Survey of Automated Malware Identification Systems

Eugene Davis
University of Alabama in Huntsville
Electrical and Computer Engineering Department
Alabama 35806
eugene_davis@eugenemdavis.net

ABSTRACT

One of the trends in the security world of the twenty-first century has been an explosive growth in malware. In the last half of the twentieth century researchers could manually perform analysis on anti-virus and produce signatures that allowed malware's detection on end-user systems, but in the first decade of the twenty-first this began to change. Now anti-virus vendors must handle millions of samples of suspected malware a year, which all must be analyzed to detect if they contain malware, and if containing malware have new signatures generated to match for deployment on customer's systems. To make this problem harder, there are millions of good programs which may exhibit malware-like behaviors, such as the digital rights management systems on many games and media. To continue successfully combating the increase in malware, there has been an increasing reliance on automated systems which can identify if a sample is malware and create a signature to match future instances of the malware. Better yet are systems which are capable of identifying the family of malware from which the sample came, in order to help with its future disinfection and classification. This paper surveys some of these automated techniques for detecting and identifying malware.

Categories and Subject Descriptors

K.6.5 [**Operating Systems**]: Security and Protection—*Invasive Software*; I.2.1 [**Computing Methodologies**]: Artificial Intelligence—*Industrial Automation*; I.2.8 [**Computing Methodologies**]: Problem Solving, Control Methods, and Search—*Heuristic Methods*

General Terms

Security

Keywords

Malware, Automation, Machine Learning

1. INTRODUCTION

The amount of malware in the wild has seen a huge proliferation in the early twenty-first century, especially when compared to the end of the twentieth century.[13] Where malware used to be created out of a desire to experiment with systems, and a desire to spread fame as a malware author, modern malware is driven by an underground economy, and is targeted towards making money through the extraction of personal information such as banking passwords or through the control of massive computing resources, as with botnets.[12] The upshot of this transformation is that malware researchers are swamped with malware to analyze. Vendors of commercial anti-virus software in particular must be able to rapidly analyze samples to identify if they contain malware, and then create and distribute signatures that will reliably detect the malware on consumer systems, without flagging any legitimate software as being malicious. To give a concrete idea of the scale of incoming samples, McAfee Labs reports that as of the first quarter of 2013, they have received 128 million samples since they began operating, with approximately 15 thousand in that quarter alone.[1]

In order to attempt to keep up with this backlog, researchers have turned to automated systems to do some of the initial work of detecting malware within samples, and identifying what malware family it belongs to. In the academic arena, these systems usually just try to detect new malware samples, or identify what family a sample may belong to. In the commercial arena, these systems perform identification, but also attempt to generate signatures that can be distributed to the vendor's customers to identify each strain of malware.

A number of approaches attempt to solve the problem of automatically analyzing malware. There are two major categories of systems, the first being static analysis systems, which look at the machine language or assembly directly, and attempt to identify malware without ever running the suspect code. The second is dynamic analysis systems which run the suspect code in order to trace the code as it runs or observe its behaviors.

Automated malware analysis also faces a major challenge. Even in the face of changing and heavily obfuscated code, they must be capable of identifying a particular strain of malware, without accidentally identifying good software as bad. This challenge is made intense by the increasing use by malware authors of artificial intelligence techniques to obfuscate their code and make the behaviors of the malware look like those of legitimate software.

Automated malware analysis relates to artificial intelligence because it uses techniques from machine learning and machine reasoning. It also has great emphasize on heuristics, which allow identifying obvious traits common to malware at low computational cost.

This paper focuses on surveying the heuristics and machine learning and reasoning techniques used in commercial and research automated malware analysis systems, and how those can be applied to detect new and existing forms of malware, and where possible generate signatures to allow end user systems to be protected by anti-virus. The topic of exactly what composes signatures is left for future research.

2. REVIEW OF MATERIALS

Dube, et. al.[14] covers a number of means by which to perform static analysis, in particular it discusses a variety of heuristics which can be applied during static analysis to provide evidence that the sample is malicious. Griffin, et. al.[3] provides additional heuristics as well as a good explanation with regards to applying Markov chains to static analysis, all as applied to the commercial analysis system Hancock. Lakhotia, et. al.[5] provides more insight into detecting malware despite obfuscation, and briefly talks about Bloodhound, an automated hybrid static and dynamic analysis system.

