ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

Framework for Security and Privacy in RFID based Telematics

Pradeep Kumar

M.E, MISTE, MIETE, ECE Department.

Vidya Vihar Institute of Technology, Purnea, Bihar-854301,India.

Tel:917870248311 Email: pra_deep_jec@yahoo.co.in

Abstract

Telematics by its nature requires the capture of sensor data, storage and exchange of data to obtain remote servicesMost of the commercial antivirus software fail to detect unknown and new malicious code. Proliferation of malicious code (viruses, worms, Trojans, root kits, spyware, crime ware, phishing attacks, and other malware designed to infiltrate or damage a system without user's consent) in recent years has presented a serious threat to Internet, individual users, and enterprises alike. In addition malware once confined to wired networks has now found a new breeding ground in mobile devices, automatic identification and collection (AIDC) technologies, and radio frequency identification devices (RFID) that use wireless networks to communicate and connect to the Internet. RFID systems encountered a number of threats and privacy issues. In order to stay ahead and be proactive in an asymmetric race against malicious code writers, developers of anti-malware technologies have to rely on automatic malware analysis tools. In this paper, we introduce a method of functionally classifying malware and malicious code by using well-known computational intelligent techniques. MEDiC (Malware Examiner using Disassembled Code) is our answer to a more accurate malware detection method. This work is an also attempt to address the information security issues chiefly the attacks through the databases that these RFID tags called iCLASS, which are of the active type. After a particular malicious code has been first identified, it can be analyzed to extract the signature, which provides a basis for detecting variants and mutants of the same malware in the future.

Keywords: Malicious code detection, Technical and Analytical components, MEDiC, iCLASS.

1. Introduction

Malicious code is any code added, changed, or removed from software system to intentionally cause harm or subvert system's intended function. Any computer system is vulnerable to malicious code whether or not it is attached to other systems. Examples of malicious code include viruses. Trojan horses, warms, back doors, spyware, Java attack applets, dangerous Active X and attack scripts. Combining two or more of these categories can lead a fatal attack tools. Recent Gartner report ranked viruses and warms as top security threats and hence detecting malicious code has become one of the prime research interests in the field of information security. There are two approaches that are poles apart in virus detection. One approach is too generic which includes Activity monitors and Integrity management systems. The other approach, signature based virus detection, is too specific and also popular. Almost all commercial antivirus product rely on this approach. In this a database of virus signatures is maintained and a program is detected as a virus program based on the presence of these virus signatures. These are some disadvantages in this technique.

Anti-virus software and malicious code detection tools are the most commonly used technical controls to protect information systems from malicious attacks. Nowadays, more and more computers are connected to the Internet. To prevent unauthorised access or attack, firewall is introduced to control the network traffic between the computers and the Internet. By integrating the firewall and the anti-virus software, Internet security suites are introduced in the market to provide higher protection against the cyber-attacks.

Proliferation of malicious code (viruses, worms, Trojans, rootkits, spyware, crime ware, phishing attacks, and other malware designed to infiltrate or damage a system without user's consent) in recent years has presented a serious threat to Internet, individual users, and enterprises alike. Current static scanning techniques for malicious code detection have serious limitations; on the other hand, sandbox testing fails to provide

ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

a complete satisfactory solution either due to issues such as time constraints (e.g., time bombs cannot be detected before its preset time expires).

Malware authors have found the packer techniques very useful to hide the data and bypass anti-virus software. Signature-based detection tools fail to flag these kinds of malware without having their string signatures. It is very hard to create unique signature for all malwares, since malwares can be obfuscated in several different ways and numerous variants are generated for the same virus family. Fig 1 shows the percentage of usage of prominent obfuscation tools in generating malware variants on recently observed samples. Multi-layer packing methods use multiple packers in same executable. Few packers use different encryption key to create different variants.

