# Global Information Assurance Certification Paper

# The Value of Automating Malware Sample Collection

*GSEC Gold Certification*

Author: Joseph H. Schwendt II, joe@schwendt.com

Adviser: Lori Homsher

Accepted: April 4th 2006

## Outline

## 1. Introduction

Malware sample collection is an essential yet often overlooked part of incident handling. It is typically omitted due to the time it takes to collect samples when the primary goal is typically recovery. Collecting malware samples is essential for identifying new and unknown threats, as well as verifying known threats from morphing. In order to streamline the process of malware sample collection, it is imperative that the process be as automated as possible. With this automation, the value of malware sample collection becomes a positive return on investment.

This paper is intended for incident handlers to learn about the value of automating malware sample collection, and how the samples can be used to increase the overall security posture of an organization. A case study will provide insight to an in-use operational process. The paper will also compare typical manual data gathering frustrations to the automated methods. Finally, this paper will demonstrate the value automated malware sample collection brings to fighting malware by way of practical timesavings and avoided loss.

## 2. The Malware Sample Collection Process

The process of collecting malware samples encompasses first infection to first abatement. The process starts when the first system breach is suspected. This incident is

typically reported to a central incident handling team, but could be reported to any Information Technology representative within an organization. Once reported, it is the responsibility of the assigned incident handler to respond to the suspect system as quickly as possible. Incident handlers are typically people skilled in information systems forensics. The first goal of the incident handler is to determine if the system is a threat to the organization, and if so, remove it immediately from the network to prevent further spread. This has to be accomplished without altering the state of the suspect system to prevent forensic tampering.

Once the system is cleared to be worked further, the incident handler then needs to gather as much information from the system as is necessary to determine if there is indeed anomalous behavior. If determined to be infected, the incident handler then attempts to gather a sample of an infected file and confirm its observed behavior in an isolated lab environment. If the sample is found to be unknown, the incident handler then works with liaisons to Anti-virus vendors to provide updated definitions to protected against further infection and remediate any existing infections.

In the event the sample is known, it is important to verify that the sample has not morphed. This can typically be accomplished by comparing file hashes (MD5/SHA1) to a library of known malware. When a potential new variant is found, it needs to be treated as unknown malware.

After the Anti-virus vendors provide updated definitions, it is wise to verify they are able to remediate against the sample in question to ensure it operates as advertised. Once the incident handler is satisfied with results, they would then recommend immediate deployment of the updated definitions.

This process continues as much as required until all malware is known. Since malware is a constantly evolving threat, this process is used as often as needed to ensure the organization is kept secure.

## 3. Automation Return on Investment (ROI)

Automating manual processes provides two benefits: speed and consistency. Typically the return on investment for automation is measured by factoring the cost to implement and maintain the automation verses how long it will take to recoup the timesavings. The following equation captures the essence of Automation ROI:

$$ROI = \frac{NetBenefits}{Cost} \times 100\% = \frac{(Benefits - Cost)}{Cost} \times 100\%$$

In this case, Cost is the time to build the tool plus time to maintain the tool. Benefits comes in terms of time saved by using the automation. Thus, the new equation to reflect this:

$$ROI = \frac{(TimeSavingsOfAutomation - (TimeToBuildAutomation + AutomationMaintenanceTime))}{TimeToBuildAutomation + AutomationMaintenanceTime} \times 100\%$$

The timesaving of automation is in terms of time saved per use of malware collection process. Since this is highly variable, $\alpha$ will represent the number of uses of the process. The use of automation tools will reduce the time to capture malware samples by a factor of 5. The time to manually capture malware samples per event is reflected by $\theta$. Thus, Benefits equals $5\alpha\theta$.

