case where the WannaCry has a form of contextual awareness, according to a Cuckoo analysis in which it was spotted looking specifically for the Cuckoo agent in the startup directory, however it is assumed that this was a basic implementation of contextual awareness in order to avoid known sandboxes and is therefore able to execute in our setup because it is not looking for the telltale signs of a virtual machine, but only the signs of the Cuckoo setup (Figure 12).

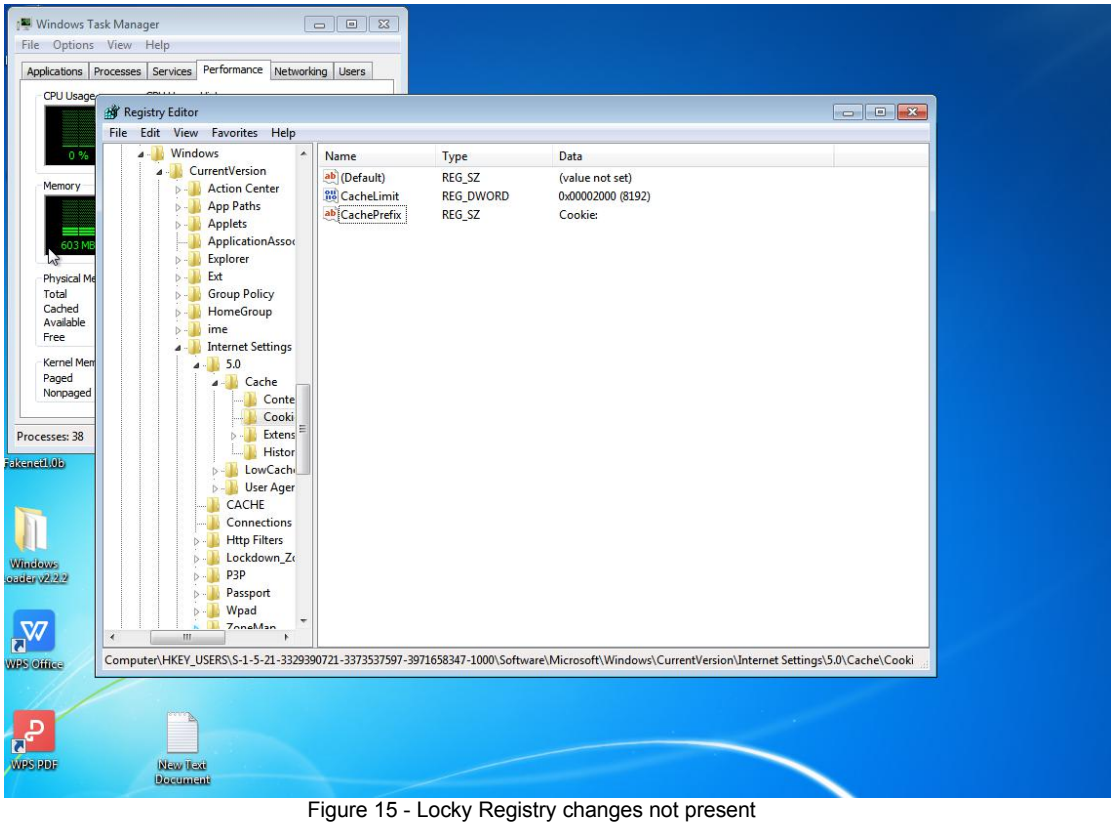


Figure 15 - Locky Registry changes not present

6 Discussion

(Here I will discuss the relevancy of the results in relation to the objectives and the overall question.)

7 Conclusions

7.1 Evaluation

(Things that are cool)
(Things I could've done better)

7.2 Future Work

(Theres alot of it)

8 References

Afianian, A., Niksefat, S., Sadeghiyan, B. and Baptiste, D. (2019) Malware Dynamic Analysis Evasion Techniques: A Survey. *Acm Computing Surveys* [online]. 52 (6), pp. 126:1-126:28. [Accessed 26 May 2020].

