Remote Fingerprinting and Exploitation of Mail Server Antivirus Engines

Jon Oberheide, Farnam Jahanian
Electrical Engineering and Computer Science Department
University of Michigan, Ann Arbor, MI 48109
{*jonojono, farnam*}*@umich.edu*

Abstract

Vulnerabilities within antivirus engines deployed at a mail server represent a serious risk to the security of organizations. If a sophisticated attacker is able to remotely probe a mail server and identify the particular antivirus engine used, he may craft a malformed message to exploit the engine with a low risk of detection. This paper explores how much information is exposed by these mail servers, how this information could be used effectively by an attacker, and what can be done to limit the exposure and reduce the impact of antivirus vulnerabilities. Towards this goal, we introduce and evaluate three techniques that can expose the vendor and version of antivirus software used by a mail server: message bouncing, detection probing, and iterative refinement. Through a series of experiments, we determine that over 28% of bounced messages expose information, 78% of response messages and 16% of delivered messages leak information through detection probing, and demonstrate the effectiveness of iterative refinement with a dataset of over 7200 malware samples and antivirus engines from 10 popular vendors. We also show that the most commonly deployed mail server antivirus engine is the most vulnerable engine and is one of the top offenders in exposing information. Finally, we conclude by suggesting methods of reducing the amount of exposure and discuss isolation techniques that may provide greater security.

1 Introduction

Antivirus software is the predominant method of protecting hosts and last line of defense against malicious software. However, due to the increasing scale and sophistication of malware, antivirus and its detection routines have become very complex. In particular, antivirus engines must be able to parse a vast range of complex file types, many of which are malformed and obfuscated to avoid detection. This complexity leads to the presence of vulnerabilities within the antivirus software itself which can be exploited by a maliciously formed input file. Significant numbers of vulnerabilities have been discovered in antivirus engines in recent years. For example, within the first four months of 2008, twelve vulnerabilities have been reported and confirmed in the ClamAV antivirus engine [8].

While vulnerabilities in antivirus are of concern in any deployment model, they are especially severe when antivirus software is deployed at a mail server to scan for malicious content and attachments. As most mail servers will accept delivery from any unsolicited party on the Internet, an attacker can craft a maliciously formed file designed to exploit an antivirus engine vulnerability and send it as an attachment to the target mail server. The resulting exploitation may compromise the integrity of the entire mail server and all of its incoming and outgoing email.

In order to craft a malformed attachment to exploit a specific antivirus engine while maintaining a low risk of detection, an attacker may elicit feedback from a mail server to assist in identification of the antivirus vendor or version used. Towards understanding such a threat, we introduce three techniques that can expose information about a mail server's antivirus software:

1. **Message Bouncing**: An attacker may address an email to an invalid recipient mailbox and gain information from the content and headers of the resulting bounced message.

2. **Detection Probing**: By sending an attachment to trigger the detection routines of the mail server's antivirus engine, the response to the probes may reveal information about the antivirus engine employed.

3. **Iterative Refinement**: Using specially-crafted attachments or malware samples that are only detected by a particular antivirus engine and no others, an attacker may be able to pinpoint the engine in use.

As each technique may vary in its effectiveness and risk of detection, we perform three experiments to evaluate them against real-world targets. We find that the first technique exposes information in 137 of 483 bounced emails (28.4%). In the second experiment, we employ detection probing with the EICAR test file [3] and observe an exposure rate of 77.9% of the response messages and 15.8% of the total delivered messages. In the last experiment, we successfully identify the antivirus software used by a University of Michigan mail server by leveraging over 7200 samples from the Arbor Malware Library [7], a collection of ten antivirus engines, and a SMTP feedback mechanism.

Additionally, we survey a number of security administrators about the antivirus they employ at their mail servers, which results in two interesting observations: (1) the antivirus engine which is most commonly deployed among the survey participants is the one that contains the most vulnerabilities of any vendor, and (2) many administrators may be unaware of the vulnerabilities in their antivirus engines and the associated risk when deployed in a mail server environment.

