Proceedings of the 13th Colloquium for Information Systems Security Education
University of Alaska, Fairbanks
Seattle, WA June 1 - 3, 2009

AEGIS: A Pedagogical Tool for Vulnerability and Patch Management

Madhusudhanan Chandrasekaran, Shambhu Upadhyaya, Newton Campbell Jr. and Horus Alkebulan Jr.

Abstract – In this paper, we present a new pedagogical tool called AEGIS for educating college students about the importance of patch and vulnerability management. The tool is designed and developed in a bottom-up fashion by a group of eight students in accordance with the NIST patch and vulnerability management guidelines. Experience gained while devising various subcomponents of AEGIS could provide students with the much needed hands-on practical training in security and system administration curricula. Lastly, the possibility of using AEGIS as an educational tool to teach and assist day-to-day users with patch and vulnerability management is explored.

Index terms – Patch Management, Security Education, Vulnerability Assessment

I. INTRODUCTION

Recently, significant effort has been expended on educating college-level students on the concerns of security, privacy, integrity and reliability. The training is imparted through academic courses that focus on application security, network security and cryptography. Regrettably, most computer security related courses taught in schools have been criticized as being too theoretical bearing little or no emphasis on real world security problems. As a result, students who enroll in these courses often fail to measure up to the industry workforce standards. There are many specialized programs proposed by both government and academia as a part of information assurance curriculum, which specifically cater to the needs of students who want to pursue a career in information security. Even though these specialized courses are rigorous in their outlook offering hands-on training, their outreach is restricted to the academic setting only. To tackle this limitation, a significant number of proposals have been made by academicians and IT professionals to develop a balanced industry-supported computer security curriculum [1] [2]. The common consensus is that, besides being treated as a monolithic entity, information security should be also made an integral part of various disciplines of computer science ranging from software engineering to theoretical computer science. Such a multidisciplinary approach would enable students pursuing a computer science degree to be exposed to fundamentals of information security at a more pragmatic level, thus preparing them to perform well outside a university environment. Metaphorically speaking, it is important that a software programmer possesses knowledge about buffer overflow attacks to guarantee security and reliability of the underlying software component. A database administrator should ensure that data is not leaked out to unauthorized personnel via SQL injection attacks. Similarly, a network administrator should be able to perform ingress and egress traffic filtering to identify anomalous patterns. While training students to build and monitor secure systems is vital for their success as software engineers, an equally crucial, yet daunting task is training them to manage internal computing resources so that their security is not compromised. Knowing how to safeguard computing resources is a skill that needs to be acquired by every computer user.

An initial step geared towards protecting enterprise-level systems from external attacks involves deployment of intrusion prevention systems (IPS) along the network periphery. Intrusion prevention systems, typically, include firewall, vulnerability analyzers, anti-virus and other software that curtail external attacks from penetrating into the internal systems. These systems seldom require manual intervention, and can operate in an automated fashion. Despite serving as a first line of defense, IPS offer little protection from exploits that target vulnerabilities present in the deployed software. Typically, these vulnerabilities are managed by applying vendor specified patches, which remediate the underlying programming flaw in the software. While timely dissemination and application of patches limit the exposure of systems to such vulnerabilities, the process of patch and vulnerability management is not always straightforward. First, the task of identifying vulnerabilities pertinent to a given network/system configuration is a tedious one requiring the expertise of trained professionals. Second, as patches are occasionally released as security recommendations, qualified personnel are required to interpret and enforce them. Finally, since patches do not always work and have undesirable side-effects resulting in system downtime and service

Department of Computer Science and Engineering,
The State University of New York at Buffalo
Buffalo, NY 14260

disruption, the decision to apply patches must be carefully reviewed by the security personnel [3]. Therefore, it is imperative that personnel responsible for patch management should have adequate understanding of these related issues.

