Descriptions of Suggested Projects

IT-S448, ITMS448 & ITMS548 Autumn 2014
Version 4

This document contains descriptions of a number of projects to be pursued as part of this course. This list of projects is divided into two focus areas; (1) cyber steganography & steganalysis and (2) cyber security. Students will form teams of 2 or 3 students to work on one of these projects; however a student can work on a project alone with approval of the instructor.

Some of the projects cannot be reasonably done in one semester. To continue the project, project members will be able to register for a follow-on course in the spring 2015. For non-stego security-related projects the follow-on courses are IT-S549 or ITMS549. For steganography and steganalysis related projects, the follow-on courses are IT-S539 or ITMS539. These follow-on courses will be described on Blackboard.

Please review these projects and decide on which projects your team would be interested in doing. Choose 3 projects and rank order them in terms of your preference in doing them. You will need to consult with possible team members in doing this.

Cyber Steganography & Steganalysis Project Descriptions for IT-S448 / ITMS448 / ITMS548

Cyber steganography is the science and art of hiding digital information in seemingly innocent files so that users and observers of these files are unaware that the hidden information exists. For instance, instructions for creating a bomb could be hidden in an image file of a picture of a bowl of fruit or your aunt Martha. Cyber steganalysis is the science and art of detecting the existence of hidden information and possibly extracting the hidden information.

Cyber stego (both the hiding and the detection) is a new but very active area of cyber warfare and countermeasures. There's much fertile area for investigation and projects.

Below is a list of possible stego projects from which to choose. For ease in describing the projects, here are some definitions:

- ***Covert***: The hidden file or information
- ***Carrier***: The seemingly innocent file in which the covert file or information is hidden. It "carries" the covert file or information.

Stego project descriptions begin on next page. They are designated with the leading characters ***S****number.* Example: ***S2*** is the second stego-related project.

S1. Video File Steganography & Steganalysis

Introduction & Background

There is evidence that drug cartels and terrorist organizations are using image, document, audio and video files to communicate. Governments around the world are concerned about this. There are many cyber stego software tools available for hiding; this is especially true for image files such as JPEG and BMP files. Many of them are free.

Issues and/or Problems Being Addressed

Video files are large and widely circulated on the Internet. They represent fertile ground for carrying large amounts of hidden information. But there are very few tools for hiding info in video files. This project will determine how to hide information in popular video files. The ***.mp4*** carrier file types will be studied.

Possible Approach

1. An absolutely necessary first step is to understand and be able to describe the details of the formats of ***.mp4*** files. This is necessary in order to identify how and where covert information can be hidden.
2. A second step is to investigate and obtain existing software tools for hiding in these carrier files. For these software tools, determine how the hiding is done. This can be done by searching for documentation on the tools and also by comparing "clean" and "dirty" carrier files using hex editing software such as WinHex.
3. Also consider other possible ways of hiding covert info in these file types.
4. Consider ways in which to detect whether a ***.mp4*** file has covert information hidden in it. This determination can be probabilistic.
5. Finally develop a software tool to do this detection.

MATLAB may be useful in this project.

Project Goals

Understand and be able to describe the format for ***.mp4*** files and also the ways that covert information can be hidden in***.mp4*** files. In addition, describe other possible ways that steganography can be accomplished.

Project Deliverables

A well written technical paper, a well written user guide and a well developed technical presentation are requirements. For two-semester projects, these will be required at the end of each semester, and at the end of the 2nd semester a robust working software tool will be required and demonstrated. Finally, a formal presentation and demonstration at ForenSecure15 in April 2015 will be required for all good projects.

Estimated Level of Effort and Challenges

Developing the software tool is definitely a two semester project. But without writing the tool, this might be a one semester project if the students have adequate background. Students would take IT-S / ITMS539 in spring 2014 to complete the project. To develop a tool programming skills will definitely be required.
Also, a second project might be considered, targeting ***.avi*** instead of .mp4.

Starter References

Two tools that are possible candidates for video stego are ***MSU Stego*** and ***TCStego***. But other tools may have similar capabilities.