As detailed information on an antivirus engine may benefit a malicious party in their attack on a mail server, reducing the amount of information leaked and eliminating the feedback mechanisms that enable the exposure are significant steps in reducing the risk of the threat. However, we also explore alternate approaches to provide isolation between the mail server and its antivirus engine. By introducing a layer of isolation, we can reduce the impact of the underlying vulnerabilities within the antivirus software and increase the overall security of the mail server.

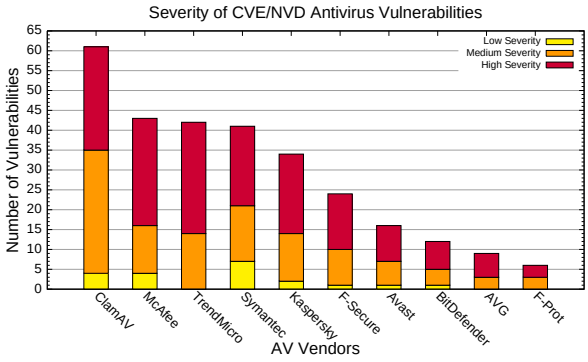

Figure 1. Number of vulnerabilities reported in the National Vulnerability Database (NVD) for ten antivirus vendors between 2005 and June 2008.

2 Complexity of Antivirus Software

Antivirus software is designed to detect malware and protect desktops, servers, mobile devices, and other computing assets against malicious attack. However, effectively achieving these design goals is a challenging task due to the vast ecosystem of malicious threats. The amount and sophistication of malware is ever-increasing as antivirus vendors scramble to adapt their products to address rapidly evolving threats. Developing the antivirus software to adequately protect against these threats leads to a considerable amount of complexity in terms of code base size, detection algorithms, and parsing routines. This complexity can lead to an increased risk of vulnerabilities within the antivirus software itself.

To provide effective protection, antivirus must possess a wide breadth and a significant depth of visibility into suspicious files. Therefore, the antivirus software must parse an enormous number of complex file types including archives, office documents, and any other format that may potentially contain malicious code. In addition, as malware authors employ exotic executable packers, mangled and malformed file headers, and obfuscation techniques to thwart static analysis, antivirus software must be able to process and analyze "the worst of the worst" as far as file formats are concerned.

Unfortunately, safely parsing these complex file types is a daunting challenge. Indeed, even the applications which create such files often have problems in

parsing their own file types [1, 6]. Simple programming mistakes (integer overflows, off-by-one errors, unbounded buffers), misinterpretations of file format specifications, and many other issues can lead in unsafe parsing routines which may result in vulnerabilities within the antivirus software. Attackers may craft a malformed file designed to exploit the vulnerabilities within the parsing routines when it is analyzed by the antivirus software, often leading to the execution of arbitrary malicious code.

Indeed, numerous vulnerabilities have been discovered in the antivirus engines of nearly every vendor. Figure 1 shows the number of vulnerabilities reported in the National Vulnerability Database [8] across the product lines of ten popular antivirus vendors between 2005 and June 2008. It is important to note that the majority of the vulnerabilities are a high severity risk (as determined by CVSS score) which often results in a full compromise of the particular antivirus engine. These significant numbers of vulnerabilities demonstrates not only the complexity of antivirus software, but also the high risk of vulnerabilities and possible exploitation when deployed in sensitive environments.

When a vulnerable antivirus engine is deployed in a mail server environment and is used to scan incoming mail and attachments for malicious content, the consequences can be dire. While in a host-based antivirus scenario, exploitation commonly requires user interaction in order for the antivirus to attempt to analyze a maliciously crafted file, a vulnerability in an antivirus engine deployed at a mail server can be triggered automatically by a rogue email from any sender with a malformed attachment. In essence, a vulnerability in the antivirus engine which was previously only locally exploitable in a host-based scenario is now remotely exploitable by any attacker who can send mail to the mail server. Even when the backend antivirus engine is run at a lower privilege level (e.g., ClamAV's `clamd` is commonly run as an unprivileged user), an attacker who is able to compromise the integrity of the antivirus process will still be able to access the contents of all the incoming email which is arguably the single most valuable asset of the mail server.

