MALUS BINARY ANALYSES WHERE ANTIVIRII CEASE

HALLA SECURITY TOOL v2.0

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Ahmed J. Syed

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

October 2017

Fargo, North Dakota

North Dakota State University
Graduate School

Title

MALUS BINARY ANALYSES WHERE ANTIVIRII CEASE
HALLA SECURITY TOOL v2.0

By

Ahmed Syed

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kenneth Magel
Chair

Dr. Jeremy Straub

Dr. Kathryn Gordon

Approved:

16 November 2017
Date

Dr. Kendall Nygard
Department Chair

ABSTRACT

In the ethereal world of information security, threats posed by virii, and malware to an organisation's business critical assets, on the perimeter or otherwise, is a matter of great importance. Identifying potential vulnerabilities and the prevention of their being exploited is equally crucial. Ascertaining threats and discovering threat sources whilst developing solutions against the aforesaid threats is of significance.

Methods there are, to guard against known threats, however a disparity there exists where unknown virii and malware fly under the radar of detection, and thereby exposing businesses to risk, and to compound the threat, their discovery and remediation are assigned to security teams with limited training, resources, and toolset.

The Halla tool, the object of the research herein, seeks to bridge the observed chasm. Its design, construction, architecture, and deployment shall be examined whilst relevant works in the field shall be studied with their potential shortcomings stressed.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Malware are as prevalent as devices with operating systems are as ubiquitous, it be embedded or otherwise, in today's technology dependent milieu. Over one million malware, most of which had been developed in the past twelve months, are said to be in circulation with the number and complexity increasing exponentially [1]. Malware usage is no longer confined to obscure bores in dim dungeons seeking excitement of sort, but is mostly sponsored by nefarious criminal elements including nation states of which intent is to engage in technological, industrial, and military espionage. Petty pillaging of sensitive data including user credentials is more of a thing of the past [2] [3].

To comprehend the extreme nature of malware infections, 80% of companies had a security incident in 2015 with such incidents arising from exploits of known vulnerabilities *inter alia* SQL injection, cross-site scripting (XSS), cross-site forgery request (XSRF), session cookie hijacking, and weak encryption [4]; further, businesses, particularly in the financial industry, observe 106 unknown malware attacks per hour [5]. The number of unknown malware infections is believed to be far much greater, albeit its implications and subsequent potential loss is harder to quantify; for instance, Stuxnet, InfoStealer, Duqu *et cetera*, had only been discovered years after their release, and some having been in play for years in major financial and investment institutions [*Ibid*]. As shown in Figure 1.1, patching malware infections has neither being consistent nor speedy, albeit the trend has changed in recent years with vendors like Microsoft, Adobe, Apple *et cetera*, adopting more aggressive patching approach, and adapting rather aggressively to the rising threat landscape [6]. In its current state, a recent survey showed the average time to detect a cyber-attack is 99 days [*Ibid*].

Patch	Malware	Patch availability	Worm attack date	Days for worm to appear
MS01-020	Nimda	Oct 17th, 2000	Sept 18th, 2001	335 days
MS02-061	Slammer	July 24th, 2002	Jan 25th, 2003	185 days
MS03-026	Blaster	July 16th, 2003	Aug 11th, 2003	26 days
MS04-011	Sasser	Apr 13th, 2004	Apr 30th, 2004	17 days
MS05-039	Zotob	Aug 09th, 2005	Aug 14th, 2005	5 days
MS06-040	Mocbot	Aug 08th, 2006	Aug 12th 2006	4 days

Figure 1.1. Patching malware infection rate

In malware analyses methodologies, two common methodologies there are: a) code analysis, which entails sifting through opcode, and actual lines of code for potential bugs or defects, and b) dynamic analysis, which entails executing programmes in real time with input and output behavioural observations effected; the former requires proficiency in disassembling code, decompiling binaries, and a great deal of familiarity with Assembly programming whereas the latter requires a network environment, for instance virtual, in which programmes are executed with their operations cum behaviours observed. With each method, gains along with constraints are found; for instance, code analyses is detailed, rather time-consuming with a great deal of proficiency on the part of the engineer required whereas the latter is swift, less detailed where familiarity with the environment is sufficient with less technical expertise required as to the internal workings of the malware [2]; however such analyses are rather reactive only undertaken post-infection where signs of malware infection have been observed, and where the damage, financial or otherwise, has been committed.

With that being the case, most businesses employ stratified, preventative security posture in the form of firewalls, proxies, and network intrusion detection systems (NIDS) appliances along with anti-virus software, but lack proactive malware detection edifice, tools, or proficient personnel capable of detecting new or unknown malware infections; for instance, earlier variants of InfoStealers, which primarily targeted investment and financial institutions in North America and Europe, date back to the late 1990s, albeit had only been discovered in the mid-2000s having

been harvesting sensitive data including credit card and banking information, user credentials, intellectual property and propriety data amongst other things with no signs of infection or detection; its discovery was rather accidental than detection-based: a cause for concern [18].

Here is where the Halla security tool, the subject of discussion henceforth, which seeks to offer a sound solution to the aforesaid quandary, comes into the equation. It must be noted the Halla tool is not a panacea to all virii and malware infections, however it serves as the *absens nexus* of the aforesaid existing chasm, where no solution is found, for new or unknown virii or malware infections. It is an addition, and not a replacement, of existing security tools designed to offer stratified security solution from perimeter to intranet systems.

CHAPTER 2: RESEARCH OBJECTIVES

In examining traditional varii and malware analyses systems and tools, and in reviewing current cyber-security threats and vulnerabilities landscape, more so in the business world, prompted attention in not only researching the field, but proposing a solution, which if not contract the disparity, could potentially precipitate an interest amongst security professionals, and vendors. Investment and financial institutions, which form the primary target for most nefarious virii and malware attacks, provide unique insight into defence edifice against malware, whilst possessing some of the best security defence posture against such attacks. Small firms with limited budget, which fall on the opposite spectrum of the scale, provide a perspective of some interest. Equally important, if not more, are Supervisory Control & Data Acquisition systems (SCADA), which provide essential services including national power grid, water supply, natural gas to name a few, all of which are at greater risk against malware attacks, and are of greater importance to strengthening their safety and security.

With the aforesaid systems in mind, and against the backdrop of an ever so volatile global cyber-security threat landscape, with state-sponsored actors in play more often than not, an automated, proactive solution that which assists security professionals is required, ergo Halla security tool, which offers [a] solution to that effect, devised to assist security professionals, and further bolster existing security tools against malware attacks in an organisation's overall security structure.

CHAPTER 3: LITERATURE REVIEW

Software testing, a subject beyond the scope herein, is argued, at best, to disregard software security viewing it as if an afterthought charge rather than an integral part of software development life cycle (SDLC), and at worse, wholly overlooking it until a bug or a vulnerability has been identified, in production in most cases; it is further contested software developers embrace the reverse philosophy in that software testing is performed to show "a program performs its intended functions correctly" or "demonstrating that errors are not present" instead of "executing a program with the intent of finding errors" [38].

Whilst large organisations now maintain security teams, a stark disparity exists between software development and security teams where each team operates in its own dominion with little, if any, direct collaboration with the other; for instance, the former regards the latter, with contempt, and as no more than glorified vandals whilst the latter views the former as no more than wooden cads. It has been asserted that, in large organisations, security teams are granted greater influence over organisation's security posture, and in security policy position, which further diverge the aforesaid teams with software development teams viewing security teams as adversaries to be circumvented, battled, and conquered [21]. One could be so bold as to posit software developers along with project managers regard security testing a superfluous appendage, and a costly inconvenience begetting nothing more than delayed deliverables, missed deadlines, and sundry project expenses; however, empirical studies show initial cost of carrying out reasonably exhaustive security tests are found to cost much less than subsequent efforts in patching a vulnerability, or addressing financial loss, or potential lawsuits, resulting from a security breach [29].

With that in mind, malware, be it virii, malware, or worms, changed the dynamics of the computing world, both from a software developer perspective, and a user standpoint, particularly in its assailant nature and fierce prevalence on the Internet. On the one hand, it persuaded software developers to heed software security and secure coding methodologies whilst on the other hand imprinting its signature in the psyche of the average user, who grasps not quite as to the gravity and enormity of its prevalence along with its adverse ramifications, financial or otherwise.

With respect to known virii and malware, security and anti-virus vendors serve as a layer of security amongst an organisation's heuristic security edifice, however detecting unknown virii and malware on systems, business critical production systems or otherwise, is a subject, of enormous interest, to which less attention and resources are being ascribed albeit bears considerable adverse implications, financial or otherwise to businesses, and to a lesser impact to persons, largely owing to failure to its being detected in time, thoroughly investigated, and correctly remediated. In the wake of unknown malware, of which signature, content, properties, or effects are neither defined nor identified, gains a foothold in a business network, succeeding consequences could be greatly injurious, and the greater the threat, the grander the ruination in its trail, particularly to business organisations, to nation state infrastructure, or to SCADA systems, namely security incidents apropos the Northeast electric grid in 2003, Springfield, IL, water supply in 2011, the International Atomic Energy nuclear power plant in 2016, and hijacking major US, Asia, and European cities' traffic control systems, in 2014 to name a few [5].

An imbalance there too exists betwixt the sophistication of unknown malware, of which threat grows in time and scale, and the utility of existing security tools most of which are found

to be less congruous in sophistication, lacking granularity whilst being post-infection-centric generally effected post the feat; equally disconcerting is the fact that the skillset of malware authors is far superior to that of security corps to whom the task of protecting, for instance the national power grid, is entrusted.

That coupled with a deterministic, if perpetual rise in the number of [daily] discovered malware, estimated at 1.4 million unique malware, excluding yet to be identified malware, which forces the subject to the fore for concerted examination [19].

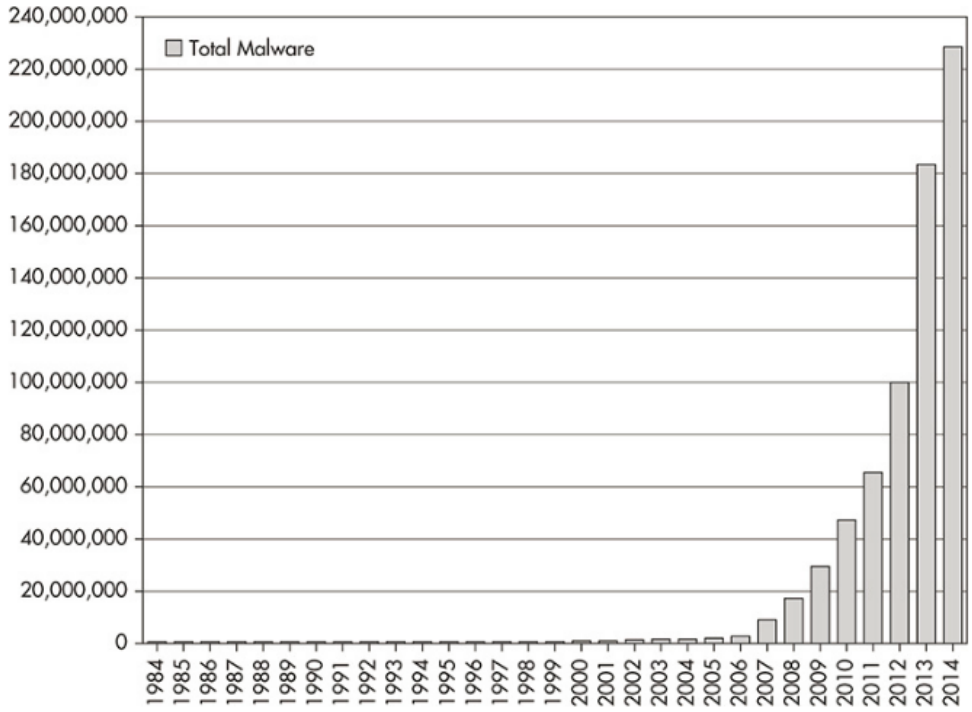


Figure 3.1. Malware discoveries from 1984 to 2014

To illustrate the legitimacy of the aforesaid argument, one must consider, for instance, variants of the InfoStealer malware, or Point-of-Sale (PoS) malware as at times referred to, targets PoS networks, and is believed to be the greatest source of stolen payment cards (ACH, PCI, DDAs *et cetera*) with its greatest effects felt from 2013 to 2015 impacting, in the process, nearly 100 million payment cards; forty (40) million credit cum debit card payments along with

seventy (70) million personal records had been compromised in a PoS malware attack against a single high street retailer in the US alone in 2015 [30]. And many a large organisation is potentially still vulnerable to the exploitation.