Lin and Stamp[7] also discuss Markov chains, as well as other approaches to static analysis of malware. Wong and Stamp[16] revisit the discussion, and briefly discuss why low false positive rates are so important.

McAfee Labs[1] provides a report with the figures for the number of samples they have received in the first quarter of 2013.

Nascimento[10], et. al. discusses the use of neural networks to aid code porting and reconstruction, including the automatic creation of control flow diagrams, something potentially useful in malware analysis.

Luger[8] covers basic artificial intelligence technology, and in particular provides a description of Markov chains. Szor[13] provides a recent history of viruses, as well as definitions of some of the more obscure obfuscation techniques like polymorphism and metamorphism. Skoudis[12] also provides information about malware, and is focused on malware in general, unlike Szor's focus on viruses and worms.

Denning[2] gives the basis for an intrusion detection system which models user behaviors and examines anomalies based on that profile, while Mutz, et. al.[9] and Lanzi, et. al.[6] build intrusion detection systems based on modeling system calls to improve the ability to detect real anomalies.

Wei, et. al.[15] covers the identification of botnets via behavioral clustering applied to network packets. Perdisci, et. al.[11] also builds on the same area, giving more detail on identifying which family the botnet came through by examining packet structure. Kuochen, et. al.[4] also builds on the same topic, but utilizes fuzzy logic in pattern recognition to identify botnets attempting to make their traffic appear human-like.

3. BRIEF INTRODUCTION TO MALWARE

Though this paper's focus is not on defining or describing malware, but rather the systems performing automated analysis on malware, a basic knowledge of malware is necessary for understanding the systems in question. This section will provide a brief introduction to malicious software to provide the necessary context.

Malware, an abbreviation for malicious software, is software which intends to do damage to a victim. In the past the damage was usually simply to the files, software or hardware of the victim's machine, but in the present is often targeted to extracting money from the victim. The terms malware and malicious software will be used interchangeably throughout the paper.

3.1 Types of Malware

This section will briefly define the basic types of malware.

3.1.1 Virus

A virus is a type of malware which can autonomously infect many files on a victim machine once introduced to the system. Note that it cannot transmit itself to a new machine autonomously, requiring human intervention (with the exception of transmitting over file shares).[13]

3.1.2 Worm

A worm is malware similar to a virus. A worm may move from networked computer to network computer.[13] For a worm, a computer is analogous to a file to infect for a virus.

3.1.3 Trojan

A Trojan is malware that masquerades as legitimate software. It tricks the victim into installing it, then attacks, often providing a rootkit.[12]

3.1.4 Backdoor

A backdoor is malware that provides access (usually remote) to a computer for the attacker.[12]

3.1.5 Rootkit

A rootkit is malware that conceals itself within the target system. Rootkits can either work at a user level, in which case they replace binaries that users have access to, or they can work at a kernel level. Rootkits are key to much of the modern success of malware, as they can prevent anti-virus from detecting them, and demonstrate one of the major barriers to dynamic analysis.[12]

3.1.6 Hybrid Malware

Hybrid malware further muddies the waters of malware analysis. Hybrid malware combines two or more other forms of malware into a new type. A typical example is a botnet, which may be distributed as a Trojan, while also propagating with a worm, use a rootkit to hid itself, and providing a backdoor to provide the actual bot functionality. Hybrid malware demonstrates problems in actually classifying malware, especially with static analysis which does not show its behaviors.[12]

3.2 Anti-Reverse Engineering in Malware

This section will briefly look at how malware fights analysis efforts. Many of these defenses present major challenges to automated (or manual) analysis.

3.2.1 Static Analysis Protection Techniques

Obfuscation varies in its level of application, but at is most basic hinders efforts to understand code. Obfuscation is better against human researchers than automated systems. Similarly, code encryption encrypts the body of the malware code, so that it cannot be directly read. Since the key must be included in the malware, this generally has limited effect, and sends a signal that something is suspicious in the code.