The task of patch management is usually thrust on system administrators who have to rely on commercial-off-the-shelf (COTS) tools such as security scanners and vulnerability assessment tools for periodically scanning the network and systems for software that do not conform to the security recommendations. These tools just act as an interface to identify unpatched software. However, given the heterogeneous nature of software configurations, it is hard even for experienced system administrators to keep abreast of all the new vulnerability reports and their patches. The process of assimilating vulnerabilities pertinent to a given network configuration is a hard problem. As large number of vulnerabilities is discovered on a daily basis, it may require manually sifting through the vulnerability reports published across numerous external bug tracking repositories to spot the relevant ones. Even worse, as the schematic structure of the reports is not standardized, it takes a significant effort to semantically parse and digest the information present in them. However, one advantage of early identification is that preemptive, just-in-time (JIT) solutions may be generated from the information contained in the reports. For example, at the time of disclosing the recent Microsoft Windows Metafile Format (WMF) vulnerability, SANS provided a set of SNORT rules to temporarily mitigate the attack. Although, in this case the corresponding patch turn-around time was relatively quick (less than a week), nevertheless these signatures could have been used for protecting against the attacks, especially during the vulnerable gap.

Despite the industry appeal, adequate training in patch and vulnerability management is not provided as a part of academic curriculum. National Institute of Standard and Technology (NIST) recommends a set of guidelines to be adopted by each organization so that early identification and distribution of patches can be facilitated [4]. Each organization is also advised to form a "patch and vulnerability group" (PVG) consisting of set of trained individuals to work in conjunction with system administrators for undertaking such remedial efforts. From this angle, we propose a tool named AEGIS (which means protection), to educate students about the effects of patch management and to shield internal resources from zero-day attacks. Zero-day attacks are exploits that target patchless vulnerabilities, especially in the time frame between their exposure and when vendor specified patches are available. AEGIS serves as a lightweight defense mechanism to assess and protect the network from recently published vulnerabilities. It consists of a local database from two different vulnerability publishing sources, viz., Secunia and NVD. The local database is periodically synchronized with the external repositories to keep it up-to-date. To reduce the assessment overhead, only those vulnerabilities that are pertinent to a given network configuration are considered. For this purpose, at the time of deployment, the configuration details of all services running on different hosts in a network are encapsulated through a Service Definition Language (SDL). Furthermore, an extensible defense-oriented representation schema (EDORS) is proposed to represent vulnerabilities in a machine-oriented format. This schema is then used by the policy engine to generate detection rules that can be accommodated in any firewall or network intrusion detection system (NIDS) capable of performing deep-packet inspection. The final decision of applying rules can be made from the recommendations provided in the EDORS report. Adequate care is taken that such rules do not pose any threat to the overall sanity of the system by causing unwanted side-effects.

AEGIS was initiated as educational research project from the support received through a Department-of-Defense (DOD) grant. The general recommendations listed in the NIST's "Creating Patch and Vulnerability Management Program" document are used as a guide for building AEGIS. The actual implementation was carried out by a group of eight students as a part of their supervised research study. Both graduate and undergraduate students who were a part of computer security research group were actively involved during the brainstorming and implementation sessions. Finally, a demo of AEGIS was presented to the general public during computer science and engineering open house for the purpose of educating them about the importance of applying patches, and also getting their feedback. In essence, this paper highlights students' experiences with designing and implementing AEGIS, and the educational value brought forth by it.

The remainder of this paper is organized as follows: Section 2 discusses the design goals of AEGIS. The various issues that may arise during its implementation and their workarounds are also provided in this section. Section 3 provides the details on customizing AEGIS according to the patch and vulnerability management guidelines provided by NIST. Section 4 elaborates on AEGIS' value as an educational tool. Concluding remarks are provided in Section 5.

II. Design Goals of Aegis

From a technical perspective, the following three main design goals must be incorporated into AEGIS.

