Visualisation of Malware Propagation Across Local Area Networks

Jacob John Williams
University of the West of England

Abstract

Malware has always been a prevalent threat to systems in all sectors and recent years has seen no delay to this trend. As the world becomes more and more technologically focused, the value of the systems and the data within them increases also, and as a result attacking these systems or stealing the data within becomes more and more lucrative too. The consequence of such a growth is the creation of malware with ever more complex and dangerous payloads, malware that can self replicate, and exploit vulnerabilities that allow it propagate itself across the world and wreak untold havoc. The aim of this study is to contribute towards the infinite game of cat and mouse that surrounds malware, we seek to create a tool that will allow for the observation, study, and ultimately visualisation of malware activity and propagation across local area networks.

Aims and Objectives

The aims are as follows:

1. The creation of a tool that allows one to create virtual network sandboxes to allow for the safe execution and observation of malware.
2. To adapt the aforementioned tool to allow for the gathering of various relevant metric data from the systems running on the virtual network.
3. To visualise the activity observed on the virtual network to seek trends and similar characteristics in different malware samples. It would then be possible to use these visualisations in:
    a) Network security decision making.
    b) Informing like-minded researchers.
    c) Showing those new to the study of malware what samples act like from a behavioral standpoint.

Development

The tool built possesses three different layers of operation, working in an adapted web app fashion. The front end is written in HTML and Javascript, here one can see all of the virtual machines that one could launch from the machine the tool is running on, upon launching the virtual machines one can configure them to your hearts content. Furthermore, one has the option to enable the gathering of metrics and enable or disable their querying. All of the functions on the front end are monitored through javascript AJAX calls, which are passed through to a Python Flask backend to perform the jobs required. The Python Flask backend then utilises system calls to make use of a tool called VboxManage developed by Oracle, and is essentially a command line variation of the typical VirtualBox program. If one selects to list all virtual machines currently available, the front end acknowledges the button press and passes it to the backend, which will use a vboxmanage command to query which machines are currently available on the system. The available machines are then passed back through to the front end where they are listed on the dashboard.

Whilst the machines are running, the tool is using AJAX to prompt queries at particular intervals, the results of said queries are stored in a CSV file that is created when one enables the gathering of metrics. As well as gathering the metrics, the tool also uses a vboxmanage functionality that allows one to take a screenshot of a queried virtual machine. It queries metrics every five seconds, and it takes screenshots of the virtual machines running every thirty seconds. For gathering an additional dimension of data, namely network data, one can use a virtual machine created that is of a simple design that I call a “net_node”. In short, it is an Ubuntu virtual machine that has high enough permissions to view all of the traffic taking place on the network, but itself does not accept or request connections, it exists merely to observe and never to act. It does this by working in Virtualboxs’ promiscuous mode whilst also having been configured machine side to be mistrustful of any incoming connections.