Further, Boleto malware, which is believed to date back as early as 2010, yet only discovered in early 2013, infected 192,277 unique IP Addresses, harvested 83,506 user credentials, costing Brazil 3.7 billion US dollars [39]. Equally elusive was Duqu, a memory resident malware, which made its detection rather arduous, believed to have targeted Iran's nuclear arsenal along with security firms with privilege escalation capabilities [27].

Furthermore, Stuxnet, a 500KB worm targeted Microsoft Windows systems, Siemens Step7 software, and programmable logic controllers enabling its authors to spy upon industrial systems, and by extension destroying centrifuges with its primary target being fourteen (14) industrial sites in Iran subsequently begetting considerable impairment to Iran's nuclear infrastructure [27]. Whilst initial detection along with ensuing remediation pales in comparison with the intricate and elusive nature of the aforesaid malware, post reversing and analyses showed an apparent incongruity betwixt competing coteries with mastery on the attacker's side rather evident.

For reflection, in its infant eons, malware analyses was mainly performed manually with a handful of tools and personnel, however whilst the dynamics, complexity, and pattern of virii and malware greatly advanced, defence techniques, methods, and concepts unfortunately remained the same albeit toolsets along with network environments, with the advent of virtual networks, relatively improved; for instance, when BRAIN, the first recorded virus, targeting floppy drive boot sectors, was discovered in 1986, stealth in nature and of its kind at the time, it was difficult to detect, and remediate it, and whilst the threat landscape drastically matured over

the years, skillset on the part of qualified personnel lags behind [22]. For instance, threat landscape has considerably evolved with ransomware attacks against businesses increasing in three-fold in comparison to previous two years, and with 44% of companies in North America suffering four (4) or more data breaches in the past year [39].

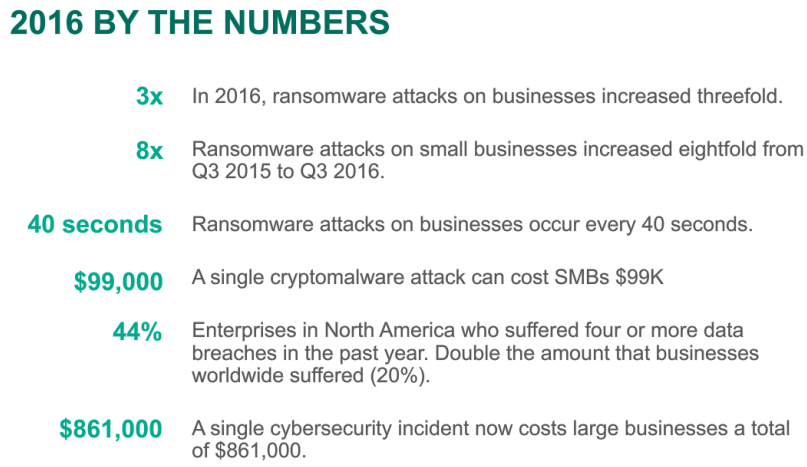


Figure 3.2. Ransomware attack increase

Whilst anti-virii software and malware detection tools and techniques considerably evolved since then, evidently there exists a disparity betwixt malware authors vis-à-vis security professionals [38]; for instance, more often than not, the former is highly skilled, dedicated, resourceful, and motivated by financial gain with a great deal of effort in remaining anonymous whereas the latter is operating under extreme circumstances lacking in resources, skill, and most importantly time.

Against the aforesaid backdrop, it is reasoned that existing toolsets along with security corps are ill-equipped to effectively address pending challenges, posed by malware, particularly where nation states, and/or nation state-sponsored organisations are concerned; further, there exist limitations with malware reversing and analyses in that malware protection techniques including encryption, packing, and obfuscation are far too complex requiring highly skilled corps and bespoke tools to meet the challenge [19]. It is further asserted that decryption, unpacking,

and de-obfuscation techniques are no mean feat, and could potentially dissuade many a security professional from embarking upon malware reversing, particularly engaging in static analyses (*Ibid*).

Automating malware analyses is where the Halla tool, the subject of discussion herein, comes into the equation. It must be noted before undertaking this solution, existing solutions had been studied, most of which had been used in business environments, where a security gap was observed, thusly inspiring the advent of the Halla security tool. Observed was deficiency on the part of anti-virus vendors in detecting unknown virii and malware infections where their respective libraries contain not the signature for the malware, be it a new variant of an existing malware, or a mint breed. It was further observed existing malware analyses engines offer a manual process requiring binaries to be fed to the engine(s) post infection, detection, and discovery, a principally rather reactive approach.

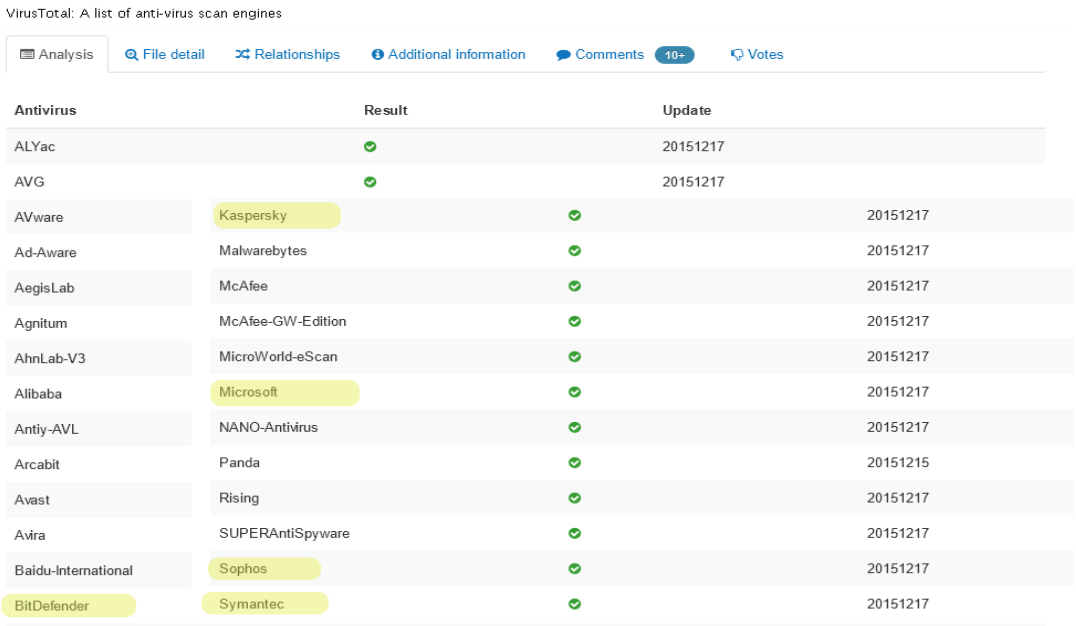
Ergo, an automated, robust malware analyses tool is a must to augment existing anti-virus software, and discussed henceforth shall be the Halla tool, a solution deemed befitting of which object is to contract the observed disparity.

3.1. Related Work

Before undertaking this solution, traditional solutions, in place, had been studied, where a security gap was being observed, ergo inspiring the advent of the Halla security tool; it was found none of the existing anti-virus vendors are capable of detecting unknown virii and malware infections provided their respective libraries contain not a signature for the malware, be it a variant of an existing malware or a new breed.

Virus-total [7], an open source anti-virus scan aggregate engine, which hosts signatures of nearly two-dozen anti-virus software vendors including Symantec, Kaspersky, Sophos,

Windows Defender *et cetera*, is used to identify signatures of all virii and malware binaries prior to their being analysed, and where signature(s) is found in its repository or is recognised, no analyses is carried out, as shown in Figure 3.3; however, if signature(s) is not recognised, then it is submitted for further analyses whence it is disassembled, decompiled, and reversed.


VirusTotal: A list of anti-virus scan engines

Analysis | File detail | Relationships | Additional information | Comments 10+ | Votes

Antivirus	Result	Update
ALYac	✓	20151217
AVG	✓	20151217
AVware	✓	20151217
Ad-Aware	✓	20151217
AegisLab	✓	20151217
Agnitum	✓	20151217
AhnLab-V3	✓	20151217
Alibaba	✓	20151217
Antiy-AVL	✓	20151217
Arcabit	✓	20151215
Avast	✓	20151217
Avira	✓	20151217
Baidu-International	✓	20151217
BitDefender	✓	20151217

Kaspersky | Malwarebytes | McAfee | McAfee-GW-Edition | MicroWorld-eScan | Microsoft | NANO-Antivirus | Panda | Rising | SUPERAntiSpyware | Sophos | Symantec

Figure 3.3. VirusTotal Anti-virus active scan engines

Stand-alone malware behavioural analyses tools including Anubis, Cuckoo, and FireEye, as well as Open Source online engines including BinaryNinja, all of which provide post-infection analytical services, had been examined as part of determining as to the severity of the identified gap. Noteworthy, it is that no pro-active tools had been found during the research period.

3.2. Definition

The Halla security tool is a proactive tool, which adds a great deal of value to the concept of defence in-depth subsequent to security industry gap analyses where anti-virus vendors, which adopt a more reactive detection methodologies than not, are found to fall short in addressing new and/or unknown virii and malware threats to the enterprise; *vigilo*, a file-watching agent is installed on hosts providing full protection against unknown and potentially malicious binaries

(.exe), dynamic-link libraries (.dlls), system configuration files (.sys | .ini), drivers (.drv), installation packages (.msi) amongst others, and thereby protecting an organisation's business critical assets. On the backend, malware analysing engines disassemble binaries deemed risky or nefarious to the enterprise, and where further analyses is considered necessary, manual static analyses and reversing pursues to ascertain the nature and type of the binary in question. A web interface there is on the front-end, where security analysts could monitor activities, conduct further analyses, or escalate, should need arises, and determine as to the requisite remedial action(s) concerning binaries of interest.

3.3. Solution

Departing from the aforementioned premise, a set of tools including the Halla security tool are being developed to assist security remediation and engineering teams in identifying potential threats prior to their begetting data breach or compromise whilst mitigating potential threats and vulnerabilities; the tool is intended to target binaries of unknown signatures, and of zero-day vulnerability exploits; it is not however to replace anti-virus software with its vast signature libraries.