A. Minimize the vulnerability aggregation overhead – In order to keep the overhead minimum, the aggregator should download only the relevant vulnerabilities that have been recently added or updated in the repositories in a machine-oriented format. While this is trivial with the National Vulnerability Database (NVD), which lists all

the newly added and modified in a XML format, most other sources maintain the information across HTML pages that are hard to process. Even though there exist tools like MBSA Microsoft, Nessus Deraison and GFI LANGuard Network Security Scanner, which perform vulnerability assessment, they, however, are used to only check for missing patches, service packs and vulnerable software, and are oblivious to vulnerabilities for which vendor specified patches are not available. Also, since the source of our vulnerability data are spread across the Internet, the system has to be extensible to plug in sources over time.

B. Generation of organizational policy based IDS signatures – The information presented in the vulnerability reports should be coupled with the organizational policy to generate appropriate defense signatures, which act as temporary solutions that help protect the system until the vendor specified patches are available. Moreover, the generated rules should be able to interoperate with the existing NIDS/firewall, and be easily revocable when the corresponding patches are released.

C. Flexible deployment requirements – AEGIS should be able to run across multiple operating systems (i.e., Windows, Linux, others). This along with the goal of being able to easily customize the code drove the decision to use Java as the programming language. Snort was chosen because it also met the cross platform design goal, but AEGIS can be enhanced to use other NIDS platforms as long as it allows rules to be updated programmatically.

III. Building AEGIS in Accordance with NIST Patch and Vulnerability Management Guidelines

AEGIS is implemented on a single host, which acts as a central agent for monitoring other end hosts in the network as shown in Figure 1. Such a thick-server thin-client model helps in keeping all the complex processing in one location. Also, to make the implementation of AEGIS complete and rigorous, the patch and vulnerability management guidelines stipulated by NIST are accommodated during every stage of its design and development. Building a system according to set guidelines helps the students to understand clearly the process involved, while also providing them with hands-on industry level experience.

"Guideline 1: Create a System Inventory."

In order to determine relevant vulnerabilities, it is important to know the services running in every part of the network. Essentially, the details about the network services and their software versions need to be captured. For the purpose of keeping the design simple and also to be less intrusive, the burden of reporting the configuration details to the central agent is delegated to the end-host itself. Periodically, every end-host encapsulates all the running services using a service definition language (SDL) and presents the gathered information to the scanner. The scanner then prepares an aggregate configuration report from the information gathered from end hosts. A local table is also maintained that maps the configuration information to the hostname or the IP address of the end host.The SDL consists of three classes of information: (i) appInfo – It defines the application names and present version of that particular application along with the underlying operating system details. Each entry also contains the corresponding IP address or hostname to identify the host. (ii) servInfo – It lists the services that are running on the system, their underlying protocol base, i.e., TCP, UDP or ICMP and the corresponding ports on which it is listening or sending. This feature is required to bind the services running on different hosts to the appropriate ports and protocol bases. The operating system information and the services can be sniffed also using p0f and libpcap programs. (iii) criticalityInfo – It denotes the measure of how important the service is to the organization. Each service is assigned a number between 1 and 5, where 5 indicates that the service is extremely critical and should not be taken down without the administrator's consent. On the other hand, 1 means that the service is less important and all ingress traffic flowing towards its port can be dropped without any side effects.

"*Guideline 2: Monitor for Vulnerabilities, Remediations, and Threats…"*

The aggregator and synchronization agent is responsible for keeping the local bug repository up-to-date with all the vulnerability disclosure reports from different bug tracking repositories. In order to accomplish this, the aggregator periodically connects to the repositories to assimilate added or modified vulnerability reports. An important factor to consider is that not all bug repositories have entries in machine oriented format (i.e., as XML or RSS feeds). HTML document object model (DOM) parsers are also built-in so that they can be customized to the format specific to a given repository. The parser operates in a stateful manner, i.e., it only retrieves newly added or updated reports. The scanning and aggregation time interval is dependent on the criticality of the system under consideration and update time of the repository.