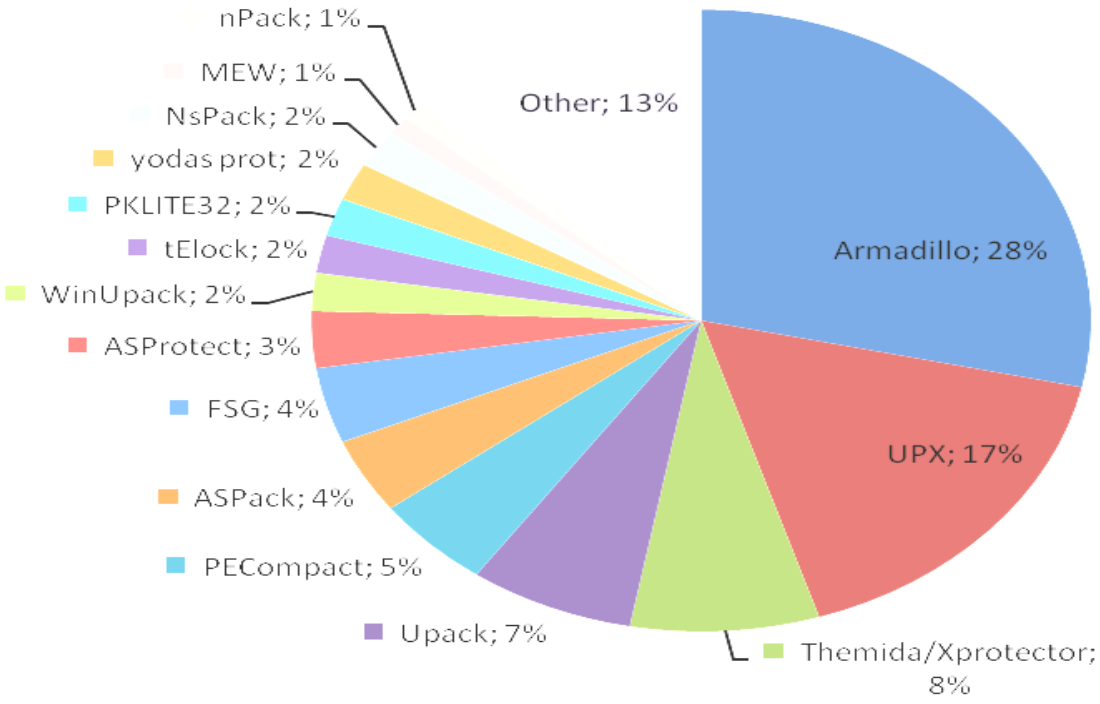


Fig.1: Distribution of obfuscation of tools used to transform malware

```
addr.sin_family = AF_INET;
addr.sin_addr = *(struct in_addr *)hent->h_addr_list[0];
addr.sin_port = htons(atoi(argv[2]));
sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
req.magic = SOCKS4_EXECBYTE;
req.polinomial = htonl(0x133C9EA2);
send(sock, (char *)&req, sizeof(req), 0);
```

The above code tries to scan the IP address and automatically connect to the other host IP address. Once the connection is established, it sends a request to accept for file transfer without user consent on the other host (by magic number). If successful it sends a copy of itself and affects the other host.

Mobile malware are on the rise and can affect both mobile phones and traditional computers. The first mobile phone malware named SYMBOS_CABIR.A spreads via Bluetooth-enabled devices, and the infected mobile started to function improperly. The risk faced by Bluetooth-enabled phone lacking antivirus software increases day by day.

ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

'Retro-viruses' like SYMBOS_DREVER.A became the most threat to mobile phone attacks, as they were capable of disabling antivirus software. These malware were capable of stealing confidential data from mobile phones.

In this paper, I introduce a method of functionally classifying malware and malicious code by using well-known computational intelligent techniques. As malicious code can affect the data and control flow of a program, static analysis may naturally be helpful as part of the detection process. We discuss the techniques in this work as a comprehensive scanning technology for today's threats and tomorrow's. Together with the dramatic increase of malware per month, the advanced nature of malware attacks has had a marked effect on the nature of scanning technology. . After a particular malicious code has been first identified, it can be analyzed to extract the signature, which provides a basis for detecting variants and mutants of the same malware in the future.

Selecting the appropriate technologies is a bit like finding the golden mean.

The technical component is a collection of program functions and algorithms that selects the data that will be analyzed by the analytical component. This data may be anything – from text strings within a file, to a specific action the program performs, to a full sequence of actions that the program performs, and more.

The analytical component serves as the decision-making system. It assesses the data provided by the technical component using one or more algorithms and then issues a verdict about the data. The security program will then use the verdict to take action on the malicious program according to the security policy that has been set in the security program.

What is making the situation worse is the ease of producing polymorphic (or variants of) and metamorphic computer viruses that are even more complex and difficult than their original versions to detect. In addition malware once confined to wired networks has now found a new breeding ground in mobile devices, automatic identification and collection (AIDC) technologies, and radio frequency identification devices (RFID) that use wireless networks to communicate and connect to the Internet.