Discussed shall be the objectives, functionalities, rationale, technological, and development approach, architectural design, quality attributes, viewpoints and perspective catalogues, constraints, design patterns cum principles, design artifacts cum their analyses, implementation artifacts cum their analyses, and code snippets of the Halla system [8] [9] [10] [11][12].

Analyses along with functionalities of the tool by and of itself shall remain within the scope herein, and in line with the project guidelines, as stated in the project specification

requirements, whilst detailed analyses of applied technologies including Python libraries, Open Sources libraries, and .NET framework fall beyond the scope herein, ergo shall not be discussed.

3.4. Functionality

The Halla tool seeks to add strength to an organisation's security edifice with the intent to contracting, if not closing the discussed security gap, which is not being met by traditional anti-virus software of which deficiencies had been researched, over a long period of time, in organational security structures; it further attempts to bolster existing security tools, and is considered a supplement rather than a substitute; it seeks to guard against immediate anomalies, malware related or otherwise, where no other mechanisms exist to protect business assets.

It shall be used in tandem with existing anti-virus and threat analyses tools to further assist security professionals to identify, and report malicious activities on, any and all, hosts on target network environments, and for purposes pertaining to virii or malware infections, phishing, identity theft, sources of malware, and other fraudulent intent.

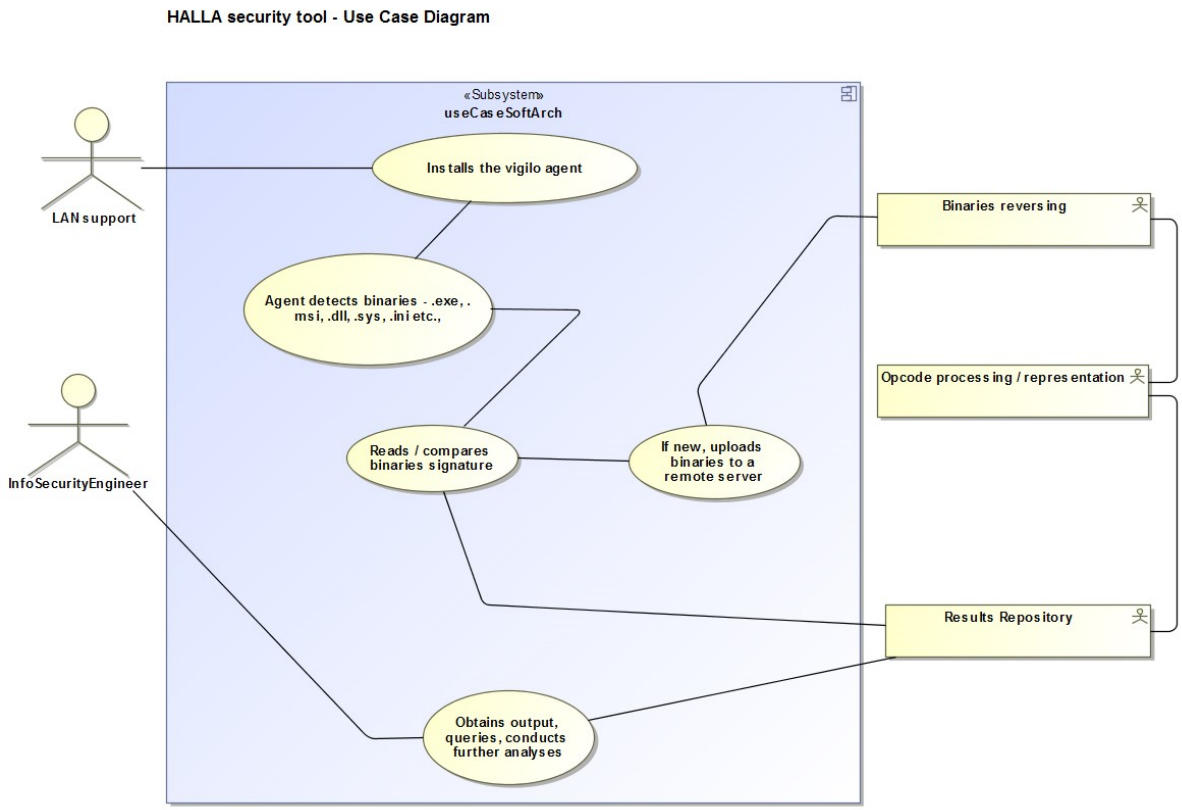


Figure 3. 4. Use case

3.5. Rationale

As an additional security layer, the Halla security tool seeks to proactively prevent zero-day exploits providing binary signature analyses that which anti-virus software recognises not, and is unable to remediate or mitigate; further, it intends to assist security engineers to proactively assess network environments for potential malware residence and exploits, and to avoid being blindsided by what could potentially cause injury to an organisation's resources.

In its design, a number of scenaria, each of which shall be discussed herein, had been considered where the Halla tool would be effective in its application.

3.5.1. Scenario A

An employee returns to work having been remotely working from home or from a trip abroad; unbeknownst to him, his host was infected with a zero-day, multifaceted network

spreading worm, which in its initial stage, targets **tftp** protocol used by network switches and routers, collects important information including routing and subnet information, routers / switches credentials *et cetera*, about the target network; on the first Saturday of each month, its client, which hooks into Internet browser, Windows Internet Explorer (IE) in this instance, remotely connects, via an encrypted channel, through web proxy, to a remote server to upload harvested data; this continues with security engineers being none the wiser, and no anti-virus, firewall, or intrusion detection mechanisms detecting worm infection, or data exfiltration.

3.5.2. Scenario B

Under this scenario, the same incident, as in Scenario A, takes place in an organisation where the Halla security tool is installed on all its clients hosts, particularly laptops; employee's computer is infected with the same worm, however at the time of the infection, *vigilo*, the file-watching agent on the host, would create a log detailing worm infection whilst recording all its subsequent activities on the target host, and creating an incident with the security engineering team, who would initiate an investigation; sample(s) of all binary files, associated with the worm, would be automatically uploaded to a Sandbox server, where its analyses would be initiated; in the event it transpires the worm infection to be of a malicious nature, posting threat to the enterprise, the entire network would be swept for signature matching instances and incidents so as to eliminate the threat. All security teams including the Incident Response team along with senior management would be notified.

3.5.2. Scenario C

Under this scenario, the same incident as in Scenario A, takes place only this time the employee does not connect to the corporate network for a period of time, be it days or weeks, subsequent to the worm infection; this is one of the shortcoming of the Halla tool, which is being

discussed at the moment, and shall be addressed in future releases where any host that which connects not to the corporate network for a period longer than three (3) weeks would have its network interface(s) disabled, and shall not be permitted to reconnect to the network until it has been checked, validated, and remediated, should need arises.

3.6. Stakeholders

Syed, Ahmed – Software Engineer & Architect

Whilst I served as the software engineer and architect for the development of the tool, other stakeholders included my employer, who financed the project providing test lab environments including hardware, software, and licensing for commercial tools. Other stakeholders included colleagues, who contributed to the design, development, testing, and deployment of the tool. For instance, one colleague was responsible for documentation with another assuming the role of testing the tool in its various stages whilst another was tasked with designing and maintaining test lab network environments.

3.7. Target Audience

The Halla tool shall be used by security professionals, remediation, and security engineers, who carry out security threat, vulnerability, virii and malware analyses and assessments.

3.8. System Implementation

Following are the system implementation details:

1. Develop the Halla tool, where binary reversing and signature analyses is being carried out,
2. Binary signature comparative analyses is being conducted,

3. A repository of all binaries along with binaries MD5 and SHA signatures are being created, and
4. A report of binaries analyses of both known and unknown signatures is being generated.