"*Guideline 3: Prioritize Vulnerability Remediation…"*

The vulnerability remediation is prioritized based on two factors: (i) criticality of the service; and (ii) criticality of the vulnerability. Usually, the criticality of the service can be pre-defined at the time of its deployment. Criticality of the vulnerability is decided based on the impact factor

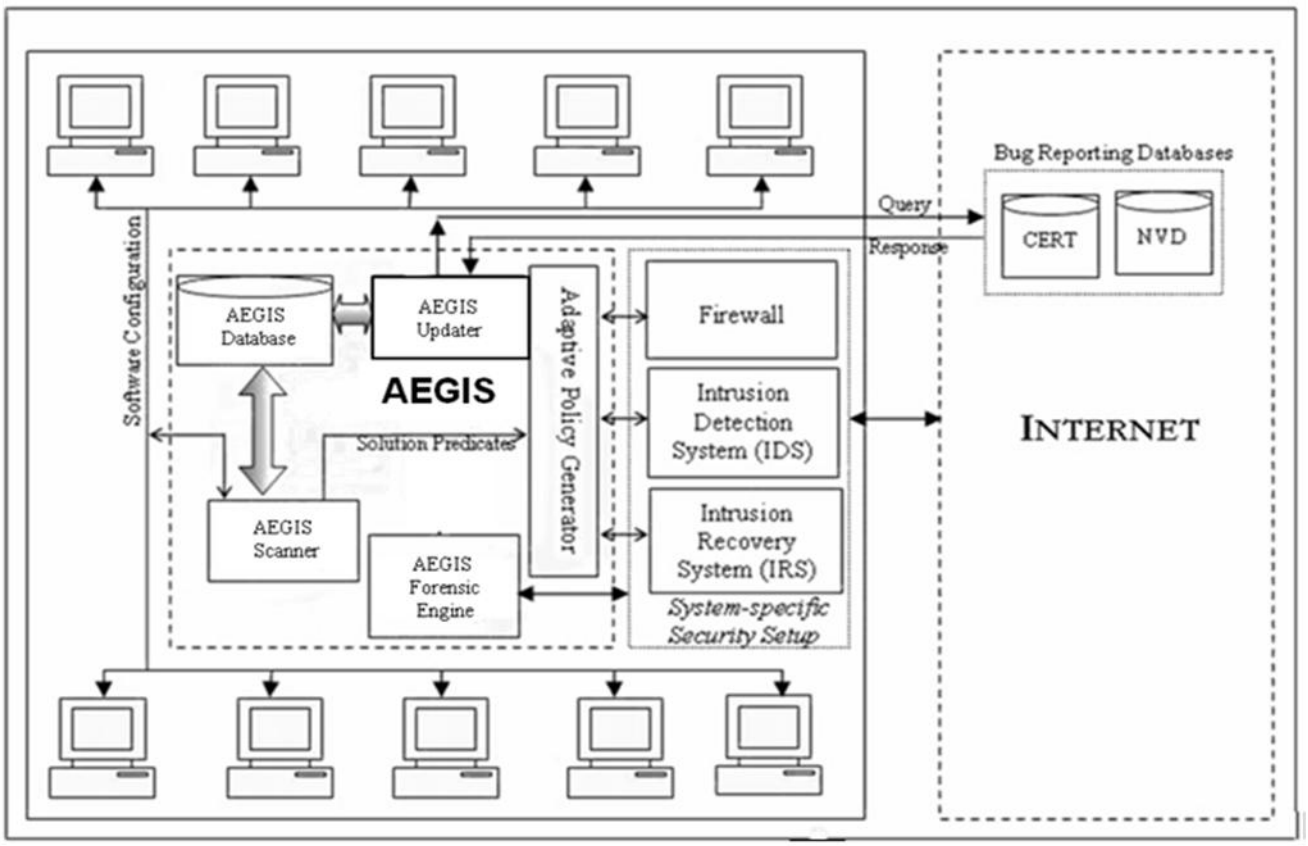

Figure 1: Architecture Block Diagram of AEGIS

given in the vulnerability report, and how it may affect the functioning of the organization. This is needed to be done by qualified PVG personnel only. Also, there may be unforeseen side-effects when trying to adopt remedial actions. Thus, the network configuration must be replicated, at least at a miniature level, using virtualization and the patches must be applied on it.

