Independent Study Proposal

Independent Study: **Cooperative virus defense system**

Student: **Nuno Brito** (nbrito@andrew.cmu.edu)

1. Statement of purpose and goals of project

The research project that I propose intends to explore the concept of a cooperative defense mechanism for the detection and removal of viruses.

To properly explain the purpose and intended goals of this project, I will trace a direct analogy from the virtual world of computer systems onto the real world at a particular point in history such as the roman civilization in Europe.

Back in the roman days, a farmer would peacefully work on his cultures, improving his farm. However, if he had the misfortune to be found by aggressive barbarians [1] then his work would be pillaged and perhaps lose his own life.

Therefore, farmers would often be found working near fortified villages [2] where they could better defend themselves as a group against a stronger enemy in common such as the barbarians in this case.

Romans advanced the concept of civilization when connecting these fortified villages with a network of roads [3] where armies could move quickly from one location to another and goods be traded amongst farmers.

This concept of cooperation allowed Romans to grow as an empire managed efficiently from a defense and commercial perspective.

Transporting this era to a virtual world – computer users are farmers from the roman era and roman roads are now a large group of networks called Internet.

In this scenario, computer users seek the same peaceful goal of doing their work but barbarians *(in this case viruses and attackers)* still pillage their properties and cause harm.

Organizations, much alike the fortified roman villages, set as a priority the defense from barbarians to keep them outside their walls.

In this virtual world, farmers attempt to protect themselves with a wide array of available weapons such as firewalls, antivirus, restricted privileges and so forth.

Although in some cases well equipped with these weapons on his own machine, a single farmer remains in clear disadvantage when fighting against a tribe of aggressive barbarians that are professionals in the art of war.

Defense weapons for companies and organizations can be managed in large scale [4, 5] by network administrators but there seem to exist scarce tools currently available on the market that allow the cooperation between workstations of a company to preemptively [6] resist and react against an outside attack.

As result, when a virus discovers a point of entry inside the defenses of a given workstation, there is often no timely response or action that avoids the attack of all the other farmers in the region as seen on the case of the Conficker botnet [7, 8].

Following the example from the roman days, I aim to research and propose a system that provides conditions for farmers to group together near the fortified villages and stand with better odds of fighting against the barbarians.

In specific terms, the goal for this research is to provide a proof of concept where workstations are capable of interacting semi-autonomously between themselves, identifying suspicious risk patterns inside their own domain and allow timely reactions both from the network administrator and the computer user, including preemptive actions from the workstations to defend themselves from recognized attacks.

2. Description of the subject matter

The concept consists in providing computer workstations the power to interact and group with other workstations in contrast to a passive attitude of no further defense after an attacker circumvents the virus/attack detection tools based on heuristics and signatures.

Defense mechanisms can assume diverse forms of action. The most common is the standard antivirus that will monitor actions or changes on files, registry settings *(on the case of MS Windows)* and processes that are running.

A modern antivirus engine will detect malicious processes based on heuristic [9] and signature [10] methods. These approaches suffer from two well-known disadvantages: heuristic detection can lead to a high number of false positives and signature detection is not updated on time to cope with the most recent malware attacks.

The proof of concept on this research is separated in two parts: **detection and repair** of malicious activity.

Detection starts by indexing and processing the details from a critical set of files inside the workstation and storing this information on a database at the workstation machine.

This database is periodically synchronized with a central repository where the information from the workstation is uploaded and recent information from the remote database is received.

The remote database merges the information of the files from all client workstations. Each client has it's own identification and this id is included with each file data submission.

After synchronizing with the remote server repository, the workstation will re-evaluate it's own index of files to validate their level of trustworthiness.

Since each workstation is submitting their own index of files with details that are merged with other clients, this validation is made with resort to a probability algorithm.

How would a validation algorithm work?

For example, let's imagine a universe of 100 workstations using the same file called msword.exe with an internal file version of 1.4.35.

If 96 workstations testify that over the past two weeks this file has a given MD6 checksum [11] and 4 other workstations disagree on this MD6 result, then we would have a positive indication that this file had been tampered from the original version on these 4 machines.

This positive indication of tampering on this particular file would trigger alarm messages be sent to the network administrator and computer users. This would be helpful to detect *(for example)*, attempts of modifying system files by malware.

The second part of our proof of concept is a preemptive repair. Given the fact that the file from our example is currently tampered on 4 machines, each workstation should request a copy from a workstation considered as trustworthy by the group of 100 machines and replace their tampered files with a file considered as safe.

Farmers are now working together to defeat a common enemy.

This type of defense is valid to share information amongst a large pool of workstation systems and disable malware that modifies critical system files or attempts to introduce itself inside the system *(as a service, on the auto-start, etc)*.

The power from this method derives from the scale of cooperating workstations that will be reflected on the accuracy of the algorithms. This defense system becomes more efficient when the group of workstations share a considerable number of identical files as available on the LAN infrastructure of organizations that are continuously under attack.

In order to keep this explanation brief, I did not explored details about handling of files that are constantly modified, untraceable malware actions or risk of false positives amongst other possible attack vectors but these considerations will be present in the deliverables.

I will focus in documenting a defense mechanism that complements the standard antivirus technology based on detecting changes on files, registry configuration and running processes.

From a software engineering perspective, the project will follow a proper development process to uphold the identified architectural drivers.

3. Explanation of relevance to interests or major area of study

Given my professional past in military level security, I see virus defense as one of the most difficult and challenging characteristics to maintain inside an organization.

This project is very relevant to my personal interests since it will improve my professional skills in security analysis and software engineering at a real world experiment research.

It will allow me to put into practice most of the concepts that were taught in the core courses to propose the architecture for a proof of concept and be able to discuss the followed approaches and decisions with mentors and professors.