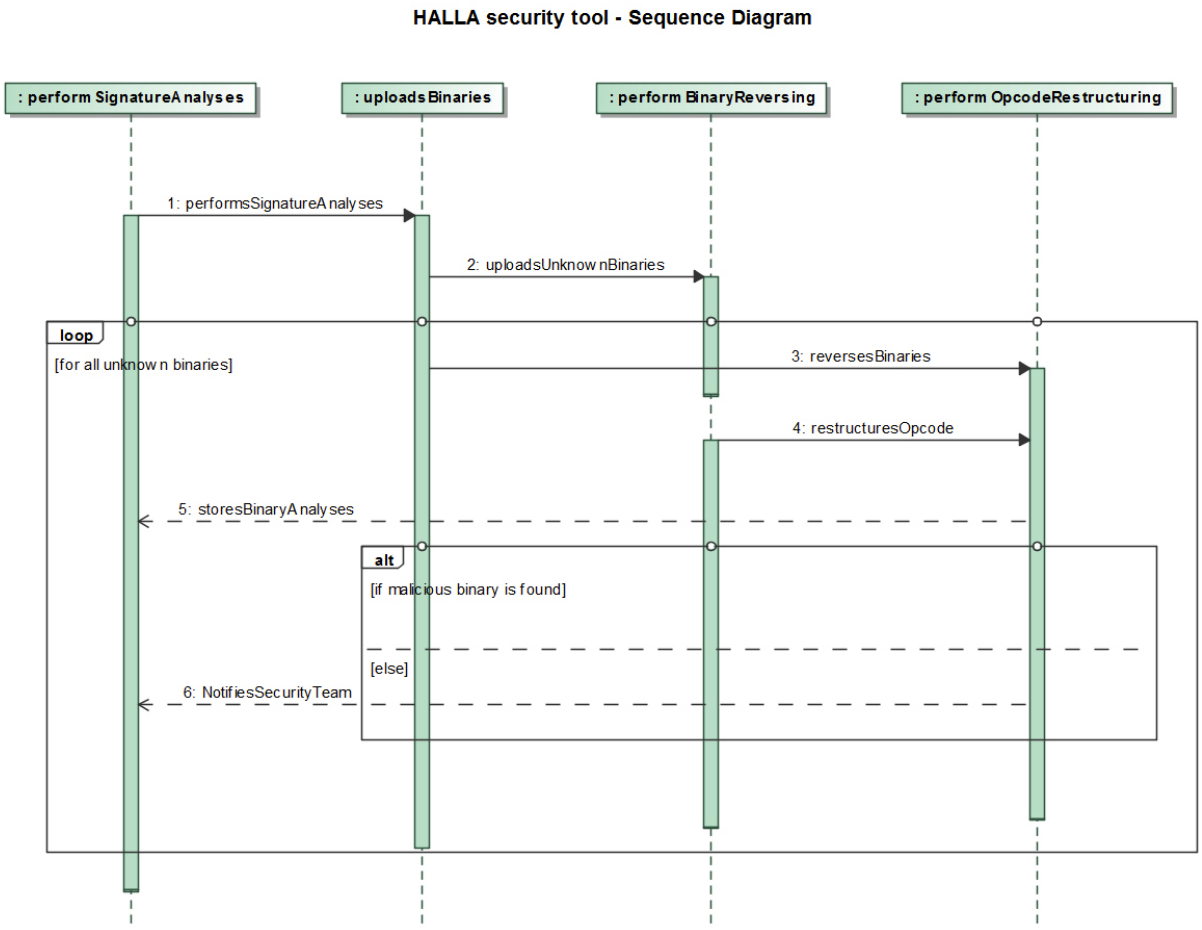


Figure 3.5. Sequence diagram

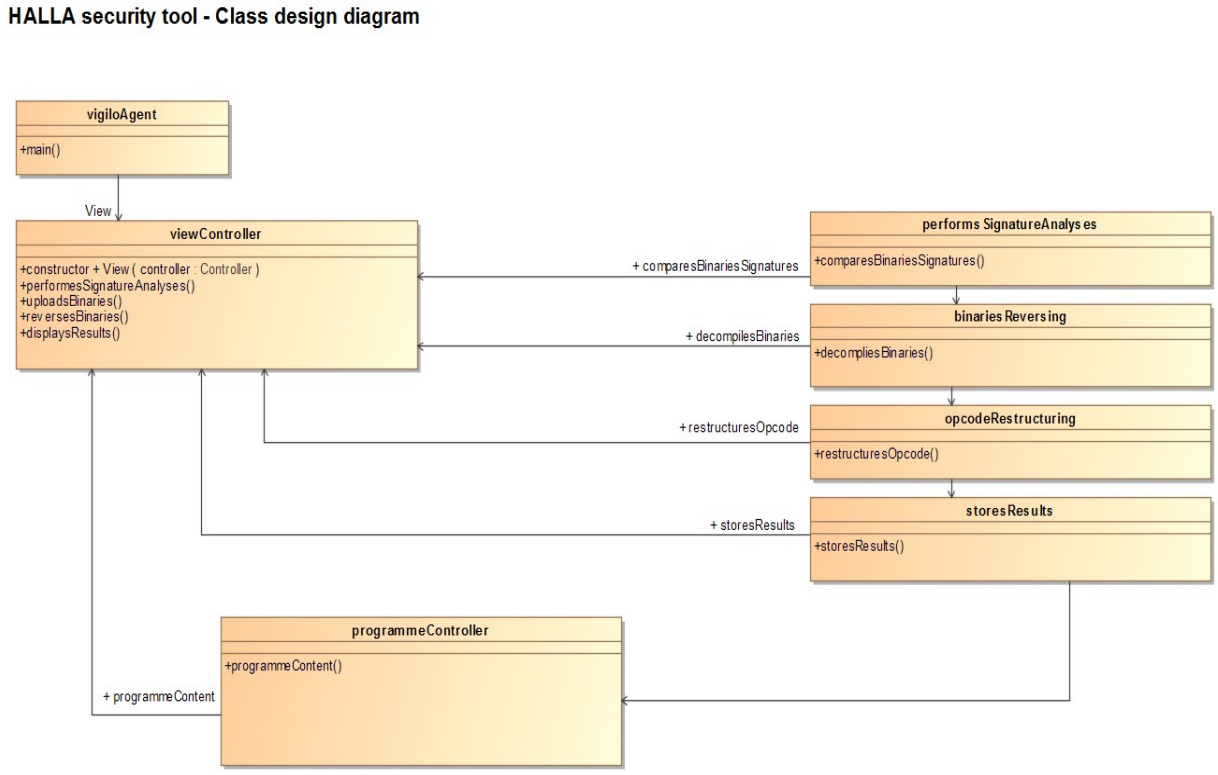


Figure 3.6. Class design diagram

3.9. Design Pattern

Model-View-Controller is an important architectural design pattern which seeks to partition system or application framework into three interoperable, manageable entities; for instance the Model represents the data layer in a logical way, and is in charge of carrying the data and making other objects aware of data changes; the view is a graphical representation of the Model responsible for displaying the Model data in suitable form, whereas the controller orchestrates the pattern in charge of intercepting user input, and interacting with the Model and/or the View [10]; here, the MVC design pattern is applied in the development process of the Halla system.

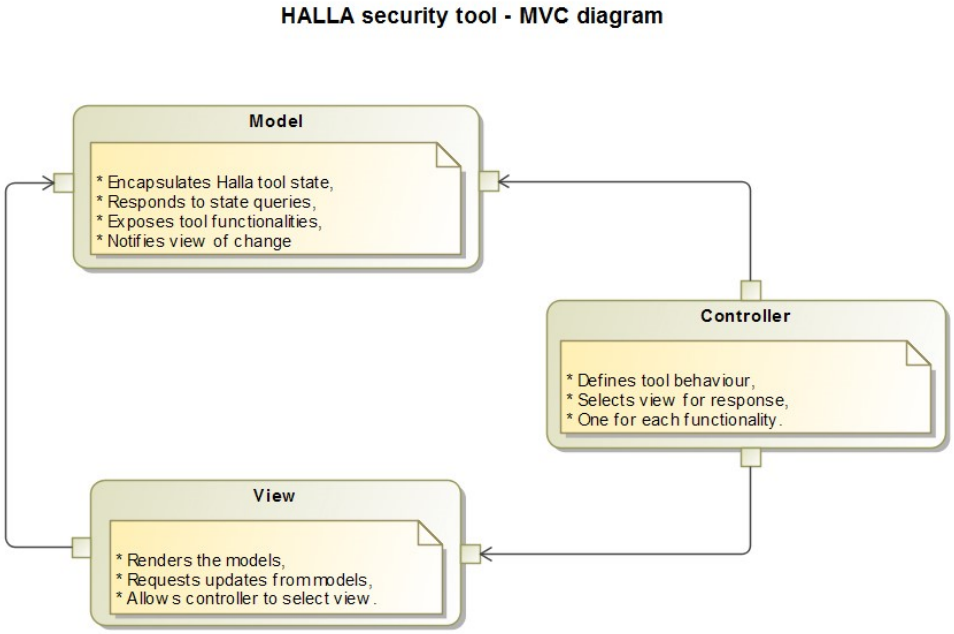


Figure 3.7. MVC diagram

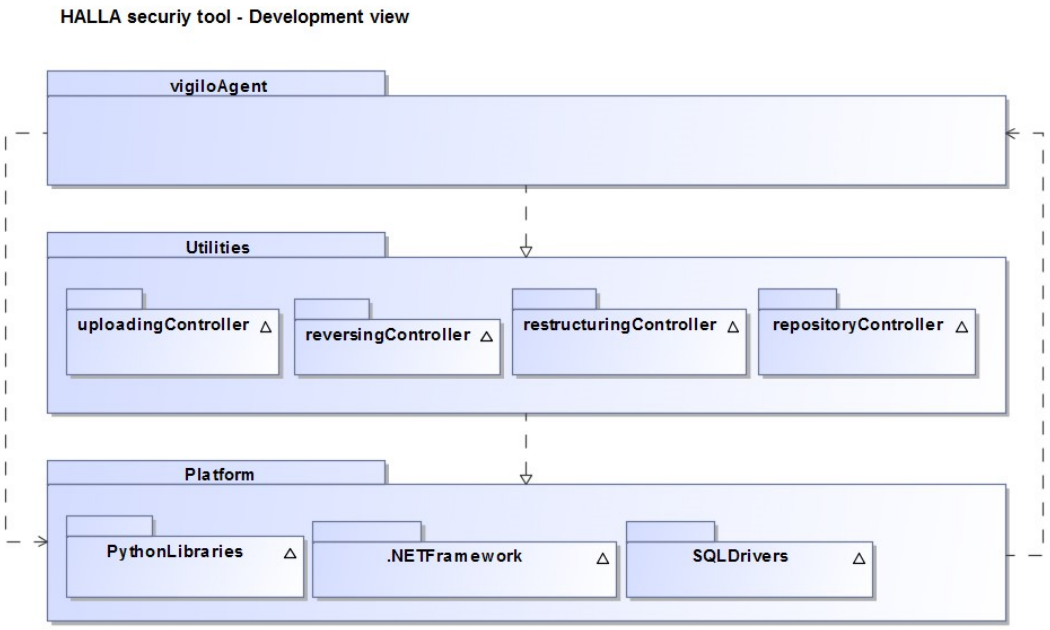


Figure 3.8. Development view

3.10. Paradigm

- Unified Process elaboration model,
- Tackling major risk items at the project commencement,
- FURPS+ modalities including security.

3.11. Architectural Patterns

- Context – a situation which gives rise to a problem,
- Problem – a problem that which arises subsequent to the aforementioned context, and
- Solution – an architectural resolution befitting the problem in the aforesaid context.

Halla comprises two primary components:

i. A file-watching '*vigilo*' mechanism, which monitors anomalies, binaries drops or executions, and file state changes including file update, deletion, creation, or renamed on a target host;

ii. A sandbox environment where all binaries are being disassembled, and reversed with signature comparative analyses, and in-depth analyses being carried out; here, as and when an unrecognised binary file(s) has been identified, signature comparative analyses is firstly conducted pursued by binary and opcode analyses in a sandbox environment; if the binary is found to be of malicious nature, and of unknown status, then a security engineer would effect a remediation plan where any host found to be hosting a matching signature is mitigated.

3.11.1. Component I: File-watching 'vigilo' agent

Under the first component, the Halla tool monitors files on the local host watching for potential threats originating from binary drops, known or otherwise, and in the event a known file is being accessed, the tool conducts comparative binary signature analyses against a central database subsequent to which a determination of whether the binary is a threat, or not, is being made.

```
33      def fileWatchMethod(self, pPath):
34          FILE_LIST_DIRECTORY = 0x0001
35          hDir = win32file.CreateFile (
36              pPath,
37              FILE_LIST_DIRECTORY,
38              win32con.FILE_SHARE_READ | win32con.FILE_SHARE_WRITE | win32con.FILE_SHARE_DELETE,
39              None,
40              win32con.OPEN_EXISTING,
41              win32con.FILE_FLAG_BACKUP_SEMANTICS,
42              None
43          )

51              while 1:
52                  results = win32file.ReadDirectoryChangesW (
53                      hDir,
54                      1024,
55                      True,
56                      win32con.FILE_NOTIFY_CHANGE_FILE_NAME |
57                      win32con.FILE_NOTIFY_CHANGE_DIR_NAME |
58                      win32con.FILE_NOTIFY_CHANGE_ATTRIBUTES |
59                      win32con.FILE_NOTIFY_CHANGE_SIZE |
60                      win32con.FILE_NOTIFY_CHANGE_LAST_WRITE |
61                      win32con.FILE_NOTIFY_CHANGE_SECURITY,
62                      None,
63                      None
64                  )
65                  for action, fFile in results:
66                      try:
67                          fullFileName = os.path.join(fFile) # excluding pPath
68                          resultSet = fullFileName + "\t" + ACTIONS.get(action, "Unknown")
```

Figure 3.9. *Vigilo* agent code snippet

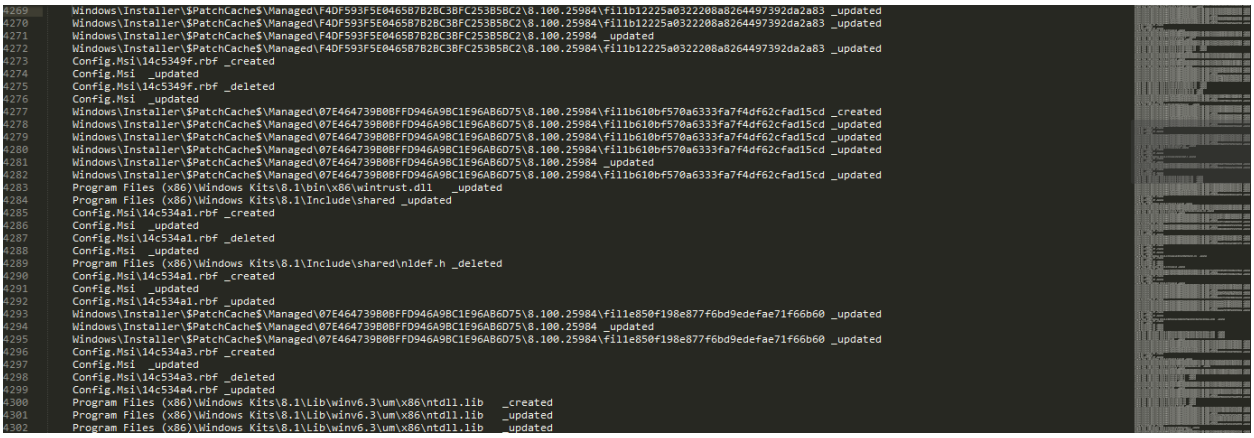


Figure 3.10. *Vigilo* resultSet

A portable executable (PE) header parser strips binaries to expose its directory tree structure along with its string representation to ascertain as to the nature, type, origin and other attributes of interest in furthering binary file analyses.