Polymorphism brings protection to a new level. In polymorphism, the code is encrypted, and ever time a new infection is created, the malware recreates its encryption engine in a new form, so no two malware samples will be encrypted in the same way.[13] Metamorphism poses the major problem to repeatably identifying malware. Metamorphic malware rewrites its code during every install, greatly reducing the number elements of itself that can be included in a signature.[13][12]

3.2.2 Anti-Debugging Techniques in Malware

Modern malware authors deploy a number of technologies to prevent debugging of their software during dynamic analysis. At the most basic level, INT3 instructions are inserted into the code to trigger the debugger, and the reaction of the computer is monitored. Other techniques include run a second process to look for evidence of a debugger and shut it or the program down. More sophisticated are red pills - a technology taking its name from a scene in the movie "The Matrix", a red pill attempts to detect if it is running on an emulator by running unusual instructions that crash many emulators. Since emulators are rarely capable of reproducing the entire instruction set of a given architecture, red pills have proven quite effective.[13] When running on the same machine as the dynamic analysis system, these may prevent any real progress from being made.

4. STATIC ANALYSIS SYSTEMS

Static analysis relies on examining machine code or decompiling code into assembly in order to discern its purpose or intent. Static analysis is common for commercial malware analysis because it is the quickest approach, which allows a single system to rapidly analyze many malware samples.[3] Static analysis also reduces containment concerns, since it never runs potentially virulent samples. Unfortunately, even as malware researcher's techniques improve, the malware author has access to a continually improving arsenal of anti-reverse engineering techniques including packing of the binaries, polymorphic code, metamorphic code, as well as camouflage techniques designed to make malicious code look innocuous.

Although automation cannot significantly improve upon the abilities of malware researchers it is capable of significantly expediting the process. Using a variety of heuristics as well as utilizing machine learning technologies an automated system is able to detect and categorize far more malware samples than any number of human researchers would be capable of handling. Most commercial systems belonging to anti-virus vendors are capable of generating signatures which uniquely identify a piece of malware. In particular static analysis must be able to identify a sample as malicious or not, even if unable to classify what type of malware it is.

This section will use the Hancock system at Synamtec which automatically generates signatures for new malware samples for a real world example of a static analysis system.

4.1 Markov Chain in Static Analysis

Most automated malware analysis systems use Markov models to model the suspected malware code that they are examining.

Markov models are probabilistic models of events or sequences. The model consists of Markov chains, which model the probability of a particular sequence of states occurring where only the probability of the previous state in the sequence is considered for the probability of the current state.[8] For most malware analysis, the type of Markov model used is an n-gram model. A gram is the unit being analyzed (it's state).

4.1.1 Markov Chains Applied to Malware Code

For static analysis, each byte of machine code becomes a gram. The system then applies an n-byte sized sliding window across the entirety of the malware, and makes Markov chains out of the entirety of the sample. Chains that repeat become identified as common, and need to be stored only once. Further pruning must be done in order to accommodate the storage of all the chains for a large number of samples, this pruning is generally system specific.[14]

4.1.2 Modeling legitimate software to Reduce False Positives

Unfortunately, directly applying Markov chains gathered from malware to a new sample to determine if it is malware is an approach doomed to failure. All software under a given architecture is bound to share many commonalities at the machine code level, be it legitimate or malware. Even if the system can be tuned enough to successfully reject most legitimate software, making it useful for a researcher, anti-virus vendors require an extremely low false positive rate or they risk loosing customer trust in their product.[3]

The solution to this is to create a database of legitimate programs. Using this database of Markov chains, it becomes possible to select candidate chains from samples, and only accept chains that are very rare within the legitimate software database as candidates to identify the malware. Since Markov models can be generated in a reasonable amount of time for a reasonable amount of processing power, very large legitimate software databases can be used to decrease the likelihood of false positives dramatically.[3]

It is important to note that even in cases where the chain is acceptable from the perspective of not being in the legitimate software database, polymorphic and metamorphic malware further complicate the issue. The chain must remain constant across each variation of the malware, or else it has limited usefulness as the metamorphic malware simply rewrites itself on each new victim.[7]

4.1.3 Markov Modeling in Hancock

The the Synamtec automated system for malware analysis, Hancock, utilizes Markov chains in order to find chains that could be included in signatures for distribution to their clients. As Synamtec is a commercial anti-virus vendor, they must have a low false positive rate as well as a high success in malware identification in order to retain their customers, as mentioned in the previous section.