Bai, J., Wang, J. and Zou, G. (2014) A Malware Detection Scheme Based on Mining Format Information. *Scientific World Journal* [online]. 2014 [Accessed 01 June 2020].

Chakkaravarthy, S.S., Sangeetha, D. and Vaidehi, V. (2019) A Survey on Malware Analysis and Mitigation Techniques. *Comouter Science Review* [online]. 32, pp. 1-23. [Accessed 26 May 2020].

Creese, S., Goldsmith, M., Moffat, N., Happa, J. and Agrafiotis, I. (2013) Cybervis: Visualizing the Potential Impact of Cyber Attacks on the Wider Enterprise. Ieee [online]. [Accessed 21 July 2020].

Gove, R. and Deason, L. (2018) Visualizing Automatically Detected Periodic Network Activity. *2018 Ieee Symposium on Visualization For Cyber Security* [online]. [Accessed 02 March 2020].

Hosseini, S. and Azgomi, M.A. (2016) A Model For Malware Propagation in Scale-free Networks Based on Rumor Spreading Process. *Computer Networks* [online]. 108, pp. 97-107. [Accessed 01 June 2020].

Mills, A., Spyridopoulos, T. and Legg, P. (No date) Efficient and Interpretable Real-time Malware Detection Using Random Forest. *(No Place)* [online]. [Accessed 16 February 2020].

Miramirkhani, N., Appini, M.P., Nikiforakis, N. and Polychronakis, M. (2017) Spotless Sandboxes: Evading Malware Analysis Systems Using Wear-and-tear Artifacts. Ieee Symposium on Security and Privacy [online]. [Accessed 21 July 2020].

National Health Service UK (2018) *Lessons learned review of the WannaCry Ransomware Cyber Attack*. Skipton House: National Health Service UK.

Patel, A. and Tailor, J. (2020) A Malicious Activity Monitoring Mechanism to Detect and Prevent Ransomware. *Computer Fraud and Security* [online]., pp. 14-19. [Accessed 16 February 2020].

Rhode, M., Burnap, P. and Jones, K. (2018) Early Stage Malware Prediction Using Recurrent Neural Networks. *Computers and Security* [online]. 77, pp. 578-594. [Accessed 15 February 2020].

Sharafaldin, I., Lashkari, A.H. and Ghorbani, A.A. (2017) Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *Proceedings of the 4th Internation Conference on Information Systems Security and Privacy* [online]., pp. 108-116. [Accessed 21 July 2020].