M. Raggo, C. Hosmer, *Data Hiding*, Syngress/Elsevier, ISBN 978-1-59749-743-5.

S2. JPEG Image Calibration Analysis

Introduction & Background

One problem in JPEG image steganalysis is how to determine whether or not an image contains hidden information. While this is not too difficult if a known "clean" version of the image is available, this is usually not the case. When a clean image is not available to compare with the stego image, then the task is especially difficult. This is called **blind** JPEG steganalysis.

A suggested solution is to create a statistical approximation of a clean image from the image in question. But how? Jessica Fridrich's Calibration Technique proposes to do this. Her approach involves decompressing the JPEG image, cropping by 4 pixels, and then recompressing using the same quantization tables as the original JPEG image. However, is this the only solution or just the best? An alternative possibility is to convert the JPEG to a BMP and then convert the BMP back to a JPEG, using the same quantization tables as the original JPEG image.

Which scheme is better? What does "better" mean?

Issues and/or Problems Being Addressed

The fundamental problem is to determine the probability of JPEG image files containing hidden information by the use of two image calibration schemes: (1) Fridrich's Calibration Technique and (2) JPG > BMP > JPG conversion; and to determine which of these two techniques performs better and under what circumstances.

Possible Approach

1. Gain a good understanding of JPEG image compression and JPEG file formats.
2. Gain a good understanding of the ways in which information can be hidden in JPEG files.
3. Gain a basic understanding of Fridrich's Calibration Technique. The Starter References will provide the relevant information. MATLAB and C++ code are available to perform this calculation.
4. Understand what it means for a JPEG image to be statistically clean. Possibly use Fridrich's feature sets.
5. Acquire a library of clean BMP images.
6. Create a library of JPG images using ***ImageMagick*** and a specified quantization table.
7. Use ***ImageMagick*** to perform the conversion from JPEG to BMP and from BMP to JPEG.
8. Create a library of images with hidden information. Use existing stego tools to hide the information.
9. Analyze a library of images using both techniques to determine which gives the closer approximation to a statistically clean image by comparing them to your original clean images.

Project Goals

1. Understand Fridrich's Calibration Technique and feature sets.
2. Understand the JPEG -> BMP -> JPEG technique.
3. Compare the two techniques.

Project Deliverables

A well written technical paper, a well written user guide and a well developed technical presentation are requirements. For two-semester projects, these will be required at the end of each semester, and at the end of the 2nd semester a robust working software tool will be required and demonstrated. Finally, a formal presentation and demonstration at ForenSecure15 in April 2015 will be required for all good projects.

Estimated Level of Effort and Challenges

Two semesters. Take 539 to finish project. Code would either be developed or found that performs each calibration technique. A knowledge of statistics would be useful for the analysis of the results.

Starter References

http://www.imagemagick.org/
http://www.ws.binghamton.edu/fridrich/research/calibration-color.PDF
http://cs.gmu.edu/~zduric/cs803/Blind_JPEGb_Fridrich.pdf

Prior work in this course at IIT on similar projects during previous years is also available.

S3. Bit-Plane Complexity Segmentation (BPCS) Steganography

Introduction & Background

Bit-Plane Complexity Segmentation (BPCS) was invented by Eiji Kawaguchi and Richard O. Eason in 1997. BPCS steganography primarily uses true color (i.e. 24-bit) images as the carrier. The embedding occurs on the bit-planes of the image. Compared to other embedding techniques, BPCS is able to embed with a higher capacity. BPCS considers image complexity and uses the "complex areas" of an image for embedding.

Issues and/or Problems Being Addressed

Compare BPCS approach with least significant bit (LSB) substitution in terms of hiding capacity, implementation complexity and steganalysis difficulty.

Possible Approaches

1. Learn about JPEG and BMP files and compression.
2. Understand LSB Substitution for both BMP and JPEG images.
3. Gain an understanding of BPCS by reading the original authors paper.
4. Understand Pure-Binary Coding system and Canonical Gray Coding system.
5. Acquire or create a set of images and determine their complexity using the BPCS formula. Try to find images with varying levels of complexity. The images should be converted to Canonical Gray Coding.
6. Select a suitable text document for embedding.
7. Embed a standardized document in each image using BPCS.
8. Embed the standardized document in each image using LSB Substitution.