"*Guideline 4: Create an Organization Specific Remediation Database…*"

Existing bug tracking repositories employ proprietary formats that are practically disjoint. The lack of commonality among them makes it difficult to automatically extract the contained information and generate remedial actions. For example, linux-ftpd-ssl buffer overflow vulnerability (CVE-2005-3524) information, represented across two popular vulnerability databases varies significantly, especially with regard to the information about the vulnerable operating system platforms. To overcome these limitations, we propose Extensive Defense Oriented Representation Schema (EDORS), which precisely represent the vulnerabilities aggregated from multiple sources.

The three essential components of EDORS are: (i) Vulnerability Preconditions – It identifies the underlying vulnerable software and the corresponding setup needed for exploiting it. This information is typically used by the scanner to determine if such vulnerability exists in given network configuration. Cross references to other bug tracking repositories are tracked to see if further information about the vulnerability is available; (ii) Impact Details – For the purpose of adopting appropriate defense measures, it is important to assess the security impact of all vulnerabilities. This also helps in understanding if the proposed defense action violates an organization's policy; and (iii) Remedial Actions – Flexibility must be incorporated in the vulnerability representation so that at the time of reporting, any inferred defense predicate can be easily included in the existing defense solutions. These remedial actions can be specified by the security practitioner who maintains the external bug tracking repository or a member of the PVG. We have used XML to represent and specify the vulnerabilities according to the proposed EDORS format. XML provides a structured format to represent the vulnerability reports. XML also provides the necessary flexibility, interoperability and portability needed in the efficient dissemination of such information. Also, availability of schema independent XML parsers makes it feasible for customizable self-documenting representation formats. The example of linux-ftpd-ssl vulnerability in EDORS format is given in Figure 2.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Format>
   <Vul>
   <ID>CVE-2005-3524</ID>
   <Reference>BugTraq:15343</Reference>
   <Reference>FULLDISC:20051105</Reference>
   <Reference>Secunia:SA17465</Reference>
   <Advisory>
   <Date_Created>11/07/2005 12:00AM</Date_Created>
   <Date_Modified>11/15/2005 04:32PM</Date_Modified>
   <Patch-Available>NO</Patch-Available>
   <Date>11/15/2005</Date>
   </Advisory>
   <OS>
         <Kernel>Linux 2.4.18-14</Kernel>
         <Distribution>Redhat 8.0</Distribution>
   </OS>
   <Package>
         <PackageName>linux-ftpd-ssl</PackageName>
         <Version>0.17</Version>
   </Package>
   <Vulnerability>
         <Range>Remote</Range>
         <Type>Privilege Escalation</Type>
         <Severity>High</Severity>
   </Vulnerability>
   <Message>FTPd Attack possible</Message>
   <Remedy>
         <Drop>1</Drop>
         <Log>1</Log>
         <Direction>IN</Direction>
         <Size>100</Size>
         <Content-String>AUTH-SSL</Content-String>
   </Remedy>
   <Signature> </Signature>
   </Vul>
</Format>
```

Figure 2: An Example Vulnerability Report in EDORS Format

"*Guideline 5: Conduct Generic Testing on Remediations…*"