Figure 3.11. PE header parser result

An API script searches, rescans, and runs anti-virus signature validation against VirusTotal, an open source virus signature validation engine.

```
class vtAPI():
    def __init__(self):
        self.api = 'd21aeffb09eae1d43c14bad1a809df6a879bd328d774d203b9060969af42a3cf'
        self.base = 'https://www.virustotal.com/vtapi/v2/'

    def Search(self, md5):
        param = {'resource':md5,'apikey':self.api}
        url = self.base + "file/report"
        data = urllib.urlencode(param)
        result = urllib2.urlopen(url,data)
        jdata = json.loads(result.read())
        return jdata

    def reScan(self,md5):
        param = {'resource':md5,'apikey':self.api}
        url = self.base + "file/rescan"
        data = urllib.urlencode(param)
        result = urllib2.urlopen(url,data)
        print "\n\tVirus Total Rescan Initiated for -- " + md5 + " (Requery in 10 Mins)"
        print ("\t+ " + str(result))

# Md5 Function

def checkMD5(checkval):
    if re.match(r"([a-fA-F\d]{32})", checkval) == None:
        md5 = md5sum(checkval)
        return md5.upper()
    else:
        return checkval.upper()
```

Figure 3.12. VirusTotal API code snippet

The threat portal is then updated with current events populated awaiting review and remediation by members of the senior security engineering team; further, the backend central database is also updated for future reference.

In the event, a binary signature is determined to be unknown, it is uploaded to a remote sandbox environment for further analyses – a hand-over to the next step of the Halla security tool.

3.11.2. Component II: Binary disassembling, reversing, and restructuring analyses

With the status of the binary file being unknown, it is deemed as a threat to the organization, thusly the binary file is received in a sandbox environment where it is decompiled, stripped, parsed, and reversed for in-depth analyses; an alert is also generated to the on-call security engineering team, who shall then initiate further analyses subjecting the binary file to more stringent analyses; here, a determination as to the nature and impact of the threat is being made.

Further, the threat portal, which comprises of critical, medium, and low threat events is updated with technical write-ups, impact analyses, plan of remediation along with comments for future reference;

Furthermore, as and when a binary file is found to be of malicious nature, the security engineering team evaluates its business impact initiating an Incident Response edict, should the event is deemed to warrant an escalation; here, senior management is briefed with principal engineering team kept abreast of event development;

Moreover, the remediation plan is being put into effect eliminating the threat whilst scoping the environment for similar events, and engaging the LAN support teams of respective lines of businesses in the enterprise;

And finally, anti-virus and threat analyses vendors are notified whilst availing the binary for its inclusion in their respective signature libraries.

Figure 3.13. String representation

Figure 3.14. Code representation

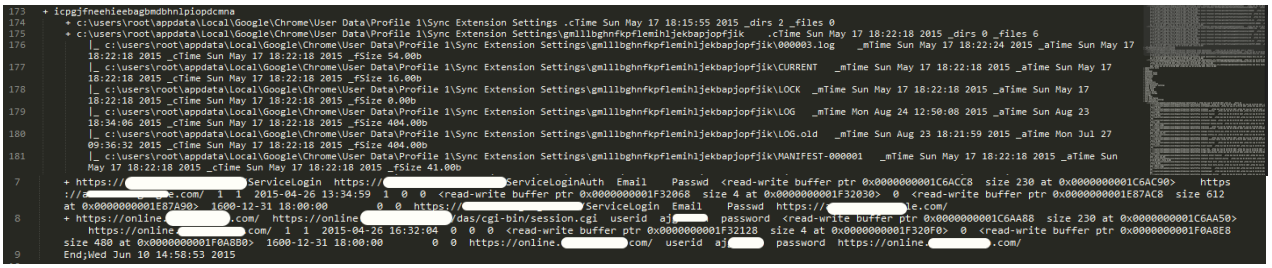


Figure 3.15. Relevant cached data from the Browser, Chrome in this instance

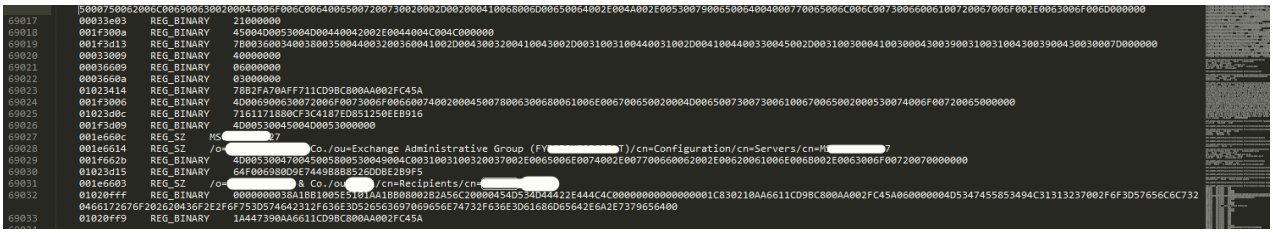


Figure 3.16. Relevant data obtained from the system registry

CHAPTER 4: RESULTS & DISCUSSION

Discussed in this chapter, shall be a small sample of selected population whilst presenting results of research findings along with data analyses. A small sample was selected for two (2) reasons: firstly, majority of the sample data might contain private, and/or propriety type data, and secondly binary files presented herein are analysed using the Halla tool whereas all binaries are submitted to multiple engines some of which are licensed commercial engines of which use is not for public usage.

4.1. Data Analyses & Findings

Results of the virii and malware analyses, of a small sample population selected out of a larger sample data, shall be presented and discussed herein whilst remaining close to the principal objective of the research. The intent was of two-fold: a) demonstrate inherent shortcomings of current anti-virus software where unknown virii and malware are concerned, and b) exhibit the efficacy and utility of malware analyses tools, meeting the challenge of maintaining secure network environments, and to enhance capabilities of conventional anti-virus software.

4.2. Sample Data

A small sample data of suspected malware binary files had been selected for analyses with the results of thirty-five (35) of the aforesaid files being reviewed. A randomly selected few shall be presented and discussed herein. So was done for reasons of brevity, expediency, and protection against divulgence of propriety data, and adherence to licensing restrictions of commercial tools.

Sample binary files were acquired through a period of thirty-six (36) months with the target population selected randomly with little, if any, prior knowledge as to their being of malware type, or not, and as the results shall support, the principal objective had been realised.

4.3. Test Lab Environment

Two (2) physical test lab network environments along with a virtual environment, comprising virtual hosts along with physical machines, with a number of engines conducting binary analyses, had been created. There are physical firewalls separating the environments with strict inbound and outbound network connections in the form of access control list (ACL) put in place, and whilst test hosts have outbound connection to the Internet, server engines are only accessible inbound via SSH (secure shell) with a minimal top-range ports accessible. To maintain the test environment, Deep-freeze software is used to sanitise hosts post infection, and prior to the next round of infections.

For the purpose of this research, a sample data of thirty-five (35) random binary files, suspected of being of malicious nature, had been submitted for analyses using the Halla tool of which results shall be presented herein. Further, five (5) other malware reversing engines had been used to analyse the same binaries for comparative reasons, albeit their results could not be shared owing their being licensed commercial software, and not for public use.

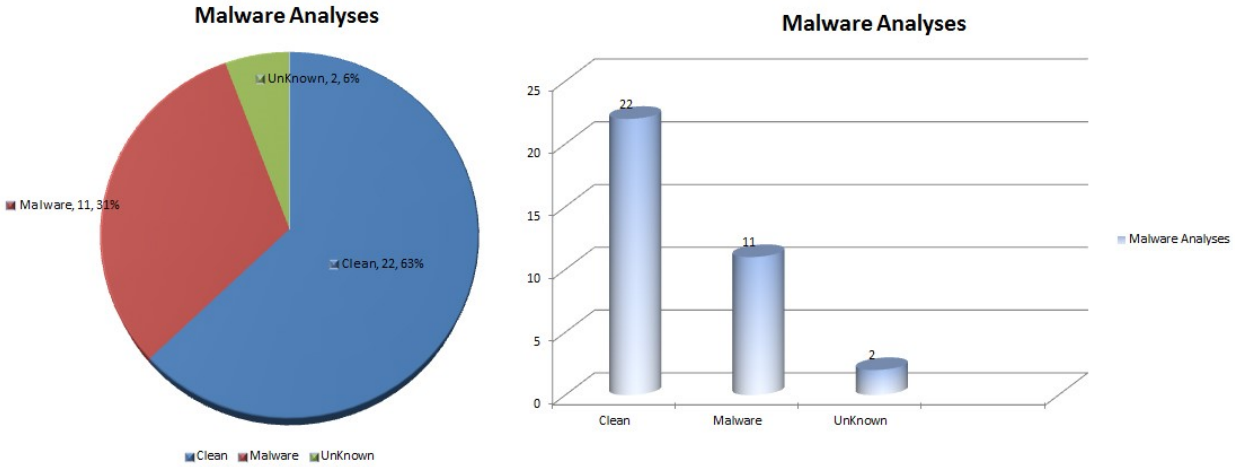


Figure 4.1. Malware samples analyses

Collected over a period of thirty-six (36) months, binaries had been firstly submitted manually for analyses to anti-virus software, and to stand-alone malware analyses engines. Of those, twenty-four (24) binaries had been found to be clean posing no threat whereas eleven (11) binaries had been classified as being of malware type posing threat.

However, when the same binaries had been submitted to the Halla tool for analyses, it classified twenty-two (22) binaries as being clean, eleven (11) binaries as being of malware type posing threat, and two (2) binaries, which are of great interest, as being unknown flagging them as requiring escalation, and further analyses.