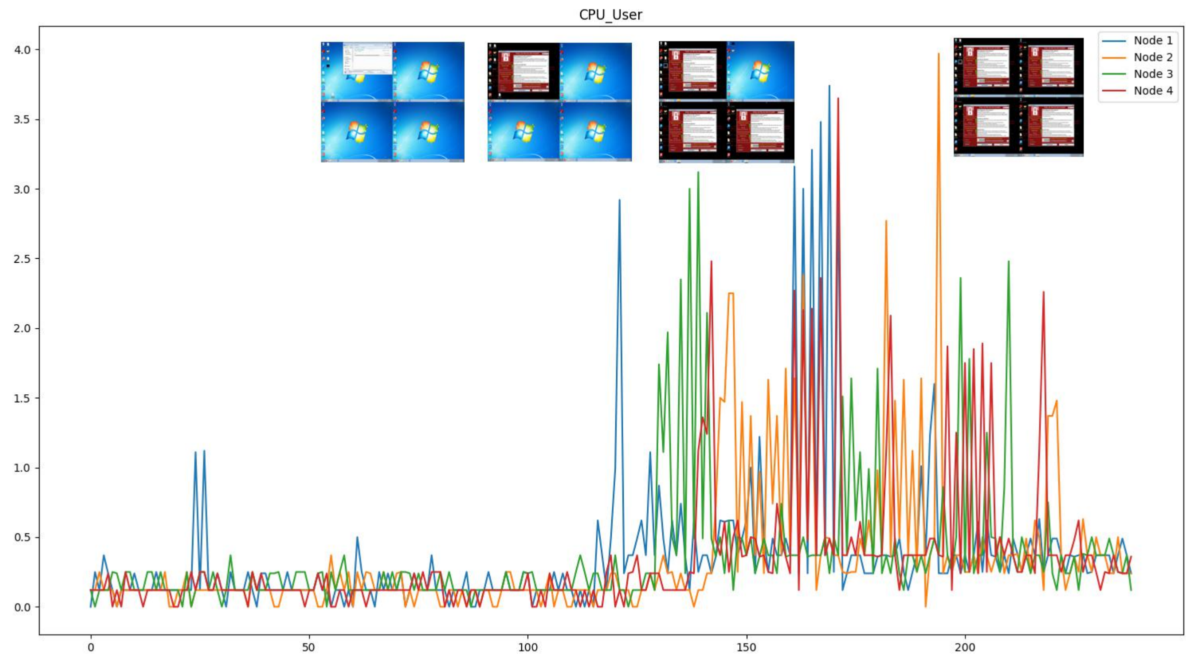
The other virtual machines used in this development and study are Windows 7 based, it was decided upon to use Windows 7 simply because Windows XP is usually the go to for this sort of sandboxing, and therefore using Windows 7 would yield us some fairly unique results and not have us unknowingly repeat another persons work. The Windows 7 machines were designed to be as open and as vulnerable as possible without intentionally installing backdoors. The machines are configured to automatically accept any incoming connection request, they have user access control disable, they have no firewall or anitvirus, and finally they were not updated past the initial installation service pack.

Testing

Numerous rounds of testing were conducted using this tool, during development I mostly used a wormless variant of WannaCry to test the systems whilst also not creating too much trouble for myself in regards to setup and tear down time. After development had reaches a point where all the metrics were being gathered, I moved on the active threat testing, where I would run various samples of malware acquired through online repositories called “Malware Zoos”. This active threat testing was done to accompany my masters dissertation, and the data gathered was used to assess the question of whether data visualisation could be used for assistance in decision making. By the end, I had tested 50 different malware samples.

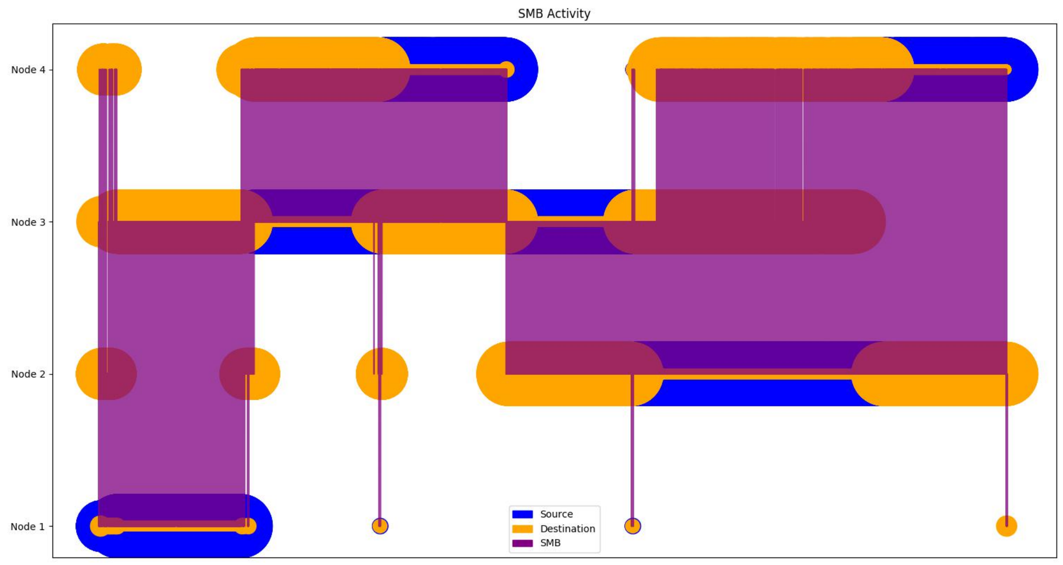
In my dissertation I used a very particular method of testing to view obvious changes in machine behaviour, achieved by at first running the virtual network for around ten minutes. During these ten minutes the machines would not be used at all, the metrics recorded would be purely idle. At the ten minute mark the sample would be activated, and then the next ten minutes would be spent recording metrics from a virtual network with an active threat. The only exception to this was NotPetya / PetrWrap, which had a unique testing case due to its unique behaviour of delaying its own execution and propagation.