As mentioned earlier, the remedial actions given in the EDORS report are first tested on a virtualization environment to see if there are any side-effects. For this purpose, all services and software reported in SDL are deployed in a test or mockup environment. Also, care is taken to ensure that testing is done only if and only when the vulnerability preconditions are met. The downloaded vulnerability reports are tested against the vendor-provided authenticity verification schemes such as PGP signatures and cryptographic schemes.

"*Guideline 6: Deploy Vulnerability Remediations… & Guideline 8: Perform Automated Deployment of Patches…*"

The adaptive policy generator acts on the input provided by the AEGIS scanner to generate remedial actions. If vendor specified patch is available for a vulnerability (specified by <Patch-Available>…</Patch-Available> tag in the EDORS report), then it is downloaded and the PVG is informed about its presence. Options can also be set in the policy generator to apply patches in an automated fashion. In case where the vendor specified patch is not available, the decision to adopt any defensive action is solely left to the discretion of PVG. PVG employs the information provided in the remediation portion of the EDORS report to generate IDS rules that act just-in-time (JIT) solution to safeguard the system until its patch becomes available. This also protects the system in the zero-day gap. The effects of these rules may vary from simple blocking to complex deep-packet probing of malicious packets destined towards a vulnerable host. The potential set of actions that could be taken on the incoming packets are allow, alert, log, or drop.

"*Guideline 7: Distribute Vulnerability and Remediation Information to Local Administrators…"*

All actions taken by the PVG are logged carefully for auditing purposes. Support to e-mail system administrators about the vulnerable software and the corresponding remediation actions is provided. It is ideal that system administrators be notified or kept in the loop during the patching process. Also, once when the vendor specified patch becomes available, the corresponding JIT solution, if any, should be revoked with system administrator's consent before application of the patch.

"Guideline 9: Configure Automatic Update of Applications Whenever Possible and Appropriate…"

Again, as with the patches, this recommendation is left to the discretion of PVG and system administrators. The automatic update is not made as part of the AEGIS as it is provided by the underlying operating system or by package managing utilities. The main reason for this is because information about software updates is not available in bug tracking repositories.

"Guideline 10: Verify Vulnerability Remediation Through Network and Host Vulnerability Scanning…"

AEGIS scanner is run again after the application of patches/updates to check if it displays the current information. This information is presented in the AEGIS front end for manual inspection.

"Guideline 11: Vulnerability Remediation Training…"
The process of actively involving students in the various design and implementation phases of AEGIS provides them with adequate hands-on training.

IV. Experiences and Lessons Gained while Implementing AEGIS

The experiences and lessons gained while implementing AEGIS can be summarized in two parts: a) Knowledge gained when implementing AEGIS; b) Knowledge gained by end-users while using AEGIS as a patch and vulnerability management toolkit. In this section, we summarize these two aspects to get a better insight on AEGIS' educational value.