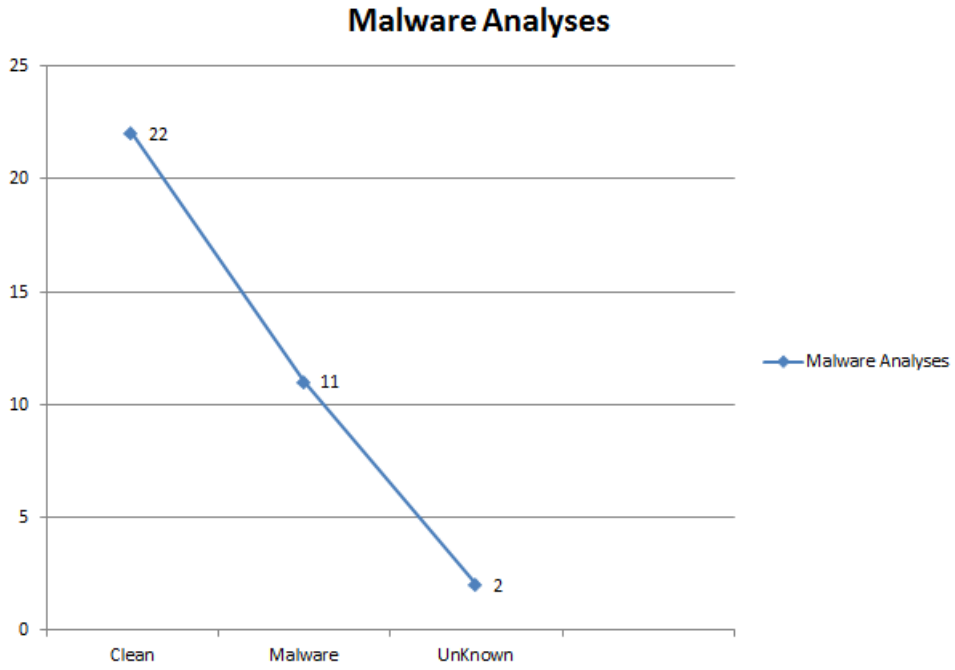


Figure 4.2. Malware samples breakdown

If a binary is not recognised, and could not be analysed, for reasons including that it is being either packed guarding it from interference, or encrypted protecting it from being disassembled, it is queued for further analyses, which entail: a) submitting it to the Halla tool to unpack or decrypt, and to analyse it with results reviewed by senior security engineers, and b) subjecting it to static analyses by a senior security engineer whilst notifying other teams, including LAN Support, Corporate Risk, Legal Department *et cetera*, should need be. Here, for instance, two (2) of the binaries were classified as "unknown" requiring further analyses, and thereby precipitating their being escalated for, and subjected to, further threat analyses. The intent of doing so is of two-fold: a) if a binary is deemed of either being shielded from interference, or encrypted, its potential threat must be addressed, and b) this prevents there existing a gap where some binaries are left unanalysed, albeit still posing a threat to the

enterprise. It further allows teams to engage anti-virus vendors to whom new virii or malware discoveries are reported for signature update on their respective signature libraries cum repositories, and by extension for the protection of the wider community.

Analyses of the two (2) unknown malware binaries, neither of which had been identified by anti-virus or conventional malware engines, shall be further examined, to demonstrate the utility of the Halla tool without which malware could potentially have spread to network environments begetting considerable injury to critical business applications, resources, and shareholder equity.

The **first** unknown binary to analyse was "simple.exe" of which properties of interest included: size, packer type, signature, directory location *et cetera*.

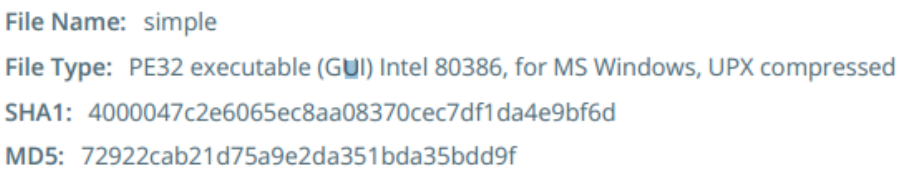


Figure 4.3. Malware sample type & signature

In disassembling the binary file, it was discovered a host browsing the Internet downloaded a binary file, which was then dropped unto the local temp directory "c:\%userprofile%\appdata\local\temp", from where it was executed at which point a number of nefarious activities including remote outbound connection to an external host, of IPAddress 198.20.167.83, of latitude and longitude: -78.8788 and 42.8825, identified as belonging to NetSolutions, a firm based in Buffalo, NY had been initiated. The importance of this location is that malware was using a legitimate website, NetSolutions's unbeknownst to its owners, to host malicious binaries, as an intermediary source, to target and infect businesses in the United States, albeit malware's original source was Russia. This was done to hide its origin, to mislead, and to throw researchers off its track.

Figure 4.4. Malware sample internals

It was further discovered the binary file was packed with “Armadillo”, as shown in Figure 4.5, primarily used to guard malware against debugging, disassembling, and decompiling. In so executing, the malware creates a set of entries in the registry, writes itself to the registry “run” key for future execution, reads a number of files searching for key data strings, creates two (2) mutices, attempts to establish an outbound connection to a remote host possibly for downloading additional malware, potentially for remote data dump, and finally deletes a set of files from the registry and infected host directories, as shown in Figures 4.6 and 4.7.

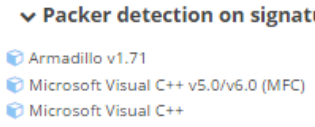


Figure 4.5. Malware packer type

CreateFile

brbconfig.tmp
C:\Windows\system32\rsaenh.dll
C:\Users\win7\AppData\Roaming\Microsoft\Crypto\RSA\S-1-5-21-3979321414-2393373014-2172761192-1000\58b8aea4ae7184a912187a66498d9b0c_c4b6765a-c53d-4b48-b576-0e1db4e9f3bc
\\.\Nsi
C:\Users\win7\AppData\Local\Microsoft\Windows\Temporary Internet Files\counters.dat

Figure 4.6. Malware creates files in registry

OpenRegisteryKey

Software\Microsoft\Windows\CurrentVersion\Run
Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache
Software\Policies\Microsoft\Internet Explorer\Main\FeatureControl
Software\Microsoft\Internet Explorer\Main\FeatureControl
FEATURE_HTTP_USERNAME_PASSWORD_DISABLE
RETRY_HEADERONLYPOST_ONCONNECTIONRESET
FEATURE_MIME_HANDLING
FEATURE_BYPASS_CACHE_FOR_CREDPOLICY_KB936611
FEATURE_IGNORE_MAPPINGS_FOR_CREDPOLICY
FEATURE_INCLUDE_PORT_IN_SPN_KB908209
FEATURE_BUFFERBREAKING_818408
FEATURE_SKIP_POST_RETRY_ON_INTERNETWRITEFILE_KB895954
FEATURE_FIX_CHUNKED_PROXY_SCRIPT_DOWNLOAD_KB843289
FEATURE_USE_CNAME_FOR_SPN_KB911149
FEATURE_PERMIT_CACHE_FOR_AUTHENTICATED_FTP_KB910274
FEATURE_DISABLE_UNICODE_HANDLE_CLOSING_CALLBACK
FEATURE_DISALLOW_NULL_IN_RESPONSE_HEADERS
FEATURE_DIGEST_NO_EXTRAS_IN_URI
FEATURE_ENABLE_PASSPORT_SESSION_STORE_KB948608
FEATURE_EXCLUDE_INVALID_CLIENT_CERT_KB929477
FEATURE_USE_UTF8_FOR_BASIC_AUTH_KB967545
FEATURE_RETURN_FAILED_CONNECT_CONTENT_KB942615
FEATURE_PRESERVE_SPACES_IN_FILENAMES_KB952730
FEATURE_ENABLE_PROXY_CACHE_REFRESH_KB2983228
Software\Policies\Microsoft\Windows\CurrentVersion\Internet Settings
Software\Microsoft\Windows\CurrentVersion\Internet Settings
Software\Policies
Software
Software\Policies\Microsoft\Internet Explorer
SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings
SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache
SOFTWARE\Policies\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache
FEATURE_DISABLE_NOTIFY_UNVERIFIED_SPN_KB2385266
FEATURE_COMPAT_USE_CONNECTION_BASED_NEGOTIATE_AUTH_KB2151543
FEATURE_SCH_SEND_AUX_RECORD_KB_2618444
Software\Microsoft\Internet Explorer\Main
Software\Policies\Microsoft\Internet Explorer\Main
Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad
Software\Policies\Microsoft\PeerDist\Service
Software\Microsoft\Windows NT\CurrentVersion\PeerDist\Service
System\Setup
FEATURE_IGNORE_POLICIES_ZONEMAP_IF_ESC_ENABLED_KB918915
FEATURE_ZONES_CHECK_ZONEMAP_POLICY_KB941001
Software\Policies\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap
Microsoft\Internet Explorer\Security
Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\
FEATURE_LOCALMACHINE_LOCKDOWN
FEATURE_ALLOW_REVERSE_SOLIDUS_IN_USERINFO_KB932562
FEATURE_USE_IETLDLIST_FOR_DOMAIN_DETERMINATION
Content
Cookies
History
{69DC4768-446B-4F82-A6B0-63966A243064}

Figure 4.7. Malware opens registries

This is common amongst malware, however what is noteworthy is that the malware imports almost every possible system library, spawns more threads, injects to a number of native processes, all of which render the infected host almost inoperable, which in effect defeats the purpose of remotely taking over hosts unless of course the desired object was to exhaust system resources, and thereby crashing target hosts.

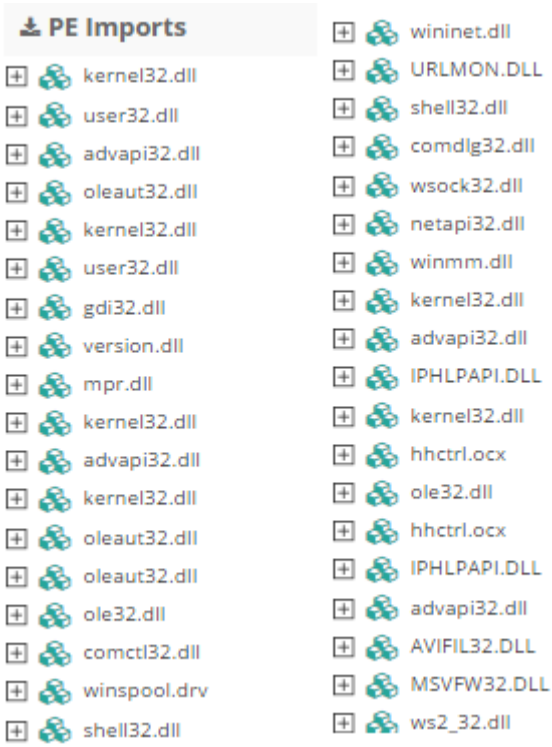


Figure 4.8. Malware .dll imports

If the latter was the desired object, then why attempt an outbound connection to a remote host? And if the former was the object, then why deplete system resource? Or perhaps neither was the intent of the author.

In carrying out further static analyses, it was discovered the malware attempts to read a set of directories on the local host including user's browsing history, session cookies, stored credentials, DNS information, subnet, local network interfaces, and routing table information,

process details, memory mapping data, proxy data, mutices, and so forth, as shown in Figures 4.9 - 4.16.

For instance, browsers tend to store user session cookies, including user credentials, on the local drive for convenience, and ease of use. Here, malware reads usernames, passwords *et cetera* of sites visited by the user(s) including web based emails, online banking, and other sites deemed of value to the attacker, as shown in Figure 4.9.