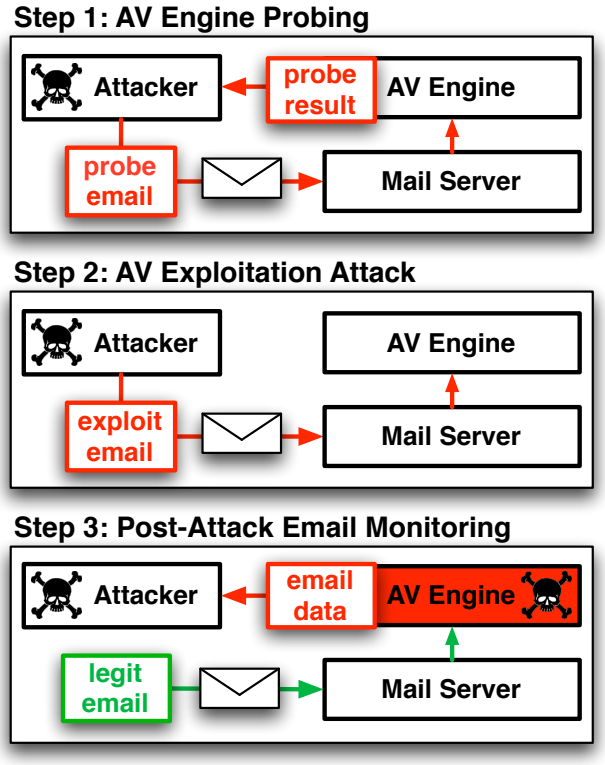

Figure 2. Example scenario of how an attacker could probe a target mail server, identify and exploit the its antivirus engine, and then monitor its emails.

3 Mail Server Information Exposure

3.1 Perspective of an Attacker

When performing a targeted attack or penetration test, it is important for an attacker to gain a foothold into the network through a high-value, low-risk target that will enable further compromises and intelligence gathering with little chance of detection. One type of target that satisfies these goals of the attacker is the target network's mail server. A compromised mail server can provide a huge amount of information throughout the entirety of an attack or penetration test due to the value of the email which it processes. However, modern mail transfer agents (MTAs) are been hardened against attack and popular MTAs such as Postfix and Sendmail have not suffered code execution vulnerabilities in several years.

Therefore, instead of targeting the MTA itself, an

attacker can target the antivirus engine that the MTA employs. However, from a blackbox perspective, the attacker may not know what particular vendor/version of antivirus software the mail server uses. If the attacker blasts the mail server with a voluminous amount of malformed attachments containing his exploit for every possible vendor and version combination of antivirus software, he exposes himself to a significantly higher risk of detection and may be prematurely disclosing sensitive parts of his malicious payload to any savvy administrators (e.g., 0-day exploits, custom backdoor routines, IP address of next stage downloader). In addition, if the attacker attempts to exploit a particular version that does not match the deployed antivirus software, the malformed attachment may actually crash the antivirus engine instead of exploiting it, resulting in near-certain detection. Therefore, being able to determine the vendor or version of the mail server's antivirus software in a low-risk manner represents a very advantageous ability for the attacker.

3.2 Remote Identification Techniques

Mail servers and their associated antivirus engines often provide both explicit and implicit feedback that convey information to its users regarding aspects of its operation (e.g., sending an error messages back to the sender when a malicious attachment is detected). While this information can be useful to legitimate users during normal operation, it may be abused by a malicious party. An attacker may be able to remotely probe or otherwise influence the mail server and its antivirus engine into a state that exposes information about the vendor or version of the antivirus engine.