Project Goals

1. Understand BPCS and the theory behind it, including the image complexity calculation.
2. Understand LSB Substitution.
3. Compare BPCS and LSB Substitution
    - Hiding capacity
    - Implementation complexity
    - Steganalysis

Project Deliverables

A well written technical paper, a well written user guide and a well developed technical presentation are requirements. For two-semester projects, these will be required at the end of each semester, and at the end of the 2nd semester a robust working software tool will be required and demonstrated. Finally, a formal presentation and demonstration at ForenSecure15 in April 2015 will be required for all good projects.

Estimated Level of Effort and Challenges

Two semesters. Take 539 to finish project. Code will either need to be developed or a suitable tool found to do the embedding (both BPCS and LSB Substitution) and BPCS image complexity calculation.

Starter References

http://datahide.com/BPCSe/principle-e.html
http://datahide.com/BPCSe/Articles/Ref-6.SPIE98.pdf
http://www.ijest.info/docs/IJEST10-02-09-173.pdf

S4. Evaluation of StegoHunt

Introduction & Background

StegoHunt is a steganalysis tool that claims to detect the existence of hidden or covert information in seemingly innocent overt files. StegoHunt does blind detection; i.e., steganalysis without the benefit of "clean" overt files.

Issues and/or Problems Being Addressed

What can StegoHunt really detect and what can't it detect and why?

Possible Approach

1. Understand in what ways covert information can be hidden in different carrier (i.e., overt) files. This will include understanding of the compression schemes that are used for JPG, MP3 and other compression.
2. Identify and then acquire a library of "clean" files that you know are not carrying hidden information. The files should be of various types such as MP4, MP3, AVI, BMP, JPG, TXT, MSWord...
3. Identify and then acquire a collection of steganography tools that you will use to insert information into your library of clean files. Make sure that you include F3 and F5 in this
4. Choose files of information that you will hide. Files should be of 3 or 4 different types and sizes.
5. Create a library of files "dirty" files (i.e., containing hidden information). You will do this by using the stego tools to hide information into copies of the clean files. Information should be of 3 or 4 different types and sizes.
6. Use StegoHunt to detect or not detect the existence of hidden information in a mix of dirty and clean files.
7. Evaluate StegoHunt's ability to detect hidden information. Analyze success or failure in terms of such factors as type of carrier file, type of hidden file, size of hidden file relative to carrier file size, and stego tool used.
8. Draw conclusions and generalizations about your results.

Project Goals

A deep understanding by students of how compression and steganography are done in for various carrier file types. A well thought out evaluation of StegoHunt. Also suspected schemes that StegoHunt uses for steganalysis, considering anomaly analysis as a possibility.

Project Deliverables

A well written technical paper, a well written guide so that others can duplicate your results, and a well developed technical presentation are requirements. Finally, a formal presentation and demonstration at ForenSecure15 in April 2015 will be strongly encouraged.

Estimated Level of Effort and Challenges

One semester. No coding is needed, but a deep understanding of both compression, steganography and suspected StegoHunt approaches to steganalysis

Starter References

Much work has been done in the past two years by previous students in MP3 and JPG steganography and steganalysis. This work will be available to students doing this project.

Cyber Security Project Descriptions for IT-S448 / ITMS448 / ITMS548

Non-stego project descriptions begin on next page. They are designated with the leading characters ***!S****number.*
Example: ***!S2*** is the second non-stego-related project.

!S1. RAID Reconstruction

Introduction & Background

RAID (Redundant Array of Independent Disks) combines multiple disk drives into a single logical unit for the purposes of data redundancy or performance improvement. The single logical unit can appear as a single mass storage volume or drive. Data is distributed across the drives in one of several ways, referred to as RAID levels, depending on the specific level of redundancy and performance required. The different schemes or architectures are named by the word RAID followed by a number (e.g. RAID 0, RAID 1). Each scheme provides a different balance between the key goals: reliability and availability, performance and capacity.