The time to build the automation is usually reflected in terms of a multiple of the time to capture malware per event $\theta$. The maintenance for the automation will also be expressed in this same multiple. Thus, the time to build automation should fall around $30\theta$ and the on-going maintenance is around $0.25\alpha\theta$. This factors the equation to:

$$ROI = \frac{(5\alpha\theta - (30\theta + 0.25\alpha\theta))}{(30\theta + 0.25\alpha\theta)} \times 100\%$$

If we are to plug in some plausible values for $\alpha$ and $\theta$, where $\alpha$=30 times/year and $\theta$=3 hours, the ROI would be 300% after the first year. These are conservative numbers, but clearly show a positive return on investment.

$$ROI = \frac{(5(30)(3) - (30(3) + 0.25(30)(3)))}{(30(3) + 0.25(30)(3))} \times 100\% = \frac{450 - (90 + 22.5)}{90 + 22.5} \times 100\% = \frac{337.5}{112.5} \times 100\% = 300\%$$

Quantifying the costs associated with malware is a mostly fruitless venture. There are

endless variables that could be taken into consideration to provide a truly qualified cost. In the case of this paper, we are attempting to quantify the costs vs. benefits of malware sample collection automation. The cost savings are associated with time saved to perform malware sample collection multiplied by resource utilization costs, taking into consideration the malware rate of infection and impact due to infection.

The costs of malware infections can be captured with the following equation:

$$ElapsedTime \times \left( (\$impactOfMalwareInfection / Machine) \times (\$to\,\mathrm{Recover}FromInfection / Machine) \right)^{RateOfInfection}$$

Automation helps to lessen this impact by directly impacting the elapsed time. It is a linear correlation between time elapsed since first infection and the total cost that will be incurred to recover. The less time spent determining what is impacting the environment, the faster recover will be had.

To explore the effects of the calculation, we will use two examples. In example A, we will assume no automation of malware sample collection is in use. In example B, we will assume automation is used.

Example A: $5hours \times \left( (\$500 / Machine) \times (\$200 / Machine) \right)^{50 Machines / hour} = \$5e^{250}$ lost opportunity cost

Example B: $2hours \times \left( (\$500 / Machine) \times (\$200 / Machine) \right)^{50 Machines / hour} = \$2e^{250}$ lost opportunity cost

In neither case are the numbers pleasant, but they to illustrate the direct linear

relationship between time to react and contain/recover and lost opportunity cost. Malware sample collection automation helps to speed up the reaction time and help to jumpstart containment and recover actions.

## 4. Manual Verses Automated Malware Sample Collection Processes

Without the use of automation, collecting malware samples and information from potentially infected machines is a time consuming and error prone process. There are two general cases to consider, one where the machine is accessible to the Incident Handler and the other where it isn't. In the case that the machine is accessible, the Incident Handler will use an information gathering process and toolset that they have developed over time to gather the samples. The Incident Handler tends to use a "bag of tricks" that they have found useful over time, and this process/toolset typically varies widely from analyst to analyst. The various tools/procedures employed will often provide highly inconsistent data gathering, with results being just as inconsistent between Incident Handlers. This process also takes a considerable amount of time due to it being entirely manual. This could be time better spent doing analysis rather than simple data gathering.

In the case where the Incident Handler doesn't have hands on access to the machine, the process becomes even more complex and prone to error. The Incident Handler must now

work with the end user over the phone to walk them trough running the various tools required to gather the sought after information. The time required to do this is considerably longer due to the relaying of information.

By automating this entire process as much as possible information could be delivered in a consistent format that any Incident Handler can work with. It also makes the malware sample collection process happen as quickly as possible, allowing the Incident Handler valuable time to concentrate on analysis.

## 5. Case Study

While the market is ripe for automated malware sample collection tools, one tool has emerged on the market as a strong open source contender: RAPIER (Rapid Assessment & Potential Incident Examination Report). RAPIER was developed by Intel Corporation as an internal project. Version 3.0 has been in use at Intel since July of 2005. It has been used several times since its initial rollout, providing a much more rapid response to reports of suspected malware in the environment. An example event was for the Snow.A virus (see http://vil.nai.com/vil/content/v_138727.htm for more information) of late February 2006. At Intel, the Help Desk received a report of suspicious activity from an individual machine in a lab. The Help Desk analyst worked with the caller to download RAPIER from an internal

website and walked them through the process of unzipping the file, running RAPIER.exe and running a Fast Scan. This whole process took the Help Desk analyst less than 5 minutes to explain and execute with the caller. The results generated were automatically uploaded to a central location and the team of Security Analysts was immediately notified to look at the results. The Security Analysts were able to analyze the results and extract a sample, which could be tested in an isolated lab. Once the sample was verified to be an unknown virus, it was sent to various Anti-virus companies for remediation to be added to Anti-virus definition files. It turned out that TrendMicro had already identified this virus, but McAfee and Symantec hadn't. The work done by the Security Analyst at Intel helped to ensure McAfee and Symantec's respective Anti-virus products could protect against and remediate this virus.