We introduce three techniques that could be used by an attacker to remotely glean information in a low-risk manner from a target mail server and identify the antivirus engine employed:

1. **Message Bouncing**: Most mail servers return a bounced message to the sender when delivery fails to a invalid recipient mailbox(SMTP error 550). Headers and other information contained within this bounced message may reveal information about the antivirus software used by the mail server. Bounced messages may also reveal information about the mail server or its operating system platform that can be used to narrow the list of candidate antivirus engines. Since the attacker can send a message to a random recipient in order to generate a bounce and administrators do not usually monitor bounced messages, the attacker can maintain a high level of stealth by using this technique.

2. **Detection Probing**: If the antivirus engine of a mail server detects that an email has malicious content or attachments, it may send a notification message back to the sender. Similar to the first technique, this response may contain sensitive information exposing the vendor or version of the antivirus engine. As this technique requires that the attacker send an attachment that triggers the detection capabilities of the antivirus engine, it is less stealthy than the first technique but may result in a more detailed exposure of information from the mail server.

3. **Iterative Refinement**: If the attacker has offline access to a group of antivirus engines, he may be able to craft sample files that trigger only one particular engine. Combined with a source of feedback (e.g., rejected connection, auto-response/vacation message) from the mail server to determine when an attachment is accepted or rejected, the attacker may iterate over his sample set to determine the particular antivirus engine in use by the mail server. For example, if an attacker sends a file attachment that he knows is *only* detected by the Symantec antivirus engine and that attachment is rejected, he can assess with some degree of confidence that the mail server employs Symantec. Additional samples only detected by Symantec can be sent to increase confidence in the result. Once a particular vendor is identified, the attacker could reverse engineer signatures updates for that vendor to construct further probing samples and refine to a specific version of the antivirus engine. While this technique is the least stealthy of the three, it can be used as a last resort if the other techniques prove ineffective against the attacker's target.

4 Experimentation and Evaluation

To evaluate the effectiveness of the presented techniques for remotely identifying the antivirus software deployed at a mail server, we performed three distinct experiments. In addition, we performed a simple survey to gain further insight into the impact of mail server antivirus vulnerabilities.

4.1 Message Bouncing

The first technique investigates the exposure of information from messages bounced back to the sender when an invalid mailbox recipient is specified. Therefore, collecting a sample set of bounced messages is a simple task. We addressed test emails to invalid mailboxes for a large number of educational institutions and collected the bounced responses. In total, we collected 483 bounced messages and manually inspected the headers and content of each one to determine if any significant information was exposed.

Hostnames of the mail server involved in the delivery of the bounced messages revealed a surprising amount of information about the protection software used. For example, 36 out of the 483 bounces (7.4%) have the string "barracuda" in the hostname of a server involved in the mail delivery (e.g., barracuda.x.edu), corresponding to the popular email appliance company Barracuda Networks [2]. Similarly, 14 out of 483 (2.9%) have the string "ironport", corresponding to IronPort Systems [4]. Identifying over 10% of the mail servers simply by their hostname in the bounced messages represents a significant exposure.

The email headers and contents of the bounced messages can reveal even more detailed information about the antivirus vendor and version. The most common occurrences of exposed vendor information included Barracuda, IronPort, ProofPoint, Trend Micro IMSS, and MailScanner. These five vendors account for exposed information in 137 of the 483 bounced emails (28.4%). Examples of some of the leaked vendor and version information is listed in Table 1.

As the message bouncing technique may reveal a significant amount of detail about the antivirus engine, while maintaining a low risk of detection, it may be very attractive to attackers as an initial reconnaissance vector. However, if the bounced message does not contain enough information to identify the antivirus engine, the attacker may attempt the next technique.

4.2 Detection Probing

In order to evaluate how much information is exposed using the detection probing technique, we performed an active measurement experiment across a representative sample set of mail servers. However, to elicit the desired response messages, it is necessary to trigger the detection of the antivirus engine used by the mail server. Sending a real piece of malware to a number of mail servers across the Internet in hopes of triggering antivirus engines would certainly be an unsafe experimentation methodology.