The rest of this paper is organised as follows. In section 2, we introduce technical and analytical components. In section 3, we introduce MEDiC algorithm. Section 4 deals about RFID and iCLASS. Results and discussion are presented in section 5 and conclusion in section 6.

2. Technical and Analytical components

The technical component collects information about the file system, files, and file contents; it then passes that information on to the analytical component. The analytical component compares byte sequences in the data provided against byte sequences known to be suspicious or malicious, and issues a verdict accordingly.

Fig.2, below, identifies many of the key concepts that will be discussed in this white paper. The horizontal axis positions technical components along a continuum and shows how they overlap. The vertical axis helps to suggest the level of sophistication of analytical components – from simple comparisons to detailed analysis.

The technical component of a malicious program detection system is the data collection system that provides the data that needs to be analyzed. These are some of the most common methods used for collecting the data that will be used to identify malicious programs –1.Treat the file as a mass of bytes. 2.Emulate the program code. (Emulation means placing the program in a different environment and "tricking" it into behaving as if it's in its intended environment so that the results can be preserved.) 3. Launch the program in a sandbox. (Provide a safe environment and launch the program to determine if it "plays nicely with others.") 4. Monitor system events. 5. Scan the system for anomalies.

ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

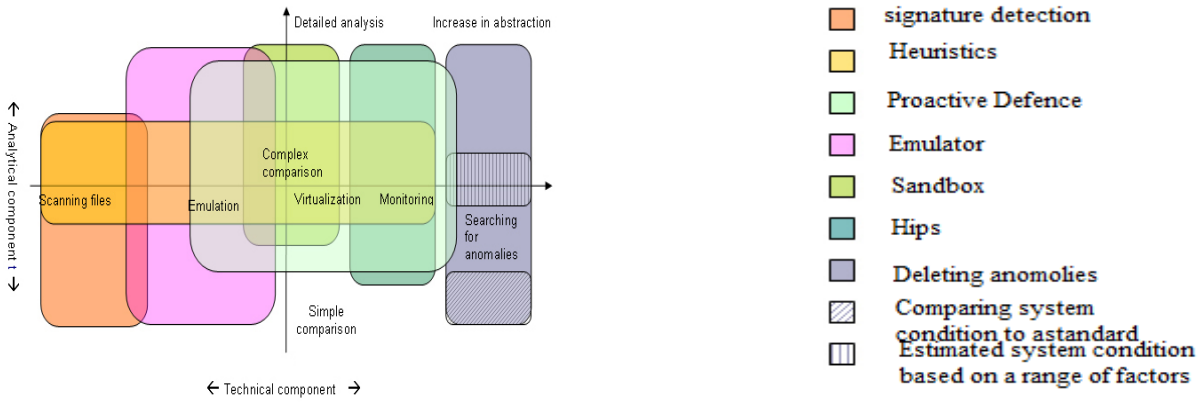


Fig.2: A Model for Assessing Methods of Detecting Malicious Code

2.1 Scanning Files

The very first antivirus programs analyzed file code as simple byte sequences.

2.2 Emulation

The emulation approach is a step between treating a program as a collection of bytes (scanning) and processing a program as a particular sequence of actions.

2.3 Virtualization – The Sandbox

Virtualization is a logical extension of emulation, and a sandbox is one form of virtualization. To continue the nanny metaphor, the plastic bubble is gone, and the sandbox is part of the real world.

2.4 Monitoring System Events

While an emulator or sandbox observes each program separately, monitoring system events is the next level of abstraction. It involves the simultaneous observation of all programs to understand their impact on the operating system. Data is collected by intercepting operating system functions. By intercepting calls to various system functions, information can be obtained about exactly which program is doing something to the system. Over time, the monitor collects statistics on these actions and transfers them to the analytical component for analysis.

Now that we've explored the technical component, we turn to the analytical component. As Fig.2 indicates, the degree of sophistication of decision-making algorithms varies as you traverse the vertical axis.

2.5 Simple Comparison

Technologies that fall into this category issue a verdict based on the comparison of a single object to an available sample. The result of the comparison is binary – a clear "yes" or "no." An example is the identification of malicious code by locating a specific byte sequence.

2.6 Complex Comparison

In a complex comparison, a verdict is rendered based on the comparison of one or multiple objects with corresponding samples. The templates for these comparisons can be flexible and the results will be probability based. An example of this is identifying malicious code by using several byte signatures, each of which is non-rigid; that is, the individual bytes are not determined.