RAPIER is focused on reducing the time spent doing malware sample collection. From the time the Help Desk received the call to the time the Security Analyst had the RAPIER results was less than 30 minutes. Previously it would have taken nearly 15 minutes to just get a Security Analysts engaged with the caller of a suspect machine. The time to execute a manual malware sample collection process would typically vary from 1-4 hours. RAPIER is able to cut this time down dramatically while ensuring consistency and optimized resource utilization.

RAPIER will be launched to the open source community in June 2006 when it is

presented at the Annual FiRST (Forum of Incident Response and Security Teams) Conference on Computer Security Incident Handling in Baltimore.

## 6. Fundamental Operation of Malware Sample Collection Automation

For the purpose of examining the operation of an automated tool, this section will use RAPIER as an example (See Appendix A-C for a more detailed explanation of the individual components). RAPIER is designed to be run locally on suspect machines in an unaltered state. It collects potential malware samples loaded into memory, enumerates recent system changes, reports basic system configuration, exposes possible backdoors, enables some recreation of events, acquires text copies of all system logs and registry settings, maps all open ports to the processes connected to them, scans for known malware, and captures packets from the IP stack.

RAPIER is distributed as a universally compatible compressed file (.ZIP) via a web site accessible to all end users. Users should decompress the downloaded file onto writable media of choice. They should then simply run the tool. If online, the RAPIER will automatically check to ensure it is the latest version. It features the ability to validate the integrity of the entire program itself from a secure source (SHA1 and MD5 checksum verified). RAPIER is also designed to not extend its footprint beyond the directory it is launched from.

This is key to ensure the state of the system remain unaltered for forensic evidence.

Once RAPIER's graphical user interface is loaded, the user should select the appropriate modules for the malware suspected. The modules to be selected should be directed by the Incident Handler receiving the results. There should be a default list which enables the most common information to be gathered, and will take approximately 10 minutes to run, i.e. Fast Scan. After selecting the appropriate modules, the user simply needs to start the information gathering process. If the system is online when running, the results should be automatically uploaded at the end of running the selected modules. If the system is offline however, the user will need to copy the results to a computer which does have network access and upload them to the central repository for analysis by the Incident Handler.

RAPIER is designed to collect volatile state information from the target system. For this reason the end user should not disconnect, shutdown, or alter the system state until after running the tool unless directed to do so by the security analyst. This may alter the effectiveness of collecting malware samples since some malware is now smart enough to detect when the network has been disconnected and go into hibernation. If however the system is causing dramatic harm to the environment, disconnecting that system prior to running the tool is a good idea.

RAPIER is also designed with ease of use in mind. It is expected that any general

support organization can run it. Any help desk analyst should be able instruct individuals to use it over the phone. For those support organizations that are physically located on-site, they can run it directly on a suspect system eliminating the user from the mix. RAPIER should essentially be run by anyone who suspects they are infected with some form of malware to provide them with information which they can use to diagnose the state of the system

## 7. Malware Sample Collection Automation Tool Feature Requirements

- Fully Modular Design – Allows for rapid development of new information gathering methods. Can be custom tailored to any environment.
- Fully configurable GUI (Graphical User Interface) – Standard Windows application look and feel. Easy to use.
- Auto-update functionality with SHA1 and MD5 – Entire application should be self updating. Checks for new version on startup (if connected to network). Uses SHA1 and MD5 checksums to ensure files are verifiably current and authentic. Only those files in need of updating are transferred, which ensures network bandwidth is conserved.
- Compressed Results – Compression of results will reduce transfer times. The

compressed file should be password protected with a unique password to ensure virus scanners do not interfere with results.