Issues and/or Problems Being Addressed

Suppose that you're a cyber forensic analyst You are given a group of disks confiscated by law enforcement and asked to determine if these disks were part of a RAID volume. And if they were, to reconstruct the RAID volume so that it can be read and investigated. Since you have no other information, you will need to analyze the disks to determine whether the disks contain information that tells you if they were part of a RAID system, what kind of a RAID system and how to recreate the original RAID system. Law enforcement (including the FBI) has asked for such a system.

Possible Approach

1. Understand what RAID is, the different types of RAID, and how it works.
2. Procure a RAID card or use the built-in RAID on the motherboard. (Provided)
3. Obtain 4 lower capacity disk drives. 80GB disks that we have will work
4. Wipe the drives clean
5. RAID 1 (Mirrior)
   a. Connect two disks in a RAID 1 configuration with no data.
   b. Forensically analyze the disks to determine what was put on the disks by the RAID controller. We have tools available to do this such as EnCase and FTK. Also dd.
   c. Add data. It could be an OS and data files or just data files
   d. Forensically analyze the disks again.
6. RAID 0 (Stripe)
   a. Connect two different disks in a RAID 0 configuration with no data.
   b. Forensically analyze the disks to determine what was put on the disks by the RAID controller.
   c. Add data. It could be an OS and data files or just data files
   d. Forensically analyze the disks again.
7. RAID 5 (Parity). [Optional] Same as steps 5 and 6, but use 3 additional disks.
8. For each of steps 5, 6 or 7, develop a process for reconstructing a RAID volume from the disks,
9. Implement the processes.
10. Using a different controller, repeat the above to determine any differences.

Project Goals

- Understand RAID systems
- Understand what a RAID controller puts on the disks to cause them to operate as part of a RAID system
- Determine, given two or three disks, whether they were part of a RAID system or not.
- If they were, reconstruct the RAID system as though you have no prior knowledge.

Project Deliverables

A well written technical paper, a well written user guide and a well developed technical presentation are requirements. For two-semester projects, these will be required at the end of each semester, and at the end of the 2nd semester a robust working software tool will be required and demonstrated. Finally, a formal presentation and demonstration at ForenSecure15 in April 2015 will be required for all good projects.

Estimated Level of Effort and Challenges

Possibly one semester if only steps 5 & 6 are done. Two semesters if step 7 and 10 are also included. Take 549 to finish project. Code will either need to be developed or a suitable tools found to do the analysis and reconstruction.

!S2. IPv6-Auto Lab

Introduction & Background

Using a virtualization platform such as vmware – kvm – xen – hyper-V – oVirt
We want to make a small network entirely connected via IPv6 internally using openvswitch. This involves writing code to automate your deployment. The scope of this project would be to define a small network of PCs switches and possibly routers all communicating via ipv6.
In addition to this requirement – the system would be accessible only via RSA keys and need to shut it self down on the hour and re-launch itself on the 15s.

Issues and/or Problems Being Addressed

The problem being addressed is the nature of IPv6 and how it is used for routing and switching and passing of data (RSA keys) via IPv6.

Possible Approaches

Chose a virtualization platform and start looking at ipv6 configuration.

Project Goals

Understand the difference between ipv4, ipv6 and ipv4 v6 tunnels and how they affect networks.

Project Deliverables

Presentation and documentation as well as a working demonstration.

Estimated Level of Effort and Challenges

One semester. Networking and Linux experience will be helpful.

Starter References

http://blog.pcbsd.org/2011/06/ipv6-only-version-of-pc-bsd-9-0-available-for-world-ipv6-day/
http://en.wikipedia.org/wiki/Ipv6

http://openvswitch.org/ · open virtual switch – combine with something like kvm and xen

http://www.virtualizationadmin.com/articles-tutorials/vmware-esx-and-vsphere-articles/installationdeployment/vmware-understanding-virtual-switch.html
http://blog.superuser.com/2011/02/11/did-you-know-that-ipv6-may-include-your-mac-address-heres-how-to-stop-it/
http://twit.tv/show/floss-weekly/203 -> ovirt

http://www.youtube.com/watch?v=lMcf6LxMgYI · ipv6

http://www.youtube.com/watch?v=uNb7wd0-jpI -> ipv6

http://twit.tv/show/floss-weekly/131 · Vyatta open source router

!S3. Dynamic Malware Detection using Network Traffic Analysis

Introduction & Background

Botnets are posing a bigger and bigger threat than ever before. The *Security Threat Report 2014* by Sophos states: "*In the past 12 months, botnets have become more widespread, resilient and camouflaged… Botnet operators are also faster and more effective at responding to countermeasures.*" The *2014 Data Breach Investigations Report* by Verizon indicates that "*More powerful botnets and reflection attacks have helped drive the scale of DDoS attacks up 115% since 2011.*"