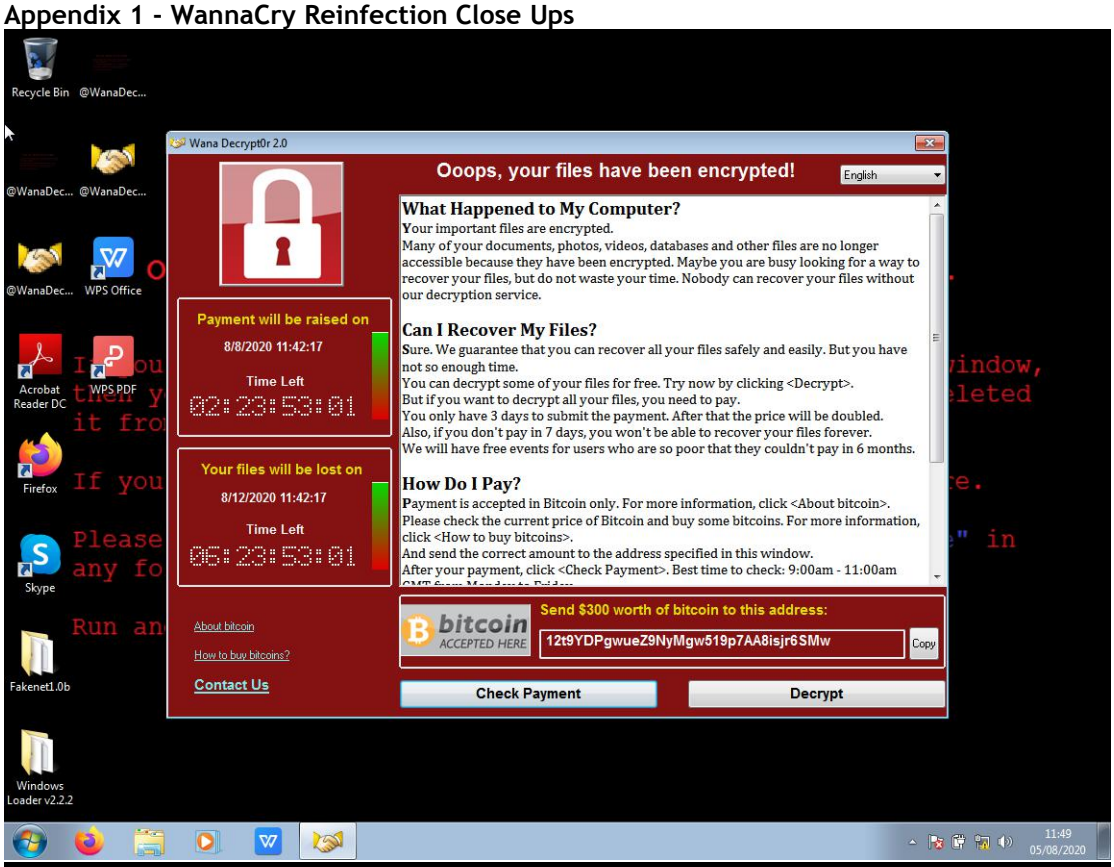
Xiao, F., Lin, Z., Sun, Y. and Ma, Y. (2019) Malware Detection Based on Deep Learning of Behaviour Graphs. *Mathematical Problems in Engineering* [online]. 2019 [Accessed 01 June 2020].