I've chosen a theme that is fairly challenging in terms of security since there is no similar technique in practice at the current date. This is a promising improvement to help fight crime and reduce damages on organizations and individuals.

4. Description of proposed research methodology

I intend to apply the knowledge acquired during the core classes of the MSE program to research and propose a system following the engineering principles expected from a professional approach.

In closer detail I will list the methods used for this research:

Software Engineering

17-651 Models of Software Systems

UML 2.2 will be used to model the proposed system. I will focus on behavior diagrams such as activity diagrams to represent the workflow of activities and sequence diagrams to describe the order of events for relevant actions. If concurrency is deemed as necessary, Petri nets can be used as modeling method.

17-652 Methods: Deciding what to design

Apply techniques as mentioned in class and assignments to better understand the problems expected to be solved by the system.

17-653 Managing Software Development

Perform requirements elicitation, address issues exposed from risk analysis, estimate costs, reason about team organization and identify available resources.

17-654 Analysis of Software Artifacts

The analysis of damage and propagation potential of an hypothetical botnet virus will be calculated with resort to queuing theory, if necessary I will also use colored Petri nets to analyze behavioral and performance issues/limitations of the system.

17-655 Architectures for Software Systems

Describe and evaluate at an architectural level of abstraction the software system for the proof of concept. Major architectural styles and patterns will be identified and the research will reason about alternatives to address exposed problems.

Social Engineering

This research project will also focus on analyzing how social engineering is employed to mislead users and overcome standard detection defenses. This section will be supported on knowledge from literature and observation of real-world cases.

Expert interviews

Technical experts in this area will be consulted to inquire about feasibility of conclusions and assumptions made during this research.

Contact was already established with experts such as Philip Porras, program director from SRI International that successfully decomposed the Conficker botnet [12] and published an extensive and detailed report to the public.

The Shadowserver foundation [13] is also available to provide detailed information about newly exposed techniques employed by malware authors.

5. Preliminary working bibliography –

Architecting Software Intensive Systems: *A Practitioner's Guide*, by Anthony J. Lattanze, Taylor and Francis/Auerbach 2008

Software Architecture in Practice, *Second Edition*, by Len Bass, Paul Clements, and Rick Kazman, Addison-Wesley 2003

Terry V. Benzel, Cynthia E. Irvine, Timothy E. Levin, Ganesha Bhaskara,Thuy D. Nguyen, and Paul C. Clark, *Design Principles for Security*, Naval Post Graduate School Report (NPS-CS-05-010), USC Information Sciences Institute Report (ISI-TR-605)

Documenting Software Architectures: Views and Beyond, by Clements, et al. Addison-Wesley 2003

David Garlan, Bradley Schmerl, and Shang-Wen Cheng, *"Software Architecture-Based Self-Adaptation"*. In Autonomic Computing and Networking, Springer Verlag, 2009

Anthony J. Lattanze, Architecture Centric Design Method: *A Practical Architectural Design Method for Software Intensive Systems*, ICSOFT, January 2007

E. Cooke, F. Jahanian, and D. McPherson, The zombie roundup: Understanding, detecting and disrupting botnets. *In Proceedings of Usenix Workshop on Stepts to Reducing Unwanted Traffic on the Internet* (SRUTI '05), Cambridge, MA, July 2005.

6. Deliverables and final report –

The research project will consist of the following deliverables:

- A paper between 10 to 20 pages in length documenting the architectural design of the prototype;
- A final paper between 10 to 20 pages in length documenting the research process, lessons learned, results and reflection;
- A prototype showcasing the proof of concept product.

7. Timeline –

The semester will be divided into weekly iterations based on the 8 ACDM stages [14].

On figure 1, the timeline is decomposed into 4 groups of ACDM stages across the 12 weeks of the summer semester, delivering the results from the study at least one week before the deadline for final grades.

This is an initial timeline estimation that will be adjusted as necessary during development.

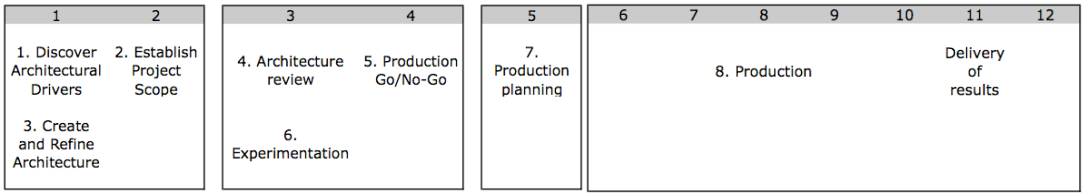

Fig 1 – Estimated timeline

8. Reviews –

Weekly status meetings will be held with the advisor to gather feedback and report progress. To provide advice on this research I have asked support to Professor Benoît Morel, which lectured my elective course: 19-601 Information Warfare and CyberSecurity.

9. References

[1] http://en.wikipedia.org/wiki/Barbarians

[2] http://en.wikipedia.org/wiki/Castrum

[3] http://en.wikipedia.org/wiki/Roman_roads

[12] http://mtc.sri.com/Conficker

[7] https://infosecurity.us/?p=6313

[11] http://en.wikipedia.org/wiki/MD6

[13] http://www.shadowserver.org/wiki/

[8] http://mtc.sri.com/Conficker/addendumC/

[6] http://www.thefreedictionary.com/preemptively

[9] http://en.wikipedia.org/wiki/Antivirus_software#Heuristics

[4]http://www.microsoft.com/windows/windowsintune/default.aspx

[14]http://en.wikipedia.org/wiki/Architecture_Centric_Design_Method

[5]http://www.avira.com/en/products/avira_security_management_center.html

[10]http://en.wikipedia.org/wiki/Antivirus_software#Signature_based_detection