Fortunately, there exists a test file called the EICAR Standard Anti-Virus Test File [3], which was designed to allow administrators to test whether their antivirus software is operating properly in a safe manner. The EICAR test file is a 68-byte ASCII string which is actually a valid COM executable. The full contents of the EICAR test file is as follows:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-
STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

The majority of vendors have included a signature for the EICAR test file in their antivirus products which is usually classified along the lines of "Eicar-Test-Signature". Therefore, we can send the EICAR test file as an attachment for the detection probing technique to elicit feedback in a safe manner from the mail server antivirus software without having to send a real malware sample.

To gain a representative sample of mail server responses, we sent a probe message with the EICAR test file attachment to approximately 1500 mail servers hosted by educational organizations. The recipient of the message was the "postmaster" alias to minimize invalid recipient errors as the "postmaster" alias is required by RFC 2821 [5] for SMTP email. Of the 1500 messages sent, only 1053 were accepted for delivery by the destination SMTP servers, 213 of which resulted in response messages being returned to the sender. Through manual inspection, 166 of these 213 responses contained information explicitly identifying the antivirus engine used by the mail server. This rep-

Identified Engine	Information Exposure
ProofPoint / F-Secure	X-Proofpoint-Virus: Found Infected X-Proofpoint-Virus-Version: vendor=fsecure engine=1.12.7160:2.4.4,1.2.40,4.0.166
IronPort / Sophos	X-IronPort-Anti-Spam-Filtered: true X-IronPort-AV: E=Sophos;i="4.27,718,1204520400"; v="EICAR-AV-Test'3'rd"; d="txt'?com'?scan'208";a="929062"
Trend Micro	X-TM-AS-Product-Ver: CSC-0-5.5.1026-15998 X-TM-AS-Result: No–10.22-4.50-31-1 Eicar_test_file was detected in the file (eicar.com)
McAfee	The WebShield(R) e500 Appliance discovered a virus in this file. The file was not cleaned and has been removed.
Barracuda / ClamAV	X-Barracuda-Virus-Scanned: by Barracuda Spam Firewall at x.edu

Table 1. A few examples of the information exposed through the bounced messages and detection probing techniques that can be used to identify the vendor or version of an antivirus engine.

resents an exposure rate of 77.9% of the response messages and 15.8% of the total delivered messages.

The information contained in the message responses exposed numerous protection vendors including Antigen, Barracuda (ClamAV), BorderWare, ClamAV, IronPort (Sophos), Kaspersky, Mirapoint, McAfee, MailFoundry, MailScanner (ClamAV), ProofPoint (McAfee, F-Secure), PureMessage (Sophos), SmoothZap (ClamAV), Sophos, SonicWall (McAfee, Kaspersky), Symantec, Trend Micro, and Untangle (ClamAV, Kaspersky). By far, the most frequent response was generated from Barracuda's email filtering appliance. Many of these mail security vendors actually employ antivirus engines from the top vendors in the backend (e.g., Barracuda's filtering appliance uses ClamAV), thereby inheriting the large number of vulnerabilities detailed in Section 2. Several examples of the vendor and version information exposure are detailed in Table 1. The detail of information exposed through this detection probing technique is generally more fine-grained than the message bouncing technique and could prove very valuable for an attacker attempting to exploit the antivirus engine.

Many of the mail server that did not send a response message to the probe email may have not deployed antivirus software, used antivirus that ignores the EICAR test, or simply routed postmaster messages to a null mailbox. Due to these limitations, the final result may be an underestimate of the actual set of affected mail servers but is the best approximation possible that can be performed in a safe, non-malicious manner.

4.3 Iterative Refinement

To determine whether the iterative refinement technique is effective in identifying a particular antivirus engine, we performed an experiment against a mail server hosted at the University of Michigan, of which we had no prior knowledge of its protection software. The University's mail server was configured to reject malicious email content and attachments during the SMTP conversation with a 554 error, providing an effective feedback mechanism. While a 554 SMTP error is a generic "Transaction failed" message, the error message returned by the mail server also included a link to the University's anti-spam and virus filtering information page, indicating that a malicious attachment had been blocked.