Hancock utilizes a very large database of legitimate software to generate its legitimate software chains, on the order of several terabytes. In order to scale to this to a large size while retaining usability, they identify areas of high-information byte sequences in the program and capture chains only off of those sequences. In order to keep their false positive rate under one percent, they only accept byte sequences that are very rare, that is occurring in a tiny fraction of their legitimate software database, as potential signatures for identifying malware.[3]

The actual Markov chains in Hancock are 5-gram, but the generation of chains also includes generating grams smaller in order to have more granular sequences as well. In order to support all of these chains, Hancock prunes chains based on their amount of information gained from it, if it gains little information then the chain is cut. This pruning is done after the creation of the model, in order that the relative information of the chains can be calculated. Since these models are quite large in comparison to the sample they represent, this results in memory management issues, resolved by modeling only small portions of the data and pruning them individually before merging them into the final model.[3]

4.2 Heuristics in Static Analysis

Though the fight against malware sees a constant evolution in the techniques used by malware authors, several decades of experience from malware researchers still has a place in automated analysis in the form of heuristics. These heuristics identify basic things that are in common amongst many types of malware.

One major concern when using heuristics that they often have a tendency to false positives. This problem prevents heuristics from being the only part of an automated malware analysis system.[16] In general, heuristics seem to be useful for increasing the suspicion that a sample is malicious, but play a secondary role to Markov chains.

4.2.1 Obfuscation and Camouflage

Most legitimate software does not have convoluted mathematical instructions. Malware on the other hand often include things like many XOR operations, ADD operations, and more, generally to provide for some obfuscation or encryption to the main body of the malware code. As a result, systems like Hancock include a heuristic to check for these types of operations.[3]

In addition to noting mathematical operations, another heuristic can be applied to detect encryption. Since many forms of malware encryption are XOR encryption, they can easily be detected when examining the code using a technique developed by Eugene Kaspersky.[13] Better forms of encryption can be spotted by applying a rule to look for random data in the file.

4.2.2 Entry Points, Imports and Exports

A number of heuristics look at the basic structure of a suspect binary file to add support to the belief that the sample is malicious.

Entry points are the first of these, viruses and similar types of malware will often infect an executable and place the virus body within a non-code portion (usually the data segment) of the executable. Initially virus writers used this approach because older scanners only looked at the code portions of an executable. This is now longer the case, and including this heuristic in an automated analysis system provides a very low false positive rate when it is triggered.[14][13]

Much like entry points, imports provide an excellent area to apply a heuristic. While most programs employ a glut of libraries, especially when performing common activities, malware will often skip their import, since it often spreads via a low bandwidth channel or has limited space available to store its libraries.[14] Conversely, heuristics can also be employed to look for combinations of libraries common to malware, for instance looking for encryption libraries and networking libraries could help indicate a worm.

For a final structural heuristic, analysis systems can also look at the exports of a binary. If a dynamically-linked library fails to provide exports, this is almost a sure sign that it is a piece of malware.[14]

Structural heuristics are very useful for inclusion in an automated malware analysis system. Unfortunately, as malware becomes more sophisticated, the most dangerous malware becomes less likely to be detected by these heuristics, as malware authors intentionally avoid triggering them. However, a significant chunk of malware being created is written by inexpert authors, so these heuristics are still valuable for inclusion in an automated analysis system.

4.2.3 Simple Heuristics for Repeatable Identification

One of the major requirements for most commercial automated malware analysis systems and some research systems is the ability to repeatably identify malware. This requires the ability to identify certain characteristics that can be included in a malware signature, and distributed to clients for detection of the malware on general purpose computers.