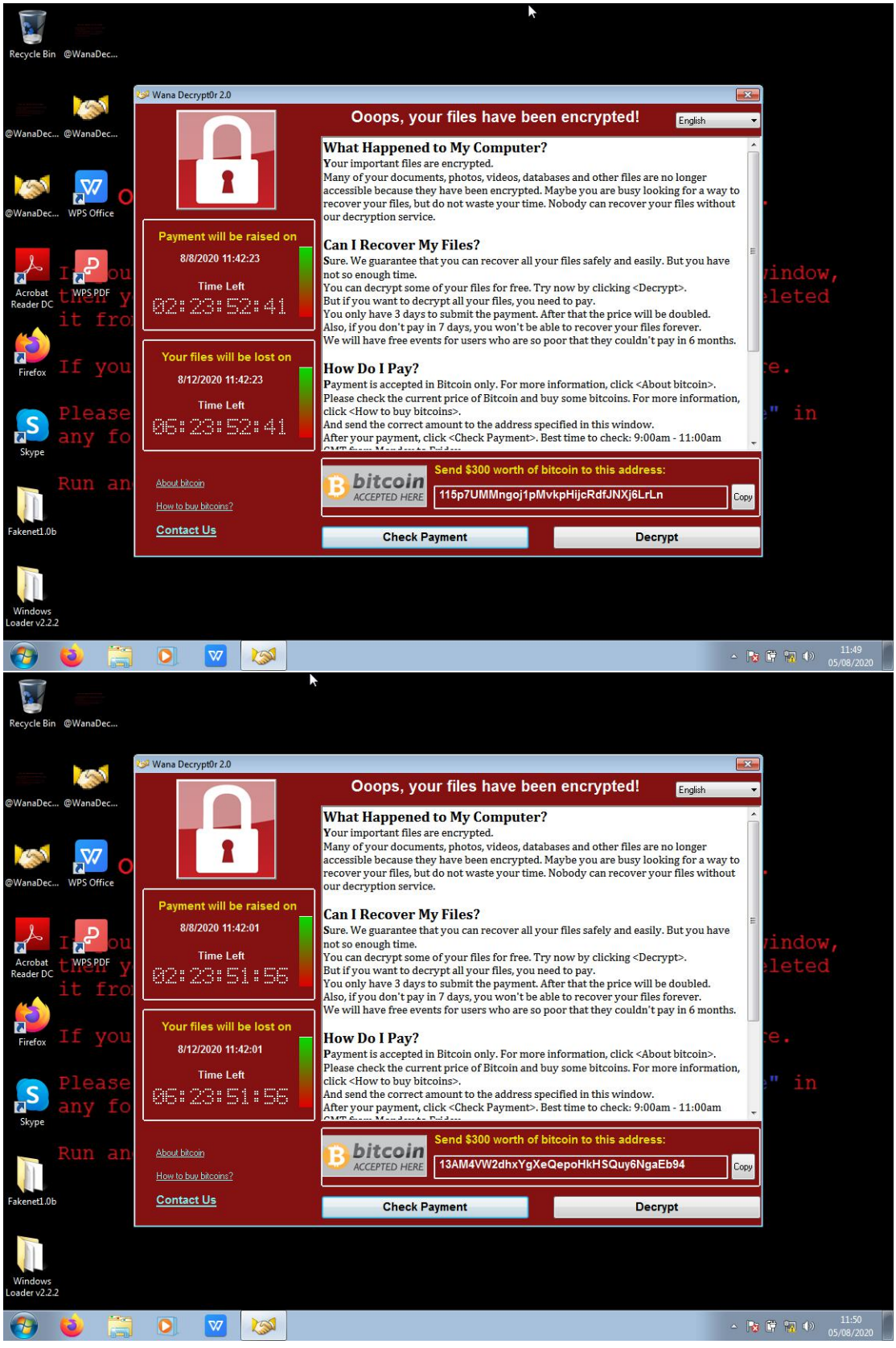
Yu, S., Gu, G., Barnawi, A., Guo, S. and Stojmenovic, I. (2015) Malware Propagation in Large-scale Networks. *Ieee Transactions on Knowledge and Data Engineering* [online]. 27 (1), pp. 170-179. [Accessed 14 February 2020].

Zhuo, W and Nadjin, Y (2012) MalwareVis: Entity-based Visualization of Malware Network Traces. *Acm International Conference Proceeding Series* [online]., pp. 41-47. [Accessed 15 February 2020].

9 Appendices

Appendix 1 - WannaCry Reinfection Close Ups




Wana Decrypt0r 2.0
Ooops, your files have been encrypted!
English
What Happened to My Computer?
Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.
Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.
How Do I Pay?
Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am
Payment will be raised on
8/8/2020 11:42:23
Time Left
02:23:52:41
Your files will be lost on
8/12/2020 11:42:23
Time Left
06:23:52:41
About bitcoin
How to buy bitcoins?
Contact Us
Send $300 worth of bitcoin to this address:
115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
Copy
Check Payment
Decrypt
Recycle Bin
@WanaDec...
WPS Office
Acrobat Reader DC
WPS PDF
Firefox
Skype
Fakenet1.0b
Windows Loader v2.2.2
11:49
05/08/2020

Wana Decrypt0r 2.0
Ooops, your files have been encrypted!
English
What Happened to My Computer?
Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.
Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.
How Do I Pay?
Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am
Payment will be raised on
8/8/2020 11:42:01
Time Left
02:23:51:56
Your files will be lost on
8/12/2020 11:42:01
Time Left
06:23:51:56
About bitcoin
How to buy bitcoins?
Contact Us
Send $300 worth of bitcoin to this address:
13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
Copy
Check Payment
Decrypt
Recycle Bin
@WanaDec...
WPS Office
Acrobat Reader DC
WPS PDF
Firefox
Skype
Fakenet1.0b
Windows Loader v2.2.2
11:50
05/08/2020

Appendix 2 - CryptoLocker (November 2013) Infection