Using a collection of over 7200 malware samples from the Arbor Malware Library (AML) [7] and antivirus engines from ten vendors (Avast, AVG, BitDefender, ClamAV, F-Prot, F-Secure, Kaspersky, McAfee, Symantec, and Trend Micro), we were able to isolate malware samples that were detected by *only one* specific antivirus engine. While the selection of ten antivirus vendors certainly does not represent every possible antivirus product ever developed, it covers the most popular and commonly-deployed vendors.

By analyzing the malware dataset with each engine and correlating the results together, we were able to identify a substantial number of samples for each engine that were only detected by that engine and no others (Avast: 13, AVG: 6, BitDefender: 6, ClamAV: 23, F-Prot: 6, F-Secure: 8, Kaspersky: 14, McAfee: 4, Symantec: 8, and Trend Micro: 22).

Armed with these unique samples, we iterated through each vendor and sent one sample from each as an attachment to the target mail server. As seen in Table 2, the attachments that were only detected by ClamAV, Symantec, Kaspersky, and BitDefender were accepted for delivery. Besides the obvious security deficiencies of a mail server accepting actual malware samples, this process reveals that the mail server is likely not employing any of these four vendors. However, when we sent the malware sample only detected by McAfee, it was rejected by the mail server. This rejection event indicates with high probability that the mail server employs the McAfee antivirus engine. Additional samples only detected by McAfee were sent to increase the confidence of this result and the conclusion was later confirmed as correct by University administrators.

4.4 Mail Server Antivirus Survey

The last experiment performed was an informal survey distributed to the security contacts of numerous organizations which contained three simple questions:

1. *Do you employ antivirus software at your mail server to detect and block malicious email attachments?*
2. *If so, what antivirus vendor do you employ?*
3. *When was the last time you checked for security updates from the vendor to address vulnerabilities within the antivirus software?*

The majority of the survey participants indicated that their organization's mail server employs the ClamAV antivirus engine. Barracuda's spam and antivirus filtering appliance was also very popular, which uses ClamAV internally. Given that ClamAV has had the most vulnerabilities of any of the antivirus vendors as seen in Figure 1 back in Section 2, this result is of great concern. Also, the fact that the Barracuda appliance is one of the most verbose responders to the detection probing technique represents a serious threat to numerous organizations.

In response to the third question regarding security updates to the antivirus software itself, a significant number of survey participants indicated that they frequently update their *signatures*, rather than specifying how often they update the engine itself. While some of the participants may have misinterpreted the question, the lack of understanding exemplifies how security administrators may be unaware of the serious vulnerabilities in their antivirus products and the associated risk in a mail server environment.

5 Addressing the Threat

The threat of an attacker using leaked information from a mail server to specifically target a particular antivirus engine is very real threat that must be addressed to harden networks against such an attack. In this section, we explore two approaches to addressing this threat: one which is a short-term approach to reduce the exposure of a mail server; and one to address the impact of the underlying vulnerabilities within the antivirus software.

5.1 Reducing Information Leakage

One approach to reducing the threat of an attacker remotely identifying the antivirus software used by a mail server is to reduce or eliminate the information sent by the mail server. For example, bounced messages from invalid recipient mailboxes or detected viruses can be scrubbed clean of identifying information. While security by obscurity is not a wise protection mechanism to solely rely on, it can add an additional hurdle for an attacker to leap. Of course, eliminating certain feedback mechanisms from the mail server is a delicate balance of usability and security. For example, preventing the third technique from being effective when the attacker possesses a feedback mechanism such as a user's automated vacation message may be more difficult and require additional detection methods and alerting thresholds.