Looking for unusual address offsets provides an easy mechanism for identifying specific malicious software. Since a large offset from the base pointer usually indicates a large data structure, it is unlikely that a good program and a malicious program will share the same offset. By comparing the offset to offsets in a database of good software it may become useful for inclusion in a malware signature.[3]

Local function calls also are useful for repeatably identifying a piece of malware. Since these function calls are not shared with good software, unlike library calls, the setup for the function call may be unique to the piece of malware.[3] As a result, local function calls are potentially useful for inclusion in malware signatures.

4.3 Neural Network Static Analysis

Neural networks can be trained to convert assembly code into control flow diagrams.[10] This technology can be useful for attempting to understand malware, but in the end virus writers utilizing trying to obfuscate their code can probably still prevent control flow from being of much help to identify malware. However, neural nets can be of help to detect certain varieties of malware, and recognize malware from the same family, as described below.

Neural networks can also be trained to recognize families of malware with a very low false positive rate. This technology has been integrated into the IBM AntiVirus which has a false positive rate under 1% and successfully detected three-fourths of all boot sector viruses since release [16]. Unfortunately, neural nets show little aptitude at detecting genuinely new malware, and thus tend to be left out of newer systems, such as Hancock.[3]

4.4 Limitations and Benefits to Automated Static Analysis of Malware

Static analysis survives in both manual analysis and automated systems due to the safety factor it adds by never running the malware. It also is very capable at generating signatures that can be distributed to detect the malware in the future. However, as it cannot see the results of running code, it must fight against the non-deterministic nature of disassembling code, large code bases, obfuscation techniques, and ends up being most capable at identifying related pieces of malware rather than identifying wholly new malware.

5. DYNAMIC ANALYSIS SYSTEMS

Dynamic analysis is the counter part to static analysis. Unlike static analysis, automated dynamic analysis systems do not just focus on repeatably identifying malware, but also on detecting the presence of malware compared legitimate human presence. Techniques used in dynamic analysis range from debugging to utilizing anomaly detection systems. Dynamic analysis using anomaly detection benefits from research into intrusion detection systems, which often also depend on anomaly detection.

5.1 Anomaly Detection on Host

Anomaly detection on a host is a system which resides on the same machine as it wants to monitor. This allows it to detect changes in the behavior in the system, and identify malware or at least its presence.

One approach to detecting the presence of malware is to model user behaviors on the host system. In this case, profiles of user access to resources, their login time, privilege level, and other useful factors are built up. The data collection can either take place in real time, or by scanning audit logs containing the same information. The user profile can be modeled a number of ways, a common approach to this appears to be through the use of Markov models. Assuming a clean initial state, this allows the detection of sudden changes in behavior that are ascribed to a particular user, and may be due to malware operating on their behalf.[2] By creating a profile for how malware behaves, it becomes possible to identify something as malware, rather than a generic intrusion attempt.

Unfortunately, using user activity has a major pitfall, mainly in that humans fail in repeating themselves as predictably as a computerized detection system might like. However, through using dynamic analysis it is possible to hook function calls and determine the system calls being used by software. As it turns out, even though exactly what a human may do with a system is erratic, the system calls used during normal usage are more reliable. One approach to building up the profile of normal system calls is through using a Bayesian network to simply classify each individual call as normal or anomalous.[9] A more sophisticated, and potentially reliable approach is to use Markov models, described in the AccessMiner subsection below.

5.1.1 AccessMiner - Using Markov models and System Calls

AccessMiner is an experimental anomaly detection system that checks system calls to discover anomalies. It utilizes a large database of system calls collected across several Windows systems used in various environments. Much like in static analysis of code, AccessMiner used the n-gram Markov model. Instead of using bytes as a gram, it uses system calls as a gram, allowing it to track not only how unusual a specific system call is, but also how unusual a specific sequence of system calls is.

Having gathered a database of normal system usage on the order of 100 gigabytes, AccessMiner was able to identify unusual system call chains by comparing them to the database of normal system calls, and thus better identify anomalous behavior.[6] By extending this approach with a larger database of normal behavior on known clean systems, and further training it with malware sets in a separate database, this system has the potential to be able to classify malware within families or even determine the type of malware based on the system call chains. At the moment, the authors of AccessMiner recognize that it could potentially be used in the malware classification area, but suggest that it would have short comings and may identify too many legitimate programs as malware.