- Results Auto-uploaded to central secured repository – If system is online, results are automatically sent to a central repository where an incident handler can analyze them in a lab environment.
- Email notifications – Incident Handlers are notified immediately on successful results upload using a standardize template which can easily be forwarded as a text message to pagers and cell phones.

## 8. Results Analysis

The results that the tool produces are not designed to be interpreted by the average user. They are tailored towards a trained Incident Handler, who is knowledgeable in the dark art of malware analysis. If an Incident Handler is unavailable, the results can be directly provided to an anti-malware software vendor, (i.e. McAfee, Symantec, Panda, Sophos, Trend Micro, etc.) through appropriate channels. Since the results may include PII (Potentially Identifiable Information), there should be a Non-Disclosure Agreement in place with the vendor prior to sharing results.

With the collected results, the Incident Handler should be able to:

- Determine if a system has been compromised
- Potentially determine the method of infection
- Determine the changes to the system since infection
- Determine how to clean and/or recover the system

The Incident Handler should start by looking at the results to get a general idea of the system configuration, what software is installed (as well as what version it is), and the network configuration. From there, the handler should be able to narrow down possible infection vectors. For instance, if a specific version of Microsoft Office is installed, they can usually discount vulnerabilities not affecting that version. Based on the network configuration, they might be able to tell which network interface the infection may have come from. They can also narrow their search to the IP address(es) of the system. If provided by the tool, the handler can look for non-standard files created within the last 24 hours. As long as the infection took place within the last 24 hours, this will help to narrow down the list of possibly infected files. Also, mapping that list of files against the list of files from the output of tools like fport, we can see which files have been created in the last 24 hours and have a network port open. Many times when malware is actively spreading itself, it will talk to the network directly, and will thus show up on this correlated list.

Other infection vectors, such as Browser Helper Objects from Microsoft Internet

Explorer, should also be investigated. Since most malware will stay memory resident, examining the samples collected from the executable of all running processes can provide the exact malware executable. Also, using full physical memory dumps, the handler can look for certain chunks of suspect text or byte code segments. Finally, the output of a network sniffer will provide the analyst with a detailed view of exactly what the system is doing on the network. This can help to identify protocols or ports to disable for containment or for identifying other infected systems on the network through the creation of network signatures. Together, the results from the various modules can paint a very detailed picture of the state on the potentially infected system. In the hands of a trained Incident Handler, this information can be invaluable information to process of malware containment and remediation.

## 9. Conclusion

Malware sample collection automation fills a very important niche in the Information Security response toolset. Prior to considering automated tools, Incident Handlers must either have physical access to a system or manually walk end users through the painstaking process collecting system information and malware samples. Using automated tools, users are able to quickly gather the information the Incident Handler needs to respond to a possible malware infection, and remediate their system to minimize downtime. Automation also allows for faster identification of malware signatures and infection vectors, so that containment steps

can be taken before the infection grows to out of control proportions.

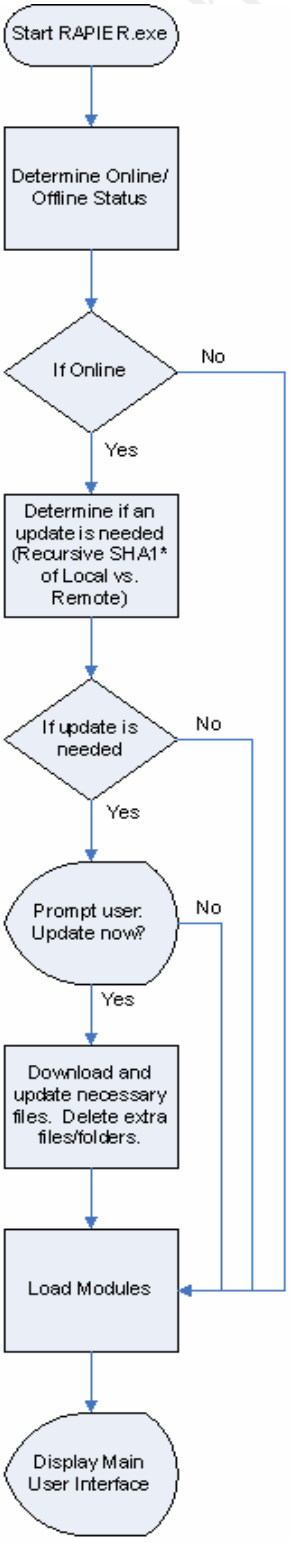
Using automated malware sample collection tools, organizations are able to respond to possible malware infections in record time, and are thus able to prevent serious service interruptions. It is easy to show a positive ROI for these tools given the burden of developing the automation. Using open sources tools such as RAPIER, the ROI is highly positive for any organization, making malware sample collection a necessary part of an incident handling process.