2.7 Expert Systems

Expert systems issue a verdict only after a sophisticated analysis of data. An expert system may include elements of artificial intelligence. One example of an expert system is identifying malicious code not by a strict set of parameters, but by the results of a multifaceted assessment of all of its parameters at once, taking into account the "potentially malicious" weighting of each parameter and calculating the overall result.

2.8 Practical Facets of the Technical Component

The technical component controls three important facets of the malicious code defense system that affect its desirability for a particular user or environment –

ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

Resource consumption is the share of processor time and RAM required either continually or periodically to ensure protection. If the technical approach being used requires a lot of resources, it may slow down system performance.
Security is the level of risk that the operating system and user data will be subjected to during the process of identifying malicious code. This risk is always present when malicious code is run in an operating system.
Protection is the extent to which a technology may be vulnerable, or how easy it may be for a malicious program to avoid detection. Packing files, polymorphism, and root-kit technologies are a few approaches that virus writers use to combat file detection. It's a little tougher to circumvent emulators, but it is still possible.

2.9 Practical Facets of the Analytical Component
The analytical component of a technology also has three important facets that must be taken into consideration in evaluating a solution –
Pro-activity - refers to a technology's ability to detect new, not previously- identified malicious programs. For example, the simplest type of analysis (simple comparison) represents the least proactive technologies.
The false positive rate is also directly related to the complexity of a technology's analytical component. If malicious code is detected using a precisely defined signature or sequence of actions, as long as the signature (be it byte, behavioral, or other) is sufficiently long, identification will be absolute. The signature will only detect a specific malicious program, and not others.
The level of user involvement is the extent to which a user needs to participate in defining protection policies – creating rules, exceptions, blacklists, and white lists. It also reflects the extent to which the user participates in the process of issuing verdicts by confirming or rejecting the suspicions of the analytical system. The level of user involvement depends on the implementation.

3. MEDiC ALGORITHM

MEDiC (Malware Examiner using Disassembled Code) is our answer to a more accurate malware detection method. While MEDiC is still using static detection method, it checks an unknown executable more thoroughly. In MEDiC, a signature of a malware is determined by its disassembled code. While we do not use the whole disassembled code, we still use at least the pseudo-code part of the algorithm.

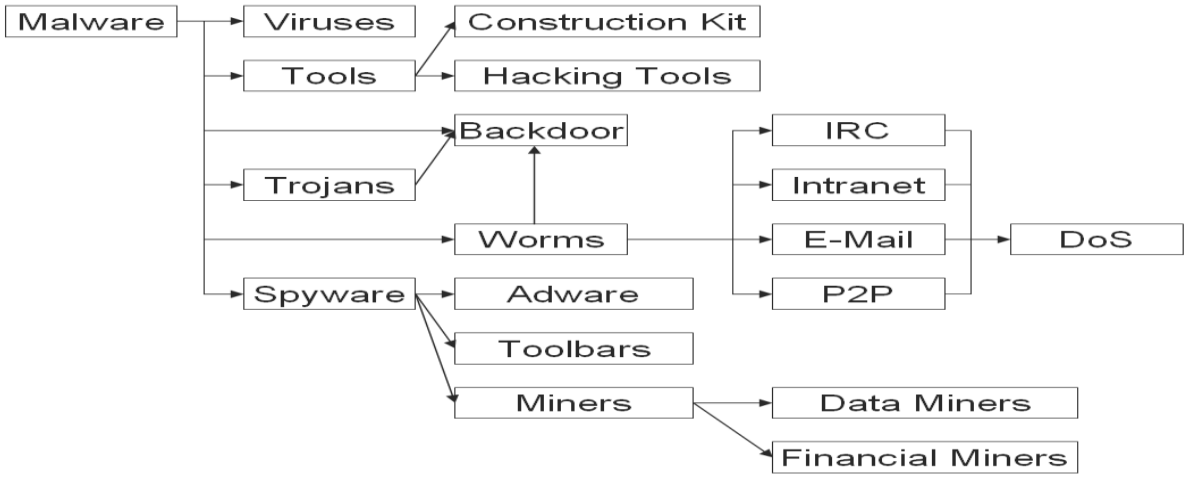


Fig.3: Malware taxonomy

When an executable is run through a de-assembler, the resulting code includes variable and constant declarations, a list of API calls it makes, and the main algorithm itself in the form of a procedure with a label and a sequence of instructions. In MEDiC, our only concern is the procedures. We treat each procedure as a label and instructions pair that we call *a code island*. A code island is a set of key/value pair with label being the key and the instructions it contains being the value. A label followed by a sequence of instructions is similar to a function or a procedure in a high level language. Each code island in the disassembled code is treated as a checkpoint.

ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

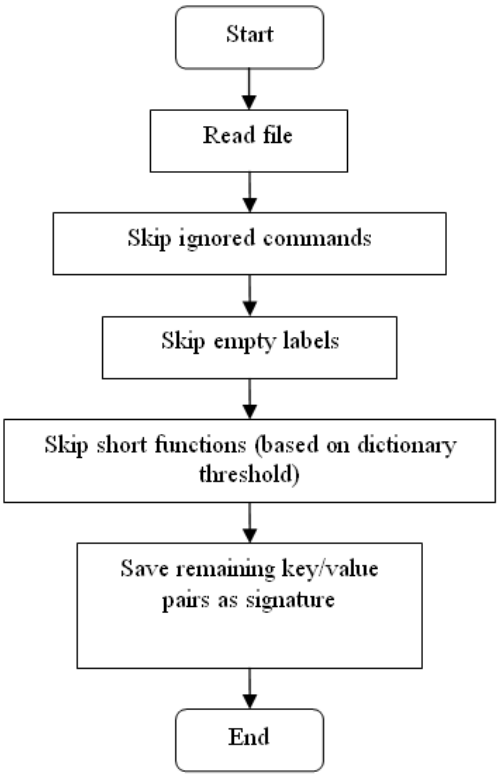


Fig.4: Flowchart to describe the process of identifying a signature

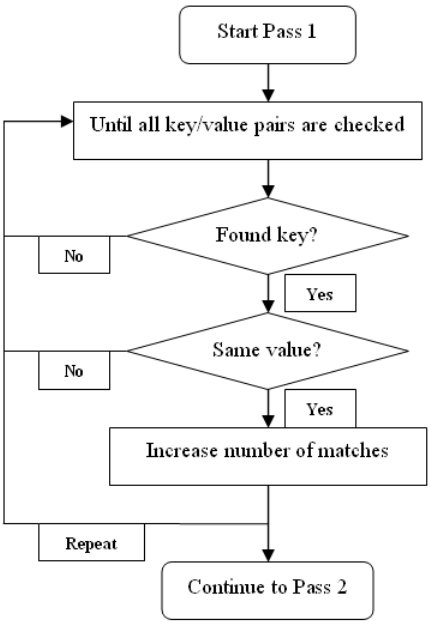


Fig.5: Flowchart to describe the first scanning process

In order to simplify the detection algorithm, we do not include all label/instructions pairs as code islands. In our analysis, we use a dictionary threshold**.** This threshold represents how many instructions there are in a code island in order to be considered important. For example, if we set the threshold to 10, there has to be ten instructions associated with a code island in order for the label to be considered as a code island and be included in the signature set.

For detection algorithm, we use the same method of creating a list of checkpoints from a potential malicious code. First, we disassemble the code use the PE Explorer to get an ASM file. Then, we create a list of checkpoints from the disassembled code. The list of checkpoints contains key/value pairs for the code islands in the code, similar to the ones we get for the signature set fig.4 and 5.

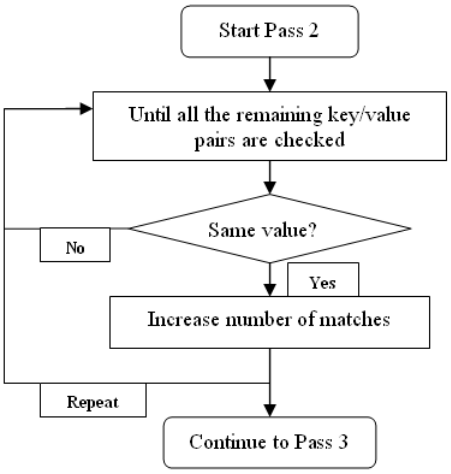
Since signature matching returns a number of matches and a number of checks, our program sets a virus threshold. This is the lowest ratio of matches over the number of checks performed that will conclude the code as malicious.

Based on the checkpoints, our program performs a series of scanning. The first scanning is to match the key/value pairs with the ones in the signature set. If the number of matches exceeds the set virus threshold, then we stop scanning and conclude that this code is malicious. If not we continue to pass 2 in order to perform the second scanning step fig.4 and 5.