5.2 Botnet Detection on Network Based Systems

Network based anomaly detection sits on the network rather than the host infected. Just as for host based systems, network anomaly detection systems are used for more than just detecting malware, but can be used for that purpose. A network based system has the advantage of being separated from the host, and thus not subject to being manipulated by the malware, but at the same time is incapable of detecting forms of malware that do not use networks, such as viruses, or even backdoors, which can be hard to distinguish from legitimate remote access protocols. However, one of the major new types of malware of the past few years, the botnet, is very susceptible to detection from network detectors.

5.2.1 Behavior Clustering for Botnet Detection

One approach to detecting botnet traffic is through behaviorial clustering, which is an approach to classifying things via their statistical similarities. For detecting protocols that are being used by malware, the packets are examined, and their structural characteristics are used to cluster them. In order to handle the large amount of traffic, the network flow cannot be clustered directly. Instead, two steps are taken to allow scalability. First there is a course-grained clustering step, which looks at easily discernible characteristics such as size, number of packets, and material in the packet headers. This generally can identify malware, but may not be able to pin down a given family for the botnet.[11] The second step is fine-grained clustering. which works with the groups that the first pass created. This step examines structural characteristics of the packets to discover differences between different types of malware.[11] Because it naturally discovers what makes the botnets unique, behavioral clustering allows for the creation of signatures that can be used in generic instrusion detection or prevention systems.

Clustering can be done with a number of different mechanisms, which can be used in conjunction with each other for better results. The first approach is a simple, signatures-based classifier, which is basically a rule driven system. In this approach, basic characteristics of packets are examined in order to identify them as packets of interest. This approach works best as part of the course-grained pass, since no signatures exist when doing the classification of an unknown botnet. Signatures can, however, help to identify potentially interesting traffic.[15]

Markov chains, once again in the form of n-grams, can also be used. Clustering these behaviors is an obvious application of a Markov chain, because it even has a temporal component, making the sequence of states clear. The temporal component is an important factor in detecting botnets this way, because it can be used to distinguish a botnet from a human. Where a human, being slow compared to a CPU, will always take some time before they respond to any given piece of information that has been received over the network, bots communicating with their command and control almost invariably respond instantly. Generally, each gram will be composed of a packet. Once enough packets have been gathered, the chains can be compared against a database of normal traffic, and other identified malware traffic to attempt to identify the botnet in question.[15]

It is worth noting that these signatures, unlike signatures generated by all previous mechanisms, are capable of detecting malware without ever examining the code of the malware, or directly observing its behavior on its victim.

5.2.2 Fuzzy Pattern Recognition

Fuzzy pattern recognition technology has been applied towards tracing the command and control servers of a botnet network, as part of an effort to disable a botnet network. The creators of this approach first recognized that botnets often generate failed DNS queries and failed network flows due to incompatibilities in their current network. Additionally, they took into account the fact that DNS queries repeat upon failure at the same rate across all bots on the network, and that they generate packets with payloads of about the same size. By applying fuzzy logic to each to the results of these heuristic as they are triggered, numbers can be assigned to how closely a botnet matched each of the expect behaviors. From the resulting numbers, it can be decided if the behavior is a human behavior or a botnet behavior.[4]

One important benefit of using fuzzy pattern recognition is that it helps detect malware attempting to make its activity appear human-like. This counteracts a major defensive technique against dynamic analysis, making the behaviors seem legitimate, assuming it can be applied well enough to detect human-like malware.

5.3 Limitations and Benefits to Automated Dynamic Analysis of Malware

Dynamic analysis can actually observe behaviors to discover things that seem malicious. As a result, dynamic analysis can be very good at detecting the presence at malware, but can only rarely create signatures that allow the detection of the malware before it is actually run. Also, due to the fact that it takes time for the malware expose all its behaviors while running, dynamic analysis is more time consuming, and generally more resource intensive than static analysis, thus limiting its adoption on commercial analysis systems.