## Appendix A: RAPIER 3.0 Engine Design

RAPIER 3.0 is essentially a framework (engine) which runs individual modules to collect information and upload it to the central repository. The engine is responsible for the following tasks (see Figure 1):

- Determining if the system is online and can contact the server
- If online, determine if the application and modules are up to date and verifiably authentic.

- Update the application and/or modules if anything is not up to date or verifiable authentic if the user chooses to do so
- Present the user with a GUI (Graphical User Interface) (See Figure 3)
- Allow the user to configure the following options:
    - Module Directory – Directory where the RAPIER modules are located. Defaults to a directory called “Modules” in the same directory as the RAPIER.exe
    - Results Directory – Directory where the results from running




* Recursive SHA1 Algorithm is being submitted as a separate invention

**Figure 1 – RAPIER 3.0 Startup Process**

the selected RAPIER modules should be placed. Defaults to a directory called “Results” in the same directory as the RAPIER.exe

  - Auto-ZIP results – Enables/Disables the automatic ZIPing (compression) of the results at the end of running the selected modules.
  - Auto-Upload results – Enables/Disables the automatic uploading of the results at the end of running the selected modules.
  - Only enable-able if the system is online and Auto-ZIP results is enabled.

- Allows the user to upload a results ZIP file collected from an offline system
- Allows the users to view online help
- Allows the user to individually select the modules. Quick select buttons are available for end user ease of use. “Fast Scan” is targeted at systems as a default if user is unsure what to select. It is designed to run in approximately 10 minutes. “Slow Scan” enables all “Fast Scan” modules as well as some additional modules which take much longer to run and provide much more detailed information. A “Slow Scan” can take as long as 2 hours to run and produce results as large as 1.5 times the size of the physical memory in a the target system.

- Runs the selected modules. Passes the path to the results directory to each individual module (see Figure 2).