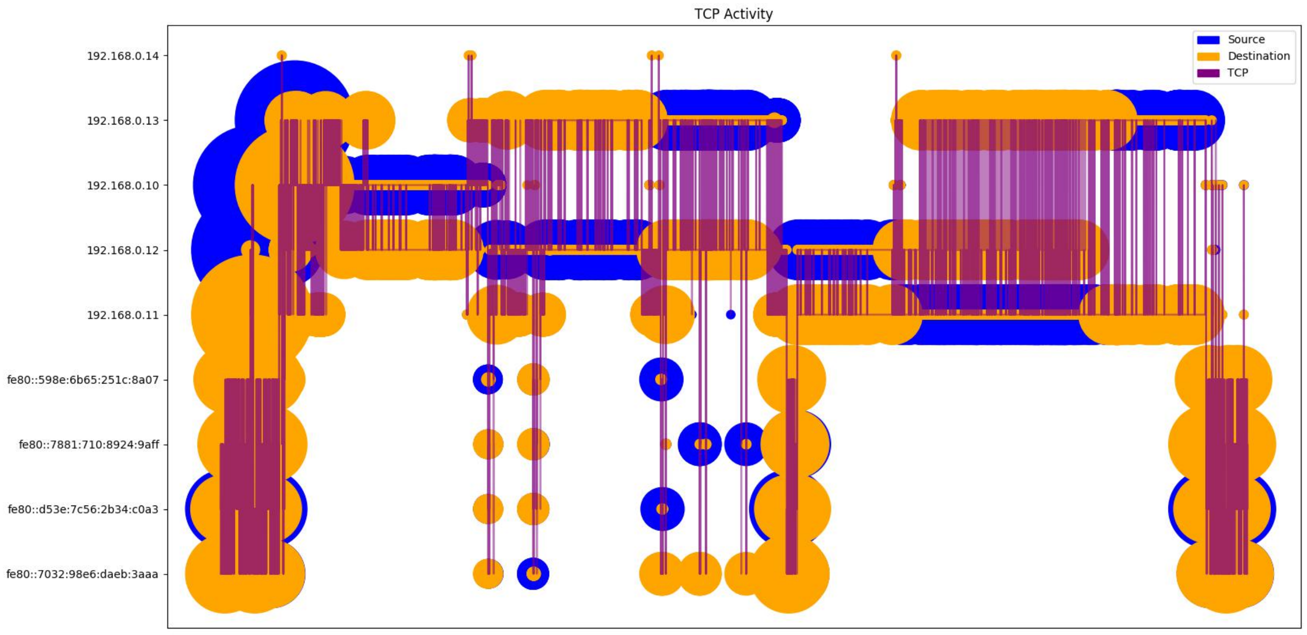
In regards to this report, there is far to much data in those fifty tests to effectively communicate all fifty of the findings, therefore here we will explore a run of a sample of WannaCry, the ransomware behind the attack and subsequent shutdown of various NHS services in the year 2016. WannaCry is a ransomware cryptoworm that tends to propgate across the air gap through phishing emails, once in a network however it has some very interesting abilities. WannaCry exploits a vulnerability found by the United States National Security Agency and subsequently leaked by a group called the Shadowbrokers, this vulnerability is known as Eternalblue. WannaCry uses Eternalblue to propagate across networks without having to explicitly be moved from host to host via infected usb drive or by email, Eternalblue exploits a vulnerability in the SMBv1 protocol that allows an attacker to overflow the SMB buffer and leak into main memory, from here WannaCry is able to install itself on systems without human intervention.



Above is the CPU data across a four machine virtual network, one can see the obvious difference in performance that occurs when the active threat is introduced into the environment, I consider the initial spike in Node 1 activity towards the beginning to be erroneous and can rather easily be ignored due to the fact there is not particular activity captured around that time that would have caused it. The interesting thing about the distribution of CPU data is that one can essentially see how WannaCry propagated and in what order, the sample was introduced on Node 1, it then propagated to node 3, then node 4, and then finally infected node 2. From this distribution is is hard to say there is any rhyme or reason to the order in which it propagates, one would think it would propagate linearly, being introduced on Node 1 then infecting node 2, and so on. But this is not the case, which is interesting, were Node 3 and 4 simply quicker to be infected despite being attacked after Node 2? These questions require a deeper analysis of network traffic.