6. RESULTS

Malware analysis for identification has become a major industry due to the rapid growth in malware in the wild. This paper surveys the efforts that have been made to automatically perform the analysis. None of the techniques covered seems adequate to overtake the improvements in writing malware that have been occuring, so it is likely that problems in rapidly analyzing and responding to malware will continue. Additionally, neither dynamic nor static anaylsis seem to be capable of identifying truly new malware without human intervention.

6.1 State of Automated Static Analysis

Automatic static analysis seems to be in a more advanced state than does dynamic analysis. This is a result of the drive in the anti-virus industry to handle the huge influx in malware that has occurred in recent years, since the industry cannot afford the time to perform dynamic analysis on every piece of malware. Additionally, static analysis yields signatures that allow the customer to scan the file before ever running it, reducing the risk that malware will have the opportunity to infect their system.

Automatic static analysis mostly focuses on the creation of Markov chains to examine the machine code and compare it to a database of legitimate software chains, as well as using heuristics to identify characteristics of samples that are suspicious. By combining the evidence presented by heuristics and Markov chains, static systems are able to identify new malware or existing (but mutated) malware in samples with a very low false positive rate.

Overall, automated static analysis will likely continue to be a major force in automated analysis of malware. Unlike dynamic analysis, it does not rely on creating an emulator (which can be detected by malware) or actually infecting real machines or virtual machines, but can be utilized without the risk of an escaped sample. The major disadvantage to performing static analysis is it makes it is impossible to tell for sure what the behavior of a sample will be without running it. Thus being able to detect if a sample is malware depends on how well the code was obfuscated or mutated from its original form. Automated static analysis can also rarely classify the malware into a particular category of malware, unless it is able to locate the family from which the malware is from.

6.2 State of Automated Dynamic Analysis

Automatic dynamic analysis remains the younger brother of static analysis. Although dynamic analysis is a powerful tool for identifying and classifying malware, the additional risk of escape, the fact that the signatures it creates cannot be compared to a suspected file without running it, and the additional cost of running each sample prevents its entry into the mainstream anti-virus analysis systems, and leaves it mostly to the realm of academics.

Most automated dynamic analysis focuses on anomaly detection on the host machine. The initial approach to this was to create profiles of user behaviors, but as humans are not as consistent as software, this approach has mostly given way to function call driven detection, especially to examine the system calls performed. Since malware often uses sequences of system calls that do not appear in normal computer usage, this can be a very successful method of detection. Unfortunately, as mentioned before, this approach still may expose the user to malware.

It should be noted that there is a major exception to the low adoption of dynamic malware analysis systems, in the form of detecting botnets. By utilizing the existing research into anomaly based intrusion detection systems, automated systems have been created to detect and identify the family of botnet via network traffic. This technique appears to be in use in some commercial intrusion detection systems currently, and certainly is a well researched area.

Overall, the dynamic analysis approaches are best left in the field for usage as intrusion prevention systems, which can clamp down on the dangerous behaviors before they can actually do damage. For identifying particular malware, they struggle more, being able to identify that some form of malicious activity is occurring, but only rarely able to identify that it is a specific family or type of malware.

7. FURTHER RESEARCH

Since this paper was a survey, there are many areas of interest that could be of interest for further research. The sections below cover the most notable of them.

7.1 Commercial Automated Analysis Systems

Although Synamtec has released some information about systems like Hancock, most of the other anti-virus vendors seem to keep the information about their systems quieter. Researching systems backing the likes of Kaspersky or McAfee may yield information about new techniques that are superior to or complement Synamtec's.

7.2 End User Anti-Virus

Although much of the heavy lifting for analysis is done on systems at the anti-virus vendor's headquarters, most anti-virus software contains techniques to uncover suspicious activity, blending host-based anomaly detection system into a rule matching system. More time could be spent researching these tools to understand how they are able to detect new malware independently. This also includes how signatures are generated and applied.