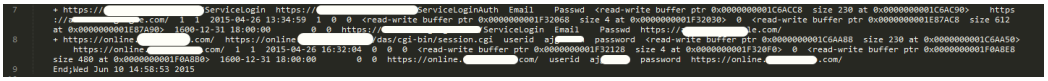


Figure 4.9. Locally stored user credentials

Equally of interest is that malware tracks, and compiles user's browser history, as shown in Figures 4.10 and 4.11.

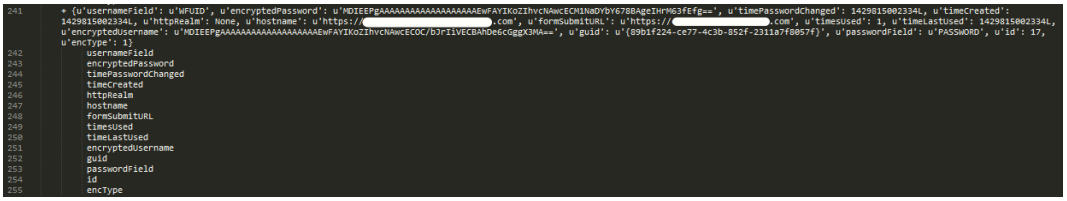


Figure 4.10. Harvesting user data through the browser

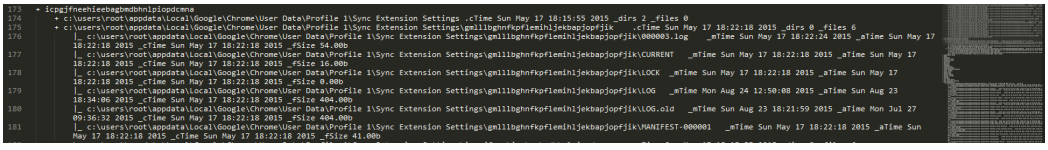


Figure 4.11. Browser history

Malware further monitors all files accessed by the user on the local drive(s) including mapped network drives, as shown in Figure 4.11.

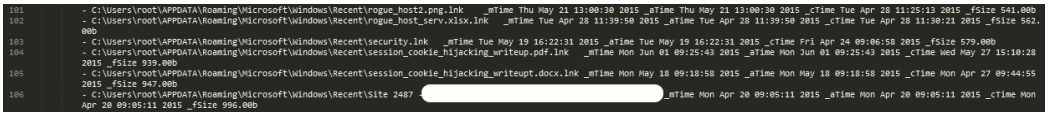


Figure 4.12. Accessed local files

For further attacks, and potential data harvesting at an organisational level, malware attempts to collect data pertaining to an organisation's **Active Directory** database schema, as shown in Figure 4.13. Attacks including stealing Domain Controllers' NTDS.dit files, which in turn allows an attacker to obtain Active Directory usernames and passwords, MS Exchange Email Server, DNS poisoning and so forth could be launched.

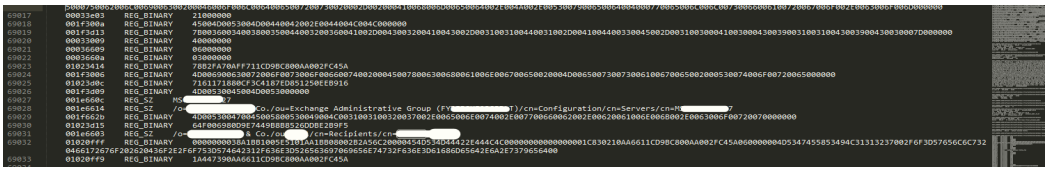


Figure 4.13. Active Directory data

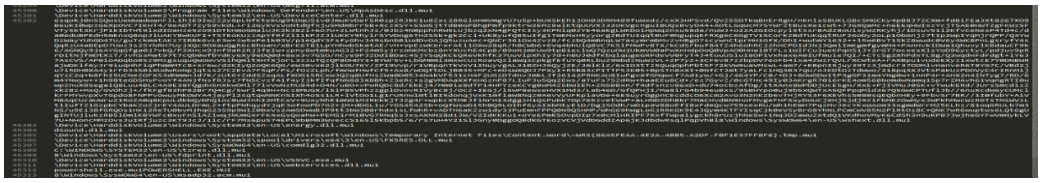


Figure 4.14. Mutices data

Of interest to the malware, is proxy configuration possibly used for data exfiltration, which the malware collects from the user's system configuration, as shown in Figure 4.15.

Figure 4.15. Proxy data

And finally, it was discovered that malware targets system memory as a source of collecting sensitive data including user credentials, and other business critical data.

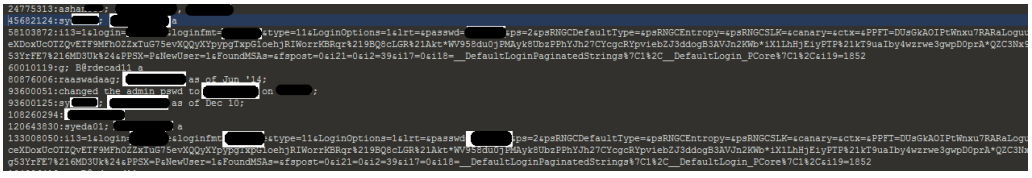


Figure 4.16. Memory analyses & data incl. logon credentials & other business critical data

Further excavation brings to the fore additional gems including local network's Active directory mapping, mapped network drives, email addresses for, any and all, users with whom infected user(s) interacted in the past, which in turn provides the attacker with a treasure trove potentially used in launching further attacks namely phishing, social engineering, or even infecting other hosts on the local network, and servers in particular for greater gains. Far more detrimental, if perhaps deleterious, is that the malware targets memory from where it accesses and gathers critical data including user credentials where administrator level user accounts with which remote code execution along with complete system takeover are targeted, and/or cloned, as shown in Figure 4.16.

The **second** unknown malware, packed with UPX, is more evasive, targets system memory, and thereby posing greater threat to the enterprise; it creates a process "services.exe" in %system32% directory spawning child processes. No outbound remote network connections were observed, nor were any local host information harvesting attempts made. The intent of the malware was understood to have been to infect the host, create a child process, imbed itself into Window's native "services.exe" process, remain hidden in memory with no visibility, and launch itself at system boot-up. Furthermore, it writes to an address space of another process whilst reading it, "services.exe" in this instance, and continues to monitor directories.

Figure 4.17. Malware sample spawns a process

If one were to compare Window's native "services.exe" file with the infected binary in its place on the target host, one could observe a few disparities including its size, MD5 & SHA-*

signatures, and other attributes where the native file is a genuine Microsoft file type with its vendor publication cum signature intact, of 400.52KB size-wise, and with the following signature (Figure 4.18):

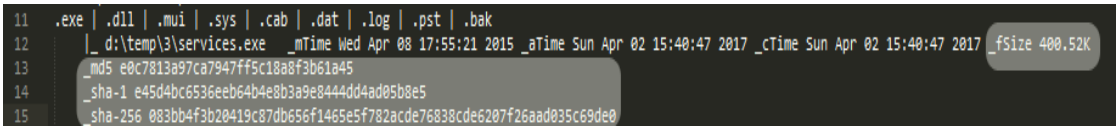


Figure 4.18. Malware signature authenticity validations

Whereas the infected binary file's properties, which clearly show the disparities betwixt the two binaries, are as follows (Figure 4.19):

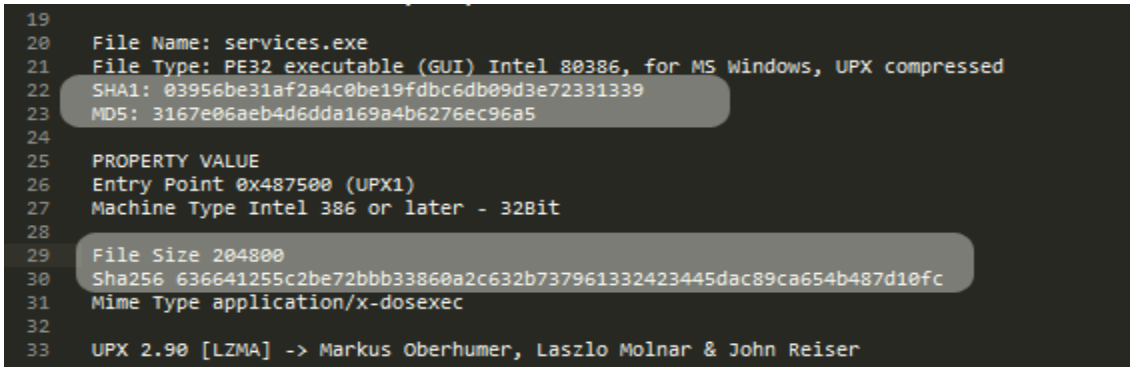


Figure 4.19. Malware signature alterations

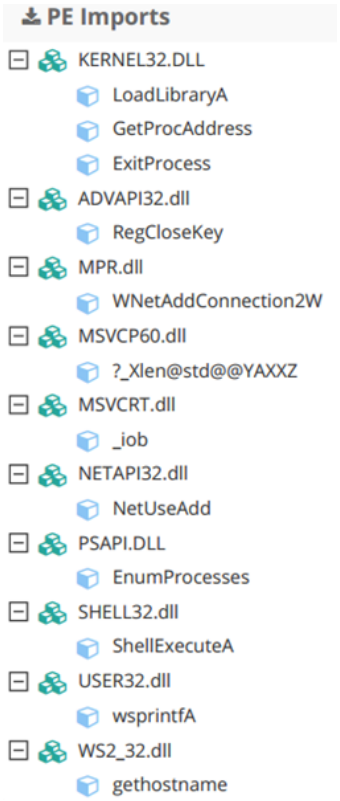


Figure 4.20. Malware PE .dll imports

An indication, this is of a native process being overtaken by a malware, and unless it has been remediated, target host shall remain infected, vulnerable to further infections and exploitations. A non-technical system administration, or a novice security professional, would be none the wiser with respect to the target host being infected, anti-virus software would not detect the infection until its signature library has been updated, and the host is being subjected to in-depth analyses using automated malware analyses tools such as the Halla tool.

As demonstrated here, the two (2) unknown binaries could not have been analysed through conventional malware analyses methods and techniques requiring closer examination, decompiling and disassembling, and the use of tools, such as the Halla tool, that could unpack or decrypt, sans being detected, lest malware corrupts itself as it detects its being analysed, or tampered with. This underscores the fact that static analyses and manual reversing is still

required, in some cases, even with automated tools, but at a much less rate and rarity than the current state.

Atop of decompiling binary files with the Halla tool, it is noteworthy each binary is submitted to five (5) malware analyses engines two of which are of open source with the remaining three being licensed commercial tools. Further, VirusTotal, an open-source virus scanning engine with numerous anti-virus engines was also used, post analyses, so as to report unknown malware signatures to the wider technical community.

Here, one could evidently observe the utility of the Halla tool where anti-virus software falls short, and as is the case herein ceases, so as to extend greater security against unknown or shielded virii and malware, and sans further analyses one is none the wiser as to the perilous nature of infection(s). As Zeltser (2015) asserts in conducting multi-layered malware reversing exercise, security corps require greater understanding of malwares' inner threading, its internal workings under the bonnet, and an insight as to its functions in each of the four (4) stages of malware reversing and analyses namely code reversing, interactive behavioural analyses, static properties analyses, and fully automated analyses [40].