This project will help demonstrate the detection of botnets independent of any specific botnet signature. Rather, it will take advantage of some essential properties of botnets: in particular, their communication patterns. Network flows will be collected and analyzed to determine the communication targets and activities using the capabilities of network routers or tools like Argus. No attempt at decryption will be made. Based on the analysis of these patterns, a probability of the presence of botnets will be determined. This analysis will be verified with empirical analysis.

Issues and/or Problems Being Addressed

Botnets have become a primary platform for attacks on the Internet. They are continually evolving and becoming more complex, making their presence difficult to detect via traditional means, yet they maintain some essential identifying characteristics. This project will demonstrate that botnets can be detected with a high degree of confidence without any a priori knowledge of any specific botnet signatures.

A Possible Approach

1. ***Native Data Collection and Analysis***
   a. Create an isolated network of computers with a single known botnet vulnerability. Use virtual machines or an emulated API like Wine.
   b. Create a set of categories of network flows such as sources and destinations and what is being done.
   c. Monitor, document and analyze communication flows on a clean system. Use a tool like Wireshark.
      i. Gather communication data (i.e., network traces that indicate who is communicating with whom).
      ii. Gather activity data (i.e., who is doing what).
   d. Run the botnet in this system using a centralized communication structure. Suggested botnets : Sdbot, Zeus.
   e. Monitor, document and analyze communication flows as described in step (3) above.
   f. Repeat steps (4) and (5) with a different botnet utilizing a P2P communication structure (suggestions: Nugache, Storm or P2P version of Zeus).
   g. Analyze your data to gain an understanding of botnet communication behavior.
      i. Compare the data gathered in steps (3) and (5) to the data gathered on a clean sandbox system. Identify the markers that indicate that the systems have been infected with botnets regarding communication and activity patterns.
      ii. Compare the data gathered in steps (3) and (5). Determine the differences in the communication and activity patterns.

2. ***Hybrid Data Collection; Native Analysis***
   a. The restrictions surrounding a sandbox may limit the ability to collect effective communication traces as the botnets typically require communication with a command and control center (centralized or distributed). If this becomes problematic, the student may wish to collect botnet network traces collected from other, publicly-available sources. The activity traces should still be collected locally as even when botnet attempt communication and it is failed or blocked, the attempted activity can be traced.
   b. In this approach, follow the steps above, but collect network traces from sites such as Netresec web site (see Malware section for an index of publicly-available malware traces).

Project Goals

Goals for Issues/Problems Stated Above:

- Develop a proof-of-concept botnet detection mechanism using a clustering analysis of network flows based on basic knowledge of basic botnet behavior.
- If possible the detection mechanism should not be dependent upon specific botnet signatures or architectures.

Goals for Student's Understanding:

- Intrinsic behavior of botnets
- Different botnet architectures (Centralized, P2P)
- Network flow monitoring and analysis

Project Deliverables

The project deliverable would be a technical paper describing the lab setup, use of tools, and the approach used to demonstrate the dynamic analysis for botnet detection. The paper would also describe the methods used to collect and analyze the data. Finally, the results of the analysis and an assessment of the efficacy of the method to detect botnets would be presented.
This paper would be the basis for a potential presentation at ForenSecure15.

Estimated Level of Effort and Challenges

This is a two semester project. Take 549 in spring 2015 to finish project.
Challenges:

1. The restrictions surrounding the collection of communication traces in a local sandbox may present specific challenges as botnets typically require communication with a command and control center (centralized or distributed). If this becomes problematic, the student may choose to go with the hybrid approach described above (option 2 under "Possible Approach").
2. Collecting network flows in an efficient manner (i.e., without being overwhelmed with data).
3. Efficiently analyzing the data. There may be coding involved to provide this capability. Another option: look at the fcapture tool in Debian Linux.