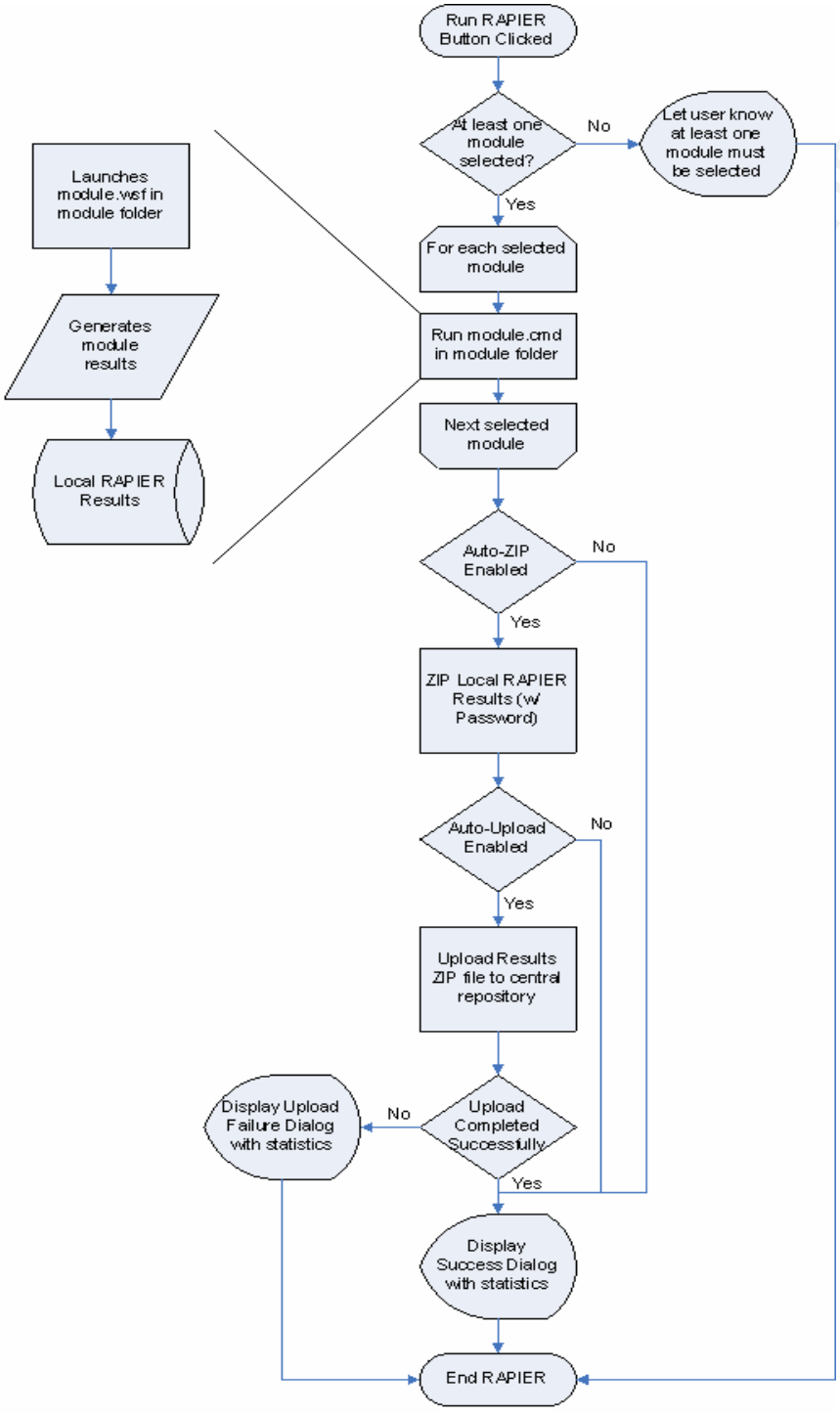



**Figure 2 – Run RAPIER Modules**

## Appendix B: RAPIER 3.0 Module Design

All RAPIER modules are designed to function completely independent of the RAPIER engine. They are built to accept one command line

option, which is the path to the directory to place the module results. Each module should create a <Module Name>.log file in the results directory, which contains basic information. Some modules may need to create additional files or directories in the results directory. For example, the DumpProcs module creates a directory and copies the exact executable from memory of each process that is currently running on the system to that directory.