A. Knowledge gained when implementing AEGIS

The core of AEGIS backend and its GUI frontend were implemented using JAVA Standard Edition 1.6. The involved students were first given a brief primer on JAVA. Since most of the students were graduate students or were in the senior year, fortunately, they had prior experience with JAVA. Appropriate design patterns were used while building the different software components of AEGIS. An internal PVG was formed with theses students as members. In order to get a list of packages installed in the network, customized scripts were written in PERL. For the sake of simplicity, UBUNTU Linux version 8.10 OS was installed in each of the end host. A set of four hosts each having different software configuration was used in the experimentation. The same setup was also replicated in four virtual PCs using VMWARE software and Norton Ghost. Virtual PCs serve as test machines to try out just-in-time solutions and unstable patches. The complete set of packages installed in each end host was obtained through the apt-get program. The set of services running on them was obtained by parsing the xinetd configuration file. In this process, students were exposed to the working of package repositories in UNIX environment. Moreover, they were also taught on how to examine services running in a system and turn off unwanted ones. SDL templates were first constructed from the information gathered by the scripts and forwarded to AEGIS as XML files so that they can be displayed in the front end GUI.

The parsing of external bug repositories was done using XML and HTML DOM Parser. JTidy, a Java port of HTML Tidy which does HTML syntax checking and pretty printing was used for the purpose. The retrieval of information from external repositories (NVD, CVE and BugTraq) is done primarily with XQuery and XPath. This helped students understand about latest XML and DOM technologies. Lessons on how to write a user agent to fetch Webpages using JAVA were also obtained. The fetched information was converted into EDORS format. The remediation section of EDORS is either completed using the information given in the reports or manually after consultation with other group members. If a patch is available, then it was tried out in virtual PCs and the side effects are analyzed. The network intrusion detection system (NIDS) selected was SNORT due to the ease of creating IDS rules and the fact that it is open source and cross platform. So adequate training was provided to the students on how to implement and maintain SNORT rules. Open source GUI front ends to SNORT also help in expediting the learning process. For the sake of testing AEGIS, an IMAP related vulnerability (CVE-2008-5006) was chosen. The vulnerable e-mail client was run on each of the end hosts. We generated and sent test mails that

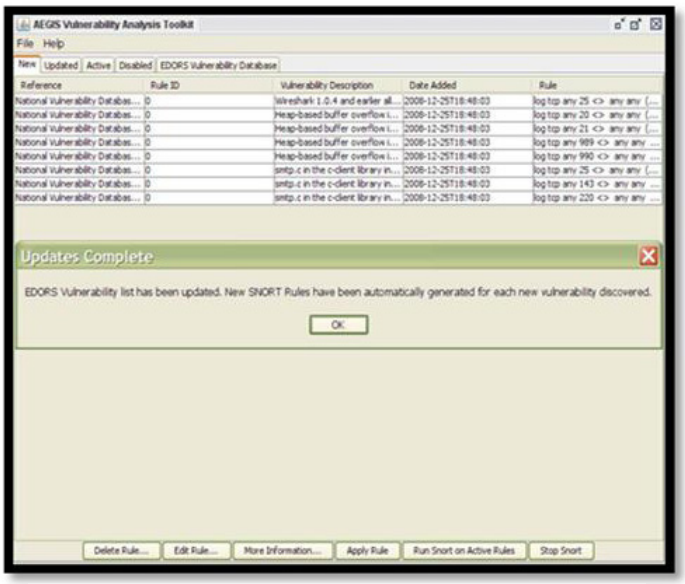

Figure 3: Screenshot of Vulnerability Checking

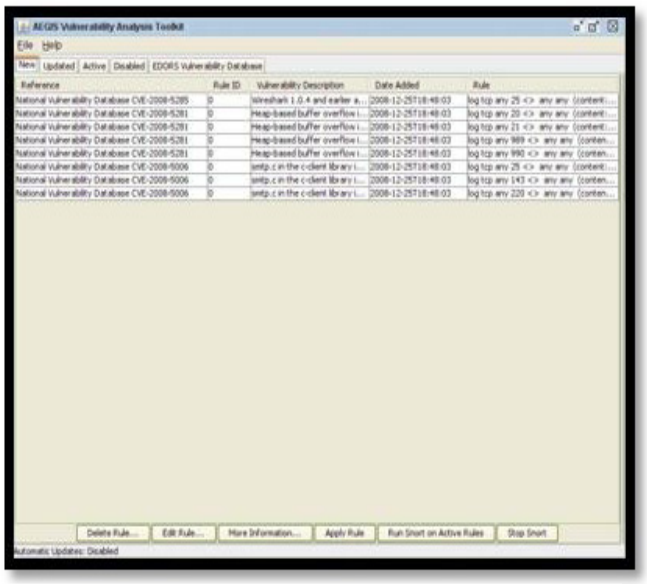

Figure 4: Screenshot Listing IDS Rules Generated from Remediation Tag in EDORS Report

contained packet content related to the vulnerabilities. Once AEGIS has created the Snort rules, the NIDS successfully logged the offending traffic. We used a similar test for the FTP based vulnerability (CVE-2008-5281). Here we had AEGIS create and apply the applicable Snort rule. Next, using an FTP client from the same computer, we downloaded a text file to generate traffic. Again, the NIDS successfully alerted when the adverse packet was scanned.