From the misses in the first scanning, we now perform the second scanning. The second scanning excludes the key in the key/value matching and determines how many value matches in case the key names are different. Key names may be different because of padding or moving functions around. Again, we conclude the scan if the number of matches exceeds the set virus threshold fig.6 and 7.

ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

The third scanning uses a more thorough process. This part is used when even the value does not match. This third scanning is used against a more difficult obfuscation than padding or moving functions. We scan and determine how many instructions in this set match our signature fig.6 and 7.



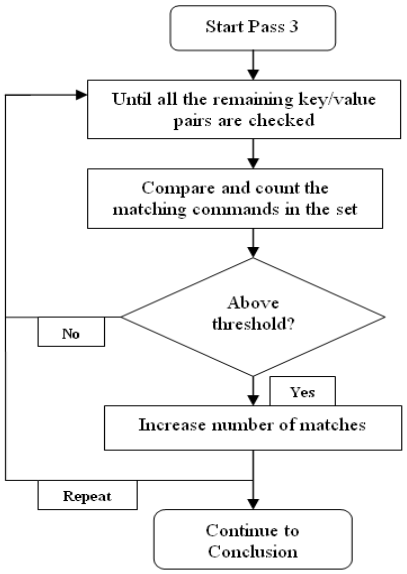


Fig.6: Flowchart to describe the second scanning steps Fig.7:Flowchart to describe the third scanning steps

Based on the search threshold value, we loosen the number of instructions we have to match with the signature as we scan more lines. For example, the search threshold value is set to 5:

- If there are 5 or less instructions that we have to match in the signature, then we have to match all 5 instructions in order to include this code island as a match.
- If there are 6-10 instructions, we have to match 5-9 instructions respectively.
- If there are 11-15 instructions, we have to match 9-13 instructions respectively.

4. RFID AND iCLASS

4.1 RFID

Daily usage of RFID is widely increasing. Its ease of tracking and identifying an object or a person from varied distances has made RFID technology more significant and rampant. An estimated of 40 million people in the US are supposedly carrying some kind of RFID device with them. This relatively shows the ubiquity of the RFID devices.

RFID technology has been around since World War II. It was used to identify aircraft as “friend” or “foe” (IFF) by interrogating approaching aircraft with a radio signal and receiving a response. It is now widely used for personnel access control, toll roads and animal tracking.

RFID systems primarily consists of the RFID tags or chips, the RFID readers, the antennas, the computer networks and finally the software that takes care of the information carried by these RFID tags. An RFID tag (shown in fig.8) is a tiny, flat microchip with a built-in antenna, which is available in various sizes, but with the same basic functionality.

ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

When a radio signal is incident on an RFID tag, the RFID is activated and broadcasts the information it contains. The RFID tags can be attached to or incorporated into a product, animal, or person for the purpose of identification using radio waves.

These RFID tags are of two types: passive and active. The passive ones do not have any power of their own. They respond only if encountered by radio waves from the readers. On the other hand, active RFID tags have a built-in power source and more recently, certain computing or sensing technology (e.g. sub-dermal chips) which emit the data in the form of radio frequency waves to be received by a legitimate reader. These active RFID tags usually have higher storage capacities when compared to their passive counterparts.

We are concerned with the back-end entity of the RFID infrastructure which is the software aspect essentially the databases. The databases have their own set of security issues. With the rise of internet usage and computing power, a lot of information from these databases is exchanged over the internet. Hence, these databases are at the verge of facing the various security problems that engulfed the internet.

RFID systems encountered a number of threats and privacy issues. This work is an attempt to address the information security issues chiefly the attacks through the databases that these RFID tags communicate with. This paper provides a proof-of-concept to run an executable on a windows machine using SQL Server 2005 as the back-end architecture.

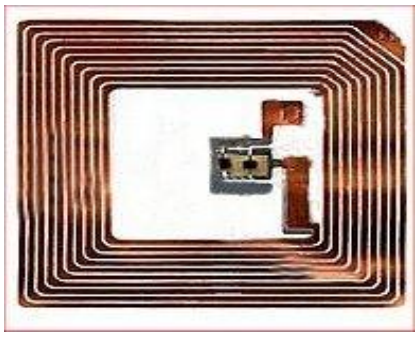


Fig.8: An RFID tag

4.2 iCLASS

Our experiments use contactless RFID tags called iCLASS, which are of the active type. They can contain 2048 bytes of data. However, the maximum effective storage capacity is 1896 bytes per tag because of all the reserved areas that contain the serial number, password, etc. Table 1 shows the storage capacity of an iCLASS tag when using different data types.