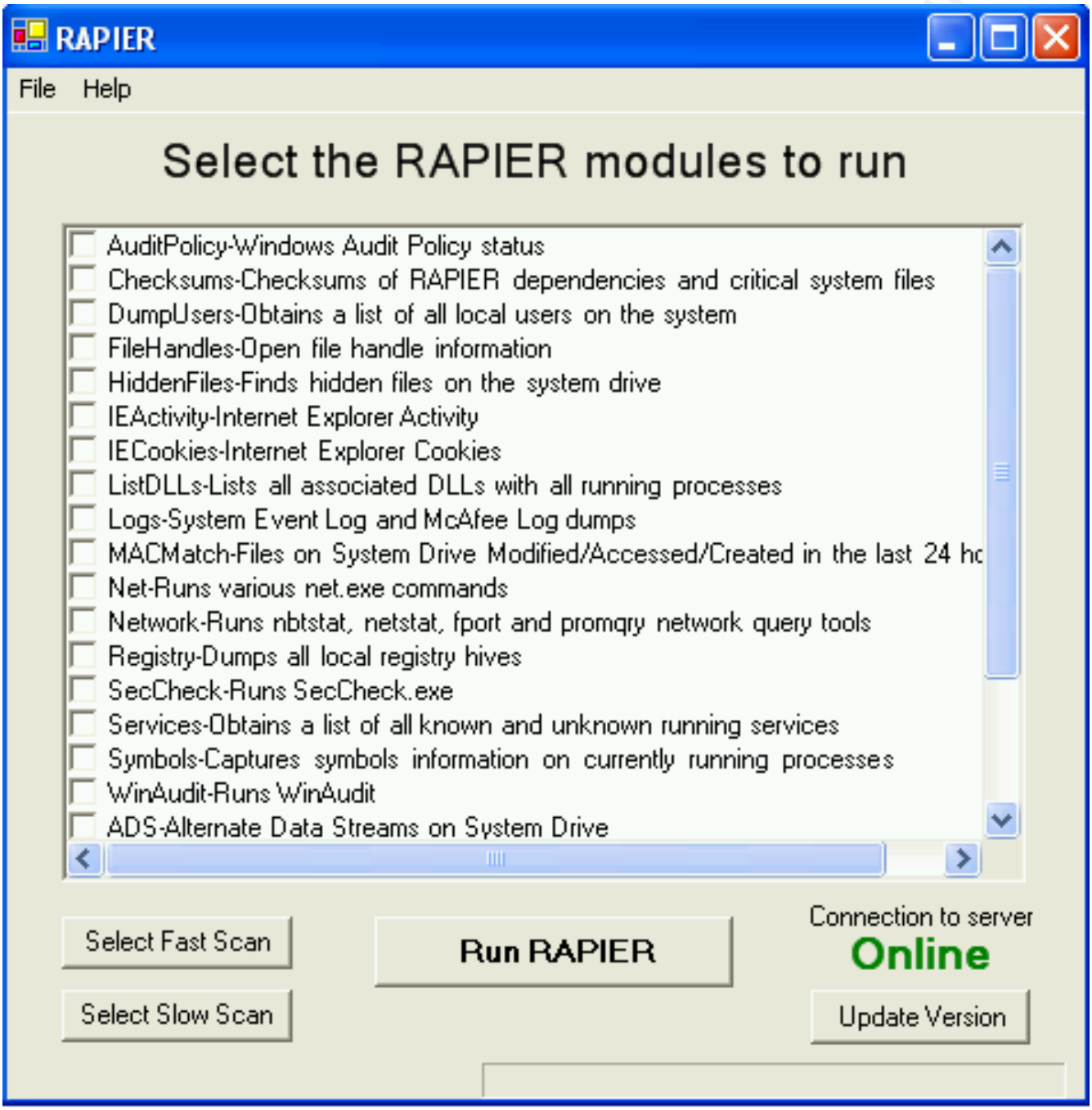



**Figure 3 - RAPIER 3.0 GUI (Graphical User Interface)**

Modules are written in Microsoft VBScript. They require Microsoft WSH (Windows

Scripting Host) 5.6 and Microsoft WMI (Windows Management Interface) 1.5 to be installed on the target system. These are standard components on all Intel Managed systems. They also utilize several VBScript libraries and external tools in the "Tools" directory. Once such library is IPACE.vbi. This is an extensive library developed as part of IPACE (Intel Patch Assistant and Compliance Enabler). Other external tools are third party freeware security tools from various sources, which provide low level access to the operating system internals. Some of the external tools have been developed in-house explicitly for use with RAPIER, but could potentially be used by any application. These include AutoproxyResolver.exe (Resolves the proxy server for a given target URL based on a given Autoproxy URL) and L3Sniffer.exe (Layer 3 (IP stack) network sniffer which dumps raw packets in byte format that can be easily converted to tcpdump format with the use of text2pcap.exe from Ethereal).

Individual module behavior can also be controlled though configuration files in the "Conf" directory located in the same directory as RAPIER.exe. Modules will need to be written to take specific advantage of this functionality.

Since modules run completely independently of one another and the engine, they can be built to collect any desired information a security analyst could possibly need from a potentially infected system. Using the VBScript libraries, module development is a very rapid process. A new module can be developed and tested within an hour to enable rapid

response to a new type of malware. The engine's self-updating mechanism ensures that new modules can be deployed as rapidly as they are available. If a module is no longer needed, the auto-updating mechanism will ensure it is removed from the system.