Starter References

- *"Correlation-based Botnet Detection in Enterprise Networks"*
- *"BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection"*
- *Botminer Presentation (points to similar projects, tools…)*
- *Botnet Dection by Monitoring Similar Communication Patterns*
- *Argus Real Time Network Flow Monitor*
- *Netresec: Publicly Available PCAP files (network traces) – includes malware traces*
- *Wireshark web site*

!S4. Evaluation of Patch Management Alternatives

Introduction & Background

Patch management is the process for identifying, acquiring, installing, and verifying patches for products and systems. This is one of the key challenges in adhering to good security practices in industry today. Keeping the software and firmware in an enterprise up to date is important because they mitigate software flaw vulnerabilities. The application of the latest patches significantly reduces the opportunities for exploitation.

However, there are issues in maintaining current patch levels, especially in real-time, high availability systems. Patches can break other applications and require extensive testing to ensure compatibility. Many patches are applied at the OS level and require OS reboots. This is disruptive and can cause loss of data and interruption of services. In addition, the procedure to perform the update is relatively complicated. As a result, many corporations require an update procedure, which also needs to be tested, along with a procedure to back out the update if complications arise.

Because of the resources and risks associated with the application of patches, there is a tendency to delay patch deployment by bundling them and performing the updates only at specific intervals (e.g., quarterly). This makes Patch Management more efficient, but it also increases the window of opportunity to exploit the vulnerability because of the intentional delay in releasing the patch.

This project will explore other mechanisms to mitigate the risk associated with software flaw vulnerabilities and determine whether they present a better risk/reward profile.

Issues and/or Problems Being Addressed

There are many challenges in executing a successful Patch Management program, especially for real-time, high availability systems. It demands many resources and introduces risk from loss of data and interruption of services. Alternatives need to be explored whereby these software flaw vulnerabilities can be mitigated with lower cost and risk.

Possible Approaches

1. Set up two identical systems (A and B):
   - They would be configured as servers.
   - Linux is the preferred platform, but Windows is a viable alternative.
   - These can be virtual systems set up with a hypervisor such as Xen or Hyper-V
   - Each system would have the same obsolete OS and database patch levels database with known vulnerabilities for which patches are available.
2. Collect baseline data
   - Document the patch level and associated vulnerabilities to which the systems are vulnerable
   - Perform a security scan on A using Nessus or an equivalent network/configuration vulnerability scanner. Note the vulnerabilities and missing patches discovered at the OS and database level.
3. Update patches on system A and collect data
   - Update to the current OS and database patch levels on A. Re-run the security scan and determine the percentage of vulnerabilities resolved that are associated with outdated software levels.
4. Install anti-malware on system B and collect data
   - Install anti-malware software on B. Suggestion: Use the McAfee product that provides the best low level (e.g., OS and database) software protection. Compare the number of items discovered to the vulnerabilities noted in bullets (2) and (3) above.
     - Variation 1: Install different anti-malware packages to determine their relative efficacy.
     - Variation 2: Install several anti-malware packages on one system and determine the degree to which the overall effectiveness is improved.
5. Re-run steps (1) – (2) above. Then take the systems to a public Wifi network and repeat the remaining steps. This enables the collection of data in the presence of an environment with a higher potential for attacks.
6. Analyze the resulting data as described in the “Project Goals” and “Project Deliverables” sections below.

Project Goals

Goals for the issues stated:

- Compare Patch Management with its alternative, anti-malware
- Provide an analysis, comparing these mechanisms with respect to:
  - Effectiveness (i.e., percentage of vulnerabilities addressed)
  - System Impact (reboots, load on system, etc.)
  - Human Impact (Complexity of operation, overhead, etc.)
  - Manageability (i.e., cost of keeping current in the face of steady influx of new vulnerabilities)
- The above analysis should be limited to Operating System and database vulnerabilities as these are the ones most likely to have the greatest impact when patching.

Goals for students' understanding:

- Greater insight into Patch Management and its challenges
- Greater insight into the risk/reward evaluation that is associated with implementing security policies in the commercial sector.
- Familiarity with anti-malware packages.
- Perspective on the larger picture of vulnerability mitigation.