AV Engine	AML Sample	Mail Server Response
ClamAV	3507c29a209d9aaf49dbd1167d633aeb	sent (250 (4866BC9D.AC670.1261): Accepted)
Symantec	04fd46c7aec6d9803a819e877e42b504	sent (250 (4865BA6A.AA080.5714): Accepted)
Kaspersky	841adece7e5317e665aca9cce308fabb	sent (250 (4865BA8E.13514.12023): Accepted)
BitDefender	78bcd78b13ccf37554796cfb100becc2	sent (250 (4866BC33.81961.19326): Accepted)
McAfee	1f7beae151277f63dbe6a89be9ebc96e	bounced (554 Transaction failed)

Table 2. Responses from the University of Michigan's mail server when utilizing iterative refinement. The AML sample only detected by McAfee is rejected, confirming McAfee as the deployed antivirus.

5.2 Isolation Mechanisms

While reducing the amount of information exposed by a mail server and its accompanying antivirus engine is a significant improvement, it does not address the underlying problem of antivirus vulnerabilities. On the other hand, while it would be optimal to deploy antivirus software which is free of security vulnerabilities, it is unrealistic to assume that a software product of such complexity could achieve that goal. Therefore, a desirable goal would be to mitigate the *impact* of the vulnerabilities present in the antivirus engines.

The impact of running a vulnerable antivirus engine within a mail server can be significantly reduced or eliminated by introducing a layer of isolation between the antivirus engine and the mail server. The use of isolation to mitigate the vulnerabilities of antivirus software has been explored in previous work. In [9], a mail server frontend (via a milter plugin) is used to transfer mail attachments to a network service for analysis as opposed to analyzing files on the mail server itself. Virtualization is employed to isolate the vulnerable antivirus engines from the mail server and the rest of the system. If an antivirus engine is successfully exploited, the virtualized container hosting that antivirus engine could simply be disposed of and reverted to a clean snapshot. As [9] provides a controlled environment for the backend engines, detection of the exploitation of an antivirus engine would be a feasible procedure (e.g., attempting to spawn an unrecognized process, initiating socket communications with external parties to leak email data, or even executing unusual code paths). Providing this layer of isolation strikes a balance between the simple approach of eliminating the exposure of information from the mail server and the difficult problem of eliminating all vulnerabilities within existing antivirus software.

6 Conclusion

Vulnerabilities within the antivirus software of mail servers represent a serious threat vector that has not received a sufficient amount of attention or exploration. Towards understanding how a sophisticated attacker could perform such an attack while minimizing detection risk, we introduced and evaluated three information exposure techniques that may allow a malicious party to remotely identify an antivirus engine deployed by a mail server. To address this threat vector, we discussed potential solutions that may reduce the exposure of information or eliminate the impact of vulnerabilities in the underlying antivirus software.

References

[1] Adobe Systems Incorporated. Apsb08-15 adobe acrobat vulnerability. http://www.adobe.com/support/security/bulletins/apsb08-15.html, 2008.

[2] Barracuda Networks. Barracuda spam firewall. http://www.barracudanetworks.com, 2008.

[3] EICAR e.V. Eicar antivirus test file. http://www.eicar.org/anti_virus_test_file.htm, 2006.

[4] IronPort Systems. Ironport email security appliances. http://www.ironport.com/products/email_security_appliances.html, 2008.

[5] J. Klensin. Rfc 2821: Simple mail transfer protocol. http://www.ietf.org/rfc/rfc2821.txt, 2001.

[6] Microsoft Corporation. Microsoft security bulletin ms08-026. http://www.microsoft.com/technet/security/bulletin/MS08-026.mspx, 2008.

[7] Arbor Networks. Arbor malware library (AML). http://www.arbornetworks.com, 2007.

[8] NIST/DHS/US-CERT. National vulnerability database. http://nvd.nist.gov/, 2007.

[9] Jon Oberheide, Evan Cooke, and Farnam Jahanian. Cloudav: N-version antivirus in the network cloud. In *Proceedings of the 17th USENIX Security Symposium*, July 2008.