7.3 Hybrid Systems

Systems which use both dynamic and static analysis appear to exist, for instance Synamtec mentions its Bloodhound technology delivered as part of its anti-virus, which they describe as using both static and dynamic analysis to identify files containing new types of malware. In Bloodhound, the code is actually run within a sandbox, so that its system calls can be analyses.[5] Most articles limit their focus to either dynamic or static analysis, but since both have strengths and weaknesses that complement each other, using a hybrid approach seems to have value.

7.4 Information Tracing

Since modern malware is focused either on retrieving information and sending it back to its master (such as passwords and other personal information), or focused on controlling the computer for the gain of the attacker, tracing the flow of information throughout the execution process can yield information about the intent of the malware. Although this technique is used in manual analysis, and it is suggested in various papers cited in this paper, there appears to be little information about using it as part of an automated system.

7.5 Applications of Fuzzy Logic

The only source found using fuzzy logic related to using it to classify network traffic to identify botnets. Since this paper suggested that fuzzy pattern recognition allow the system to detect botnets which attempt to made their actions look like human actions, it should be possible to apply this to other forms of analysis.[15] Applying fuzzy logic to static code heuristics may have use in detecting malware using new methods of camouflage.

8. REFERENCES

[1] Mcafee threats report: First quarter 2013. Tech. rep., McAfee Labs, 2013.

[2] Denning, D. E. An intrusion-detection model. *IEEE Transactions on Software Engineering 13*, 2 (1987), 222 – 232.

[3] Griffin, K., Schneider, S., Hu, X., and Chiueh, T.-c. Automatic generation of string signatures for malware detection. *LECTURE NOTES IN COMPUTER SCIENCE*, 5758 (2009), 101 – 120.

[4] Kuochen, W., Chun-Ying, H., Shang-Jyh, L., and Ying-Dar, L. A fuzzy pattern-based filtering algorithm for botnet detection. *Computer Networks 55* (n.d.), 3275 – 3286.

[5] Lakhotia, A., Kumar, E., and Venable, M. A method for detecting obfuscated calls in malicious binaries. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 31*, 11 (n.d.), 955 – 968.

[6] Lanzi, A., Balzarotti, D., Kruegel, C., Christodorescu, M., and Kirda, E. Accessminer. *Proceedings of the 17th ACM Conference: Computer & Communications Security* (2010), 399.

[7] Lin, D., and Stamp, M. Hunting for undetectable metamorphic viruses. *Journal in Computer Virology 7*, 3 (2011), 201.

[8] Luger, G. F. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 6th ed. Addison-Wesley Publishing Company, USA, 2008.

[9] Mutz, D., Valeur, F., Vigna, G., and Kruegel, C. Anomalous system call detection. *ACM Transactions on Information & System Security (TISSEC) 9*, 1 (2006), 61.

[10] NASCIMENTO, T. M., BOCCARDO, D. R., PRADO, C. B., MACHADO, R. C. S., and CARMO, L. F. R. C. Program matching through code analysis and artificial neural networks. *International Journal of Software Engineering & Knowledge Engineering 22*, 2 (2012), 225 – 241.

[11] Perdisci, R., Ariu, D., and Giacinto, G. Scalable fine-grained behavioral clustering of http-based malware. *Computer Networks 57*, 2 (2013), 487 – 500. Botnet Activity: Analysis, Detection and Shutdown.

[12] Skoudis, E., and Zeltser, L. *Malware: Fighting Malicious Code*. Prentice-Hall Series in Computer Networking and Distributed Systems. PRENTICE HALL COMPUTER, 2004.

[13] Szor, P. *The Art of Computer Virus Research and Defense*. Addison-Wesley Professional, 2005.

[14] T., D., R., R., G., P., K., B., M., G., and S., R. Malware target recognition via static heuristics. *Computers & Security 31* (n.d.), 137 – 147.

[15] Wei, L., Goaletsa, R., and Ali A., G. Clustering botnet communication traffic based on n-gram feature selection. *Computer Communications 34*, Special Issue of Computer Communications on Information and Future Communication Security (n.d.), 502 – 514.

[16] Wong, W., and Stamp, M. Hunting for metamorphic engines. *Journal in Computer Virology 2*, 3 (2006), 211–229.