Project Deliverables

The project deliverable is a technical paper that details:

- The system configuration
- List of vulnerabilities/missing patches
- Results of network/configuration (e.g., Nessus) scans
  - before and after on system A
  - protected vs. public network on system A
- Anti-malware package(s) used
- Results of anti-malware execution
  - protected vs. public network on system B
  - potentially one product vs. another or one product vs. multiple on System B
- Comparative analysis of Patch Management and Anti-Malware using the metrics described in the section above.

In addition, a presentation from this technical paper can be presented at ForenSecure15.

Estimated Level of Effort and Challenges

This is likely a two semester project.
Take 549 in spring 2015 to finish project.

Some of the challenges associated with this project:

- Setting up and configuring the virtual systems with "old" software with missing patches and cataloging the vulnerabilities associated with the missing patches
- Mapping the network/configuration scan results to the anti-malware output.

Starter References

- *"A Comparative Analysis of Anti-Malware Software, Patch Management, and Host-Based Firewalls in Preventing Malware Infections on Client Computers"*
- *NIST Special Publication 800-40 Revision 3 "Guide to Enterprise Patch Management Technologies"*
- *Nessus Vulnerability Scanner*

!S5. Characterization of Initial Malware Infections

Introduction & Background

Static and dynamic analysis will be conducted to determine how quickly clean virtual systems become infected with malware based on one of several possible factors (system type or application image). The effect on initial infection rates and types of infection will be determined. Research will be conducted to suggest ways that might slow initial rates of infection of devices.

Issues and/or Problems Being Addressed

It is said that devices are infected within minutes of being connected to the internet. This project will verify that claim and come up with data regarding the time to initial infection and characterize the types of malware that are first to infect the host. This information will be used to better understand initial attacks and postulate ways to slow them.

Possible Approaches

1. Create a collection of virtual systems using a hypervisor such as Xen or Hyper-V. Choose one of the following factors by which to vary these virtual systems:
   a. System type
      e.g., Windows, Linux, etc.
   b. Configuration/Application Image (within same system type)
      e.g., server image (complete with web servers), basic image, etc.
2. Be sure that the images used do NOT have the latest patches installed. Document the patch level of the OS, databases, web servers and other key software applications.
3. Disable local firewalls, to the extent possible. Document the firewall configuration.
4. Take the system into a public Wifi location. Activate the virtual systems and have them perform some baseline activity.
5. Take several snapshots of the virtual systems at selected points in time for offline examination. Get several data points for each device type by wiping the device after use and re-inserting into the public Wifi network.
6. Analyze the resulting snapshots for evidence of malware infection. Determine the rate of initial infection based on either system type or application image. Also determine the type of initial infections.
7. Postulate ways that initial infection rates could be slowed based on the above analysis.

If time allows, the steps above could be repeated in different network environments (e.g., IIT campus network, home Wifi network) to determine how initial infection rates vary by network environment.

Project Goals

Goals for issues/problems:

- Determine the average infestation rate by system type or software configuration.
- Characterize the type of malware infections.
- Suggest ways to potentially slow initial infection rates.

Goals for student's understanding:

- Improve understanding of static forensic analysis
- Expanded knowledge of Intrusion Prevention mechanisms.
- Improved understanding of virtual environments.

Project Deliverables

The project deliverable is a technical paper that provides data and information on the following key points:

- Description of environment and approach
- Initial infection rates on various systems or configurations.
- Characterization of initial malware infections
- Suggestion for slowing rate of initial infections (per system or configuration)

Estimated Level of Effort and Challenges

Effort is estimated at one semester, two if student if unfamiliar with forensic tools. If two semesters are needed, take 549 in spring 2015 to finish project.

Starter References

- *"Before You Connect a New Computer to the Internet – US-CERT"*
- *"Where Only Fools Dare to Tread: An Empirical Study on the Prevalence of Zero-Day Malware"*
- *Hyper-V Hypervisor Overview*
- *Xen Hypervisor*
- *The Honeynet Project (contains a wealth of applicable tools, including honeypot emulators, dynamic analysis tools, capture tools and web server emulators)*