Table 1: Storage Capacity of an iCLASS tag

Data Type	Size (bits)	Size (bytes)	How many fit in 2048-byte iCLASS tag?
Boolean	1	0.125	15168
Character	8	1	1896
Integer < 65535	16	2	948

ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

Memory areas to read and write data are classified as iCLASS Page 0 Memory Map and Application Memory Map Pages 1 through 7. iCLASS Page Memory Map is as shown in fig.9.

Blocks 0 through 5 in each page contain configuration information. For our experiments, we use all blocks in pages 1 through 7 to store data. Configuration information in page 0 is left unused, which decreases the number of bytes available to store data down to 1896 bytes. This is possible because we treat the tag as one continuous application area.

We can also divide up the tag into sixteen application areas, two per page, but at the risk of decreasing the storage even more. When using sixteen application areas, blocks 0 through 5 in every page are not available for writing as they are used to configure that particular page.

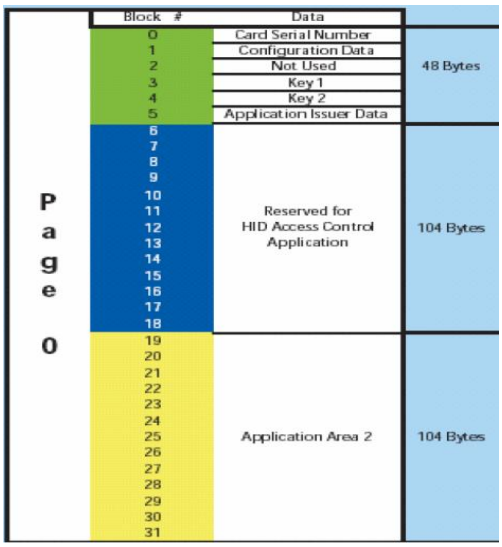


Fig.9: Memory map for iCLASS page 0

5. RESULTS AND DISCUSSION

5.1 MEDiC Experiment on Real Variants

The next experiment focuses on the variants available in the wild instead of our very own obfuscated versions of the malware. From the malware samples in our collection, we used commercial anti-virus scanners to identify each variant. Let us take Wozer:C as an example. We tested three variants of Wozer: C, E, and F. The anti-virus scanners treat the three samples as different variants. Therefore, the scanners require different signatures to recognize them.

We provided our tool with only one sample for each known malware. We set our tool to recognize a piece of code as malware when virus threshold rises above 65%. According to our experiments, this percentage is the value that minimizes the number of false positives down to zero while still maintaining its ability to detect variants of malware in its signature.

Table 2 demonstrates that our tool can detect all three samples of Wozer with only Wozer.C signature in its arsenal. There variants could still be detected because the virus threshold value is still reached when compared to its original, Wozer.C. The same trend occurs with other malware like Mydoom, Perenast, Beagle, Klez, and ZMist. While anti-virus software classifies them differently, NMTMEDIC recognizes the samples as variants.

ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

5.2 MEDiC Experiments on Spyware and Adware

Regarding spyware and adware, we include several samples of popular spyware in our experiments to see how our tool performs in this category. The main reason we include spyware and adware into our experiments is that while spyware and adware may not disable the system, their method of installation can be devious. Once installed in a system, they can and may be used by others to perform even more malicious behavior.

Table 2: Variant detection using one signature set for each strand of malware

AV Classifications	Signatures used					
	Perenast.C1	Wozer.C	Zmist.C1	Beagle.A	MyDoom.A	Klez.E
erenast.C1	✓	×	×	×	×	×
erenast.C3	✓	×	×	×	×	×
Vozer.C	×	✓	×	×	×	×
Vozer.E	×	✓	×	×	×	×
Vozer.F	×	✓	×	×	×	×
Mist.C1	×	×	✓	×	×	×
Mist.C2	×	×	✓	×	×	×
Mist.C3	×	×	✓	×	×	×
eagle.A	×	×	×	✓	×	×
eagle.I	×	×	×	✓	×	×
eagle.J	×	×	×	✓	×	×
lyDoom.A	×	×	×	×	✓	×
lyDoom.E	×	×	×	×	✓	×
lyDoom.F	×	×	×	×	✓	×
lez.E	×	×	×	×	×	✓
lez.H	×	×	×	×	×	✓