As mentioned in the initial description, WannaCry utilises EternalBlue, which exploits a vulnerability in SMBv1, therefore if we want to see exactly how WannaCry propagated we can take a look at activity that occurred using SMB on the network, the graph below is exactly that. You can see that the “flow” of SMB activity follows a similar trend to the CPU data, the initial infection takes place on Node 1, it propagates to Node 3, Node 4, and then finally Node 2. But stepping back from the obvious “flow” one can see that there is copious amount of activity taking place all across the network from all nodes. Before Node 3 and 4 have their obvious flow of SMB data to Node 2, one can see Node 2 exhibiting a similarly sized amount of traffic directed at Node 3. It would be wise to conclude that Node 2 was infact infected at a similar time to Nodes 3 and 4 despite the ransom note appearing after both of them, this is further evidenced by the fact that in the CPU data one can see an initial smaller spike in activity on node 2 around the time of Node 3 and 4’s spike in activity, but before Node 2 had its peak of activity. This leads us to the conclusion that the order in which infection through propagation occurs is unpredictable to a degree, there are numerous compounding factors that influence what becomes infected first.





Above is the TCP activity that occurs across the entire twenty minute test, and it is difficult to discern any difference between normal and malicious activity, which makes sense to a degree, since TCP is oft the most used protocol for connectivity to the point there the protocol that Eternalblue exploits, SMBv1, runs atop it. There are numerous other visualisations that came from this test, which were created to see if we can observe any malicious activity among them, but more often than not this was inconclusive.

Evaluation

Throughout this project I have achieved a plethora of different things. Firstly and most obviously, I have made the first steps in the creation of a tool that allows for the observation and visualisation of malware characteristics and propagation. Secondly, I have been able to explore many different skills throughout all of the project, starting with advancing the knowledge I had on Python by using libraries I had not used before, and the creation of a Python backend to a HTML frontend was an architecture I had not experimented with previously. Furthermore, I was able to learn much surrounding the art of visualisation, mostly researching how to most accurately present my findings in a way that can be interpreted by a broader range of people with different skill sets. Finally, before this project I had not used the command line version of VirtualBox, vboxmanage, and being able to experiment and develop with this tool was great for expanding my knowledge of virtual machines and their working parts.

As aforementioned, using the tool I created with this project I was able to create a masters level dissertation on "Can Visualising Malware Propagation Across Local Area Networks Assist in Security Decision Making?" where I tested 50 different samples of malware observing them as they worked and analysing the logs created at the end of each test. The end results were very clear, that there are very few samples that propagate through flaws inherent in a base Windows system. EternalBlue is the most prevalent, but otherwise most malware exploits vulnerabilities in certain software rather than flaws in the core operating system. Furthermore, our testing methodology was brought forward for questioning once I was testing NotPetya, since it delays its deployment it caused an erroneous test first of all, and I had to deviate from the standard I had set myself to associate for this behaviour. The concluding point from all this is that each malware functions differently, and in the creation of this tool and the testing conducted for the dissertation work I operated mostly on a quantitative basis by trying to test as many different malware as possible. But in reality, certain malware require a more qualitative approach since they are often tailored to exploit certain vulnerabilities that are simply not present in the base Windows operating system.

Future Work

Despite the worked conducted I am no where near saying that my work is finished, I intend to continue developing this tool into the future to craft it into the true vision I have for it. This will mean changing many different things. First of all I want to create a system wherein the user can create networks at a will using the dashboard as an interface and the vboxmanage backend to perform the task. Secondly, I want to create the option to have the performance data visualised as the environment is running, as is oft to happen on a more corporate dashboard. This will mean having to rework the backend somewhat, or possible reconfiguring both the back and front end so that both formats are able to work in unison.

Multiple libraries are already identified to perform this task, such as three.js and chart.js.

Furthermore, there is a plan to work towards the publication of my developments and findings through the University of the West of England, and as a result I will be continuing to work on this project for the foreseeable future as to work towards this goal.

Final Words

I extend the deepest thanks to everybody involved for this opportunity, it has been a great experience and I have learned much from my time exploring different methods visualisation and developing the tool. I hope that the work I have conducted here will represent me going forward and will open up many opportunities for me.