In summary, students were taught not just the security aspect of patch management, but also the latest software tools required to build an automated vulnerability aggregation and patch management solution.

B. AEGIS as an education toolkit for end users

A tutorial walkthrough for the AEGIS toolkit is provided below for downloading new vulnerabilities and creating Snort rules.

1) When the application first starts, it will perform a check for new vulnerabilities. This allows the end users to see all newly published vulnerabilities relevant to their system setup (see Figure 3).

2) For any new vulnerabilities found that match currently running services, patches/IDS if available are downloaded, but are not applied (see Figure 4).

4) Once the update download is complete, a new tab will show vulnerabilities for which patches and remediation rules are not available.

5) To begin the process of activating a rule, the corresponding rule should be selected and the Apply Rule button needs to be clicked by the user.

6) An active tab is provided to view the rules that are implemented in the IDS.

7) To change a suggested rule or incorporate a new rule, Edit Rule button can be used as shown in Figure 5.

Figure 5: Screenshot of Edit Rule

8) Once the rules can be exported to be applied in the production system by saving it as XML files. The patches and rules needed are also logged in and an e-mail is automatically sent to the administrator.

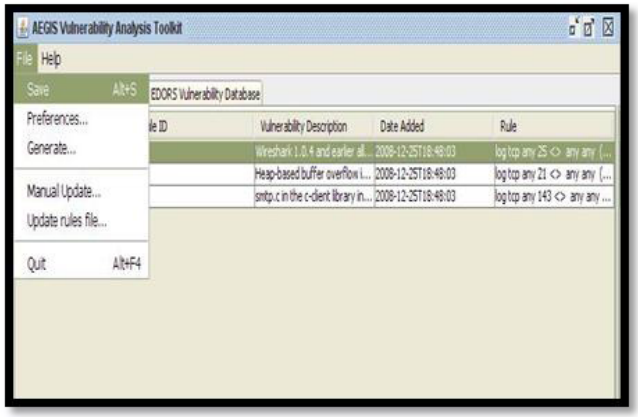

Figure 6: Exporting of Rules into Production System

9) Lastly, click the Run Snort on Active Rules to update your Snort configuration.

Figure 7: Screenshot Showing Snort after Applying the Rules in a Given Setup

V. Conclusion and Future Work

AEGIS is implemented as a patch and vulnerability management tool as per the NIST guidelines. A group of eight students were involved in its design and development. Apart from reinforcing security awareness, the exercise provided them with the much needed experience in patch and vulnerability management that is in alignment with the current industry demand. Also, the tool can be used by the interested end users to practically manage their system or network configuration. As a future work, we plan to extend AEGIS by making it an open source project so that other educators can use the tool to provide hands-on for their students in their IA curricula.

VI. Acknowledgements

This research was supported in part by DoD Grant No. H98230-07-1-0243. Authors would like to thank all the students participated in this project.

VII. REFERENCES

[1] H. Armstrong and C. Armstrong, "*The Role of Information Security Industry Training and Accreditation in Tertiary Education*", IFIP International Federation for Information Processing, 2007

[2] A. Yasinsac, "*Information Security Curricula in Computer Science Departments: Theory and Practice*," The George Washington University Journal of Information Security, Volume 1, Number 2, 2002

[3] S. Beattie, S. Arnold, C. Cowan, P. Wagle, and Chris Wright**,** *"Timing the Application of Security Patches for Optimal Uptime".* In Proceedings of the 16th USENIX Conference on System Administration (Philadelphia, PA, November 03 - 08, 2002). System Administration Conference. USENIX Association, Berkeley, CA, 233-242.

[4] P. *Mell, T. Bergeron, and D. Henning, "Creating a Patch and Vulnerability Management Program",* Recommendations of the National Institute of Standards and Technology (NIST), http://csrc.nist.gov/publications/nistpubs/800-40-Ver2/SP800-40v2.pdf