Table 3: Analysis of similarity matches for spyware

	Signatures used									
	180SA.Sauhook	Bargains	Begin2Search	Claria.A.DbAu	Ebates	Gator.CMESys	ShopAtHome	Sidefind	WebRebates	WhenU.Sync
ains	6.5	100	5.41	4.3	7.9	5.5	79.2	8.3	63.4	5.9
ia.A.DbAu	63.9	16.8	46.2	100	51.3	80.3	18.9	33.3	16.7	41.7
r.CMESys	60	12.5	34.2	36.8	40.9	100	13.6	27.7	13.1	34.6
r.GMT	70.8	30.6	67.9	99.2	63.5	84.9	26.6	72.2	29.3	55.2
	66.9	16.8	48.4	36.4	42.6	67.3	21.8	22.2	22.7	69.4
AtHome	6.0	82.5	5.4	4.8	7.7	5.0	100	8.3	86.2	7.0
Rebates	6.5	66.9	5.7	4.9	8.1	4.8	82.2	5.5	100	6.9

From the results in Table 3, we can see that NMTMEDIC can be applied to spyware as well. The numbers represent percentages of similarity of a particular spyware when compared to the signatures provided to our tool.

After analyzing the result for matches around and above 65%, there are several interesting points we can make from this experiment:

Gator e-Wallet (Gator.CMESys and Gator.GMT) and Claria DashBar (Claria.A.DbAu) have high similarity to each other because they come from the same company. Gator Corporation changed its name to Claria in October 2003. The results are highlighted in red. Hence the claim here is that in the future any similar product or versions made by this company is bound to be detected by NMTMEDIC.

Bargains, Shop At Home, and Web Rebates have high similarity matches because they are categorized as data mining programs. All these programs watch the user as and when they are surfing the internet and report the findings to the author. The results are highlighted in green.

Save and When U.Sync are also from the same company. The result is highlighted in blue.

Since Gator and Claria function libraries are so large, they are recognized using the signatures of other malicious programs. The results are highlighted in gray.

180SA also recognizes a lot of Gator/Claria functionalities. The results are highlighted in yellow.

RFID technology has its own disadvantages in security and privacy aspects. The security of an RFID-enabled system depends mostly on how secure the middleware is developed. It also relies on the data contained in RFID tags, which can surprisingly lead to a SQL injection attack, denial of service attack or even a buffer overflow.

Tag readers can communicate in two ways. There are several security issues based on these communication methods: when tag readers convey data via internet protocol and when tag readers provide and gather data to and from the tags via low power radio frequency.

6. CONCLUSION

ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)

In the real world, application of this technique has a great advantage over current detection techniques deployed by commercial anti-virus software. Suppose a new malware XA surfaces to the public. After a careful analysis, we and anti-virus software provide upgrades. Several weeks later, the malware author lets loose XB, which is a variant of XA. It gets even worse when the malware author releases his or her source code. Many more people can change and let loose their own variants that escape anti-virus detection.

ACKNOWLEDGMENT

The authors would like to thank Dr,Abhay Kumar and Miss.K.jayanthi for their insightful advice and guidance, and unknown reviewers for their useful remarks and suggestions.

REFERENCES

AT&T, "Privacy Minder," http://www.research.att.com/projects/p3p/pm

Chunhua Sun, Wei Zhang, and Khaled Ben Letaief, Cluster-based cooperative spectrum sensing in cognitive radio systems," in Proc. IEEE International Conference in Communications, pp. 2511-2515, 2007.

David L. Donoho, "Compressive sensing," IEEE Trans. on Information Theory, pp. 1289-1306, April 2006.

Ghasemi, A and Sousa, E "Collaborative spectrum sensing for opportunistic access in fading environments," in Prco. IEEE DySPAN, Baltimore, MD, USA, pp. 131-136, Nov. 2005.

Meng, J, Yin, W, H. Li, E. Houssain and Z. Han, "Collaborative spectrum sensing from sparse observations using matrix completion for cognitive radio networks", The 35th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2010.

Pati, Y, C et. al ."Orthogonal matching pursuit recursive function approximation with application to wavelet decomposition," 27th Annual Asimolar Conference on Signal System and Computers,1993.

Tian, Z "Compressed wideband sensing in cooperative cognitive radio networks," in Proc. IEEE Globecom Conf., pp. 1-5, New Orleans, Dec. 2008.

"The Platform for Privacy Preferences 1.0 (P3P1.0) Specification". April, 2002 http://www.w3.org/TR/P3P