## Appendix C: RAPIER 3.0 Modules

RAPIER 3.0 comes bundled with a rich set of modules. The following briefly explains each module's capabilities:

- Fast – Basic modules which typically run within 5 minutes individually (10 minutes collectively)
    - AuditPolicy – Reports the status of the built in Windows Audit Policy. If an audit policy is disabled, Windows will not capture information for that type of policy. Several types of malware have been known to disable auditing to cover its tracks.
    - Checksums – MD5 Checksums of RAPIER dependencies and critical system files. This is useful for determining if RAPIER and the Operating System are able to function in an uncompromised manor. If any of these files are suspect, so are the results that RAPIER produces.
    - DumpUsers – Obtains a list of all local users on the system. Non-

standard users can typically indicate a trojan or backdoor.

- FileHandles – Enumerates a list of all open file handles on the system. Open file handles indicate that a file is being accessed either for reading or writing.
- GeneralSysInfo – Gathers a bevy of general system information, NIC (Network Interface Card) settings, and installed software versions. This module provides overview information, which can help to narrow down infection vectors.
- IEActivity – Gathers Microsoft Internet Explorer activity
- IECookies – Collects all stored cookies from Microsoft Internet Explorer
- ListDLLs – Lists all associated DLLs with all running processes.
- Logs – Dumps the System, Security, and Application Event Logs, as well as the McAfee Event log.
- MACMatch – Lists all files which have been Created, Accessed, and Modified on the system drive within the last 24 hours. This is usually a good starting point to narrow down suspect files.
- Net – Runs “net.exe SHARE” (enumerates windows file and printer

shares on the system), “net.exe START” (enumerates all running Services), “net.exe USER” (lists all local user accounts), “net.exe USE” (enumerates all mapped windows file and printer shares), “net.exe FILE” (lists open files on the system, the user who has the file open, and the number of locks on the file), and “net.exe SESSION” (lists sessions between the system and other systems).

- Network – Runs nbtstat (Displays protocol statistics and current TCP/IP connections using NBT (NetBIOS over TCP/IP)), netstat (Displays protocol statistics and current TCP/IP network connections), fport (Lists all processes and the associated TCP/IP ports) and promqry (Determines if any network interfaces are running in promiscuous mode).
- Registry – Dumps all local Registry Hives
- SecCheck – Runs SecCheck.exe, which gathers various system state information, much of which is covered with other modules, but summaries it nicely.
- Services – Obtains a list of all known and unknown services and their state. Uses the configuration files to determine if services are known.
- Symbols – Captures symbols information on currently running processes

  - WinAudit – Runs WinAudit.exe, which is similar to SecCheck, but provides other wide-ranging information not necessarily related to security.

- Slow – More specialized modules which typically take longer than 5 minutes to run and produce large output files

  - ADS – Scans the System Drive for NTFS Alternate Data Streams. A relatively unknown compatibility feature of NTFS, Alternate Data Streams (ADS) provides a method of hiding root kits or hacker tools on a system and allows them to be executed without being detected.

  - ddPhysMem – Dumps the entire contents of physical memory to a file. Note that the file this creates will be equal in size to the amount of physical memory in a system.

  - DumpProcs – Dumps the executables of all running processes. This is the most effective means of capturing a malware sample.

  - HiddenFiles – Lists all hidden files on the system drive and lists their last access time

  - WebCache – Dumps the Internet Explorer cache. This is also an

effective means of capturing a malware sample, since Internet Explorer is a popular infection vector

- Special – Modules with a highly specialized purpose which may take longer than 5 minutes to run
    - AVScan – McAfee Antivirus Command Line Scan of all fixed disks. The module will do an auto-update to the latest DAT file if the system is online prior to performing the scan. It will not clean or delete any files, only scan. This is useful for determining if the suspect malware is already known and detected by the latest DAT file.
    - L3Sniffer – Layer 3 (IP stack) network sniffer, which runs for 10 minutes or 10,000 packets, which ever comes first. This module produces a tcpdump compatible capture file by using text2pcap.exe from Ethereal to inject dummy Layer 2 information. The capture file can be analyzed using Ethereal or any industry standard packet analyzer.