In automating malware analyses, and not merely relying upon anti-virus vendors, businesses are in a better position to protect their network resources, and empower their security teams and senior management in devising befitting mitigating control measures to secure business assets, and possibly identifying blind-spots that which would otherwise not have been identified.

Whilst admittedly this is a small sample analyses purely for academic purposes, observed virii and malware trends in the business world are far more intricate, intrusive, and detrimental. One clearly observes a widening chasm betwixt conventional anti-virus software and complex

virii and malware-based attacks confined not only to businesses, but to nation state critical infrastructure.

CHAPTER 5: CONCLUSION

An existence of a security chasm betwixt the current state of protecting against virii and malware infections vis-à-vis detecting unknown virii and malware infections had been identified. A solution to remedy the aforesaid security gap was proposed in the form of the Halla security tool, which attempts to bridge the observed chasm. Other works in the field, including anti-virus software, stand-alone malware behavioural analyses tools, malware scan engines *et cetera* had been studied to evaluate the severity of the identified gap where shortcomings of traditional systems had been studied.

The Halla security tool, which comprises an agent, harvesting tools, and a reversing mechanism, was developed to enable security engineers to detect and analyse unknown virii and malware binaries where current malware engines fall short. It automates the process from collecting binaries off client hosts, via secure channel, to disassembling binaries to presenting results to alerting senior security engineers where the need for static analyses arise.

The tool not only provides malware reversing capabilities, but also offers basic memory and forensics analyses of all activities by non-native binary files written to the local disk. An in-depth memory and forensics analyses functionalities, to further advance its capabilities, is currently under development. Under research is also Kernel auditing along with kernel exploitation check enhancements to further guard against attacks including rootkits, memory scrapping, and obscure clandestine malware.

As noted, it is by no means a panacea for all virii and malware infections nor a substitute for anti-virus software; rather it is a supplemental tool to guard against malware infections, to bolster existing systems, and to further broaden an organisation’s defence edifice.

5.1. Effectiveness

A number of experiments to test as to the effectiveness and usability of the Halla tool on various platforms had been carried out in lab environments comprising live and virtual hosts of various platforms including Windows VII, VIII, and Windows X, and whilst Windows VII was the initial platform the tool was built on and for, tests on subsequent Windows releases, with the exception of Windows X, had met with minimal challenges whereas for Windows X, a complete re-building had to be undertaken with privilege escalation issues, interfacing with the dynamic engine amongst other challenges encountered.

To demonstrate its effectiveness, a sample data of thirty-five (35) binary files, collected over a period of thirty-six (36) months, had been manually submitted for analyses to anti-virus software, and to stand-alone malware analyses engines. Of those, twenty-four (24) binaries had been found to be clean posing no threat whereas eleven (11) binaries had been classified as being of malware type posing threat. Subsequently, the same binaries had been submitted to the Halla tool for analyses. Here, Halla tool classified twenty-two (22) binaries being clean posing no threat whereas eleven (11) binaries had been identified as being of malware type posing threat. However, the Halla tool identified two (2) binaries as being “unknown”, and thereby alerting senior security engineers to re-examine the said binaries; after a close examination, it was discovered, the binaries were packed or encrypted. The tool’s unpacking functionality, which is currently manual, was enabled, binaries re-submitted, and results reviewed. The revelation was that the two (2) binaries in question were of malicious nature potentially posing grave threat, including loss of, and/or damage to, data, data exfiltration *et cetera*, to the organisation, as demonstrated in the analyses chapter. Here, the effectiveness of the tools was very evident. Also

apparent was the shortcomings of the anti-virii software and malware analyses engines where packed or encrypted binaries were concerned.

The tool is being staged for pre-production deployment with a small pilot hosts being targeted; all new issues encountered then shall be documented with fixes released at the time.

Further, new builds had been created for mobile devices including Android and Windows phones with a plan for iOS phone builds in the works; tablet devices including Samsung shall also be targeted in future releases.

Furthermore, new builds for *nix systems is under test with Linux Redhat and Unix Solaris being primary targets.

And finally, forensics and memory analyses tools, initially deemed rather elementary, are being deployed as an inbuilt component of the tool.

5.2. Deficiency

In its current form, as discussed in Scenario C, an identified deficiency of the tool is that it addresses not threats posed by host(s) that which fail to communicate with the network, and where no sample is obtained for analyses in the event of an infection: a known flaw of the tool, which shall be addressed in future releases with network interfaces of hosts out of reach for a period longer than three (3) weeks disabled and barred from reconnecting to the network until a complete security assessment has been undertaken. Issues concerning user frustration along with delays, which might adversely influence productivity, are predicted, and could be addressed in security orientation sessions.

Whilst the tool has unpacking and decrypting functionalities, its earlier versions including versions v1.0 and 2.0, relied upon manual unpacking or decrypting of binaries, and re-submitting them for analyses; the whole process is automated in later versions starting with version v3.0.

5.3. Future work

Future enhancements under development include deeper forensic and memory analyses along with Kernel auditing and exploitation checks, which are currently under research and development.

Further, the tool shall be compiled for mobile devices including Android and Windows phone X, and whilst environment configuration for mobile devices had been found rather challenging, rooted and jailbroken devices are under scope, with some teething problems encountered. Versions for *nix platforms are currently under development with Kali, Redhat, and Solaris being primary candidates.

And finally, an Open Source version is being considered in future releases.

REFERENCES

[1] C. Eoghan, C. H. Malin, and J. M. Aquilina, *Malware Forensics.* Syngress, 2008.

[2] L. Zeltser, *SANS course, Rever-engineering malware course.* SANS, 2010.

[3] D. Oktavianto, and I. Muhardianto, *Cuckoo Malware Analysis*. Packt Publishing, 2013.

[4] T. Seals, (2015, December 16). *80% of Companies Had a Security Incident in 2015*. [Online]. Available: www.infosecurity-magazine.com/news/80-companies-had-a-security.

[5] F. Ehsam, *Detecting unknown malware: Security analytics & memory forensics*, RSA Conference, San Francisco, USA, 2015.

[6] F. M. Halvorsen, R. W. Nergard, and H. Vegge, *Zero-day Malware*. Project Assignment, Faculty of Information Technology, Mathematics and Electrical Engineering, Dep. of Telematics, Norwegian Univ., of Sc., and Tech., Trondheim, Norway, 2008.

[7] VirusTotal antivirus scan engine aggregate, 2016. [Online]. Available: virustotal.com.

[8] L. Bass, P. Clements, R. Kazman, *Software Architecture in Practice*. Addison-Wesley Professional, 2013.

[9] N. Rozanski, *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional, 2011.

[10] R. Garofalo, *Building Enterprise Applications with Windows Presentation Foundation and the Model View ViewModel Pattern (Developer Reference)*. Microsoft Press, 2011.

[11] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall, 2004.

[12] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Professional, 2003.

[13] K. Qian, X. Fu, L. Tao, *Software Architecture and Design Illuminated*. Jones & Bartlett Learning, 2009.

[14] J. Mishra, A. Mohanty, *Software Engineering*. Pearson, 2011.

[15] J. Bach, and M. Bolton, *Oracle Heuristics*, 2010. [Online]. Available: www.testingeducation.org/BBST/foundations/OracleHeuristicsLab6b.pdf.

[16] M. Bolton (2005, January). *Testing Without a Map*. [Online]. Available: www.developsense.com/articles/2005-01-TestingWithoutAMap.pdf

[17] E. Eilam, *Reversing: Secrets of Reverse Engineering*. Wiley, 2005.

[18] F. Ankit, *The Unofficial guide to Ethical Hacking*. Cengage Learning PTR, 2006.

[19] C. Elisan, *Advanced Malware Analysis*. McGraw-Hill Education, 2015.

[20] M. L. Hale, A. Case, J. Levy, A. Walters, *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*. Wiley, 2014.

[21] K. R. Van Wyk, D. S. Peters, M. G. Graff, D. L. Burley, *Enterprise Software Security: A Confluence of Disciplines*. Addison-Wesley Professional, 2014.

[22] H. J. Highland (1997). *History of Computer Viruses - The Famous 'Trio'*. [Online]. Available: www.pdfs.semanticscholar.org/475d/3110b70bb3d0fcc8ee75534607324960ade6.pdf

[23] C. Cerrudo (2014), *Hacking US Traffic Control Systems*, 2014. DefCon. [Online]. Available: www.defcon.org/images/defcon-22/dc-22-presentations/Cerrudo/DEFCON-22-Cesar-Cerrudo-Hacking-Traffic-Control-Systems-UPDATED.pdf.

[24] S. E. Kecmer (2014). *A Curious Cyber War: Business Owners vs. Investors*. Infiltrate Conference. [Online]. Available: www.infiltratecon.com/archives.

[25] The Hague Centre for Strategic Studies, *Assessing Cyber Security: A meta-analysis of Threats, Trends, and Responses to Cyber Attacks*, 2015.

[26] Kaspersky Lab Report (2014, February). *Financial cyberthreats in 2014*. [Online]. Available: www.securelist.com/files/2015/02/KSN_Financial_Threats_Report_2014_eng.pdf.

[27] D. Kushner (2013, February 26). *The Real Story of Stuxnet: How Kaspersky Lab tracked down the malware that stymied Iran's nuclear-fuel enrichment program.* [Online]. Available: www.spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet.

[28] OWASP (2015). *OWASP Reference Guide*. [Online]. Available: www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide.

[29] J. Bach, *Blackbox Software Testing: Bug Advocacy*. Association for Software Testing, 2010.

[30] Symantec Security Response (2014, November 20). *POS malware: Potent threat remains for retailers*. [Online]. Available: www.symantec.com/connect/blogs/pos-malware-potent-threat-remains-retailers.

[31] Viega, J, and Messier, M. *Secure Programming Cookbook for C and C++*. O'Reilly Media Inc., 2003.

[32] Mandiant (2017). *M-TRENDS 2017: A View from the Front Lines*. [Online]. RSA Conference. Available: www.files.shareholder.com/downloads/AMDA-254Q5F/0x0x938351/665BA6A3-9573-486C-B96F-80FA35759E8C/FEYE_rpt-mtrends-2017_FINAL2.pdf.

[33] N. Ismail (2016, October 26). *28 years later: the malware landscape in 2016*. [Online]. Available: www.information-age.com/28-years-later-malware-landscape-2016-123462883/

[34] J. Leary (2016, December 16). *The Biggest Data Breaches in 2016*. [Online]. Available: www.identityforce.com/blog/2016-data-breaches

[35] G. J. Myers, *The Art of Software Testing*. Wiley, 2004.

[36] RSA Research Group, *RSA Report: Bolware – Onyx Variant*, 2014. RSA Conference 2014, San Francisco, CA. [Online]. Available: www.slideshare.net/emcacademics/rsa-report-bolware-onyx-variant.

[37] SANS Internet Storm Center (2002 - 2016). [Online]. Available: www.isc.sans.edu

[38] H. F. Tipton, and M. Krause, *Information Security Management Handbook*. CRC Press, 2003.

[39] Kaspersky Lab (2016). *Kaspersky Security Bulletin 2016*. [Online]. Available: www.kasperskycontenthub.com/securelist/files/2016/12/KASPERSKY_SECURITY_BULLETIN_2016.pdf.

[40] L. Zeltser, *SANS course, FOR610 Reverse-Engineering Malware: Malware Analysis Tools and Techniques course.* SANS, 2015.