Attack Traffic Libraries for Testing and Teaching Intrusion Detection Systems

Jeffery Burroughs
Department of Computer Science
University of Wyoming
Laramie, WY 82071, USA
jburroug@uwyo.edu

Dr. Patrick Engebretson
College of Business & Information Systems
Dakota State University
Madison, SD 57042, USA
pat.engebretson@dsu.edu

Dr. JoshuaPauli
College of Business & Information Systems
Dakota State University
Madison, SD 57042, USA
josh.pauli@dsu.

ABSTRACT

Various attack traffic libraries have been available for quite some time. However, a majority of these libraries contain additional miscellaneous or highly unorganized network traffic. Our research aims at creating a framework for building small, clean, reusable, attack specific network captures. These captures can be used for teaching intrusion detection system monitoring, access control list creation, device configuration, and testing. The lack of organized individual attack captures makes finding specific examples of attack traffic difficult. Our work takes the concept of attack traffic libraries and builds on it. The PCAP Attack Library (PAL) that we have created is simple to use and easily expandable. We captured individual examples of attack traffic and classified each attack according to the Common Attack Pattern Enumeration and Classification (CAPEC) library. Each of our cataloged attacks include; a re-playable traffic file, a corresponding CAPEC attack identification number for obtaining further attack details, and a working Snort Intrusion Detection System rule (SIDSr) which can be used to detect the specific attack. A framework for cataloging and extending the PCACP Attack Library is also presented. This work is equally valuable to instructors and professionals responsible for maintaining an intrusion detection system. Educators and professionals now have access to specific replayable attack traffic without needing the original tools or knowledge required to create the attack.

Keywords: Network Attack Traffic, Attack Library, Snort, Intrusion Detection, PCAP, CAPEC, Ethical Hacking, Network Security, Education

1. Introduction

Testing Intrusion Detection Systems (IDS) to ensure the most malicious attacks are detected is a cornerstone of these systems, but there is no standardized method to execute these tests. Running live exploitation is not always a viable option – especially when the rule set isn't finalized, and clients are often nervous about the use of "hacker tools" on their networks. Furthermore, educators struggle to teach IDS concepts as a standalone principle without teaching attack methodologies at the same time. We introduce the PCAP Attack Library (PAL) to help solve these problems. The PCAP Attack Library consists of individual pre-captured attack files that can be easily replayed for IDS testing and education. This library is completely preassembled, clean, and extendable to include further additions of attacks. Our initial library is created from the findings in the Common Attack Pattern Enumeration Classification (CAPEC) from the Department of Homeland Security [1]. The PAL can be utilized and replayed by any tool capable of reading and sending .pcap files. Tools such as TCP Replay allow users to send attacks to a specific target or broadcast to an entire subnet of machines. Additional features include the ability to select individual or multiple simultaneous attacks as well as provide layer 2 and 3 packet level manipulation. We conclude by presenting a methodology for capturing attacks and adding them to the public library.

An attack traffic library consists of a collection of network traffic capture files. Each capture file consists of network traffic containing packets of data, organized chronologically in the order that the packets were produced at the time of capture. Attack traffic libraries entries are usually in the form of a raw TCPdump. The organization of an attack library can vary from library to library [2][3]. However, most libraries are organized by the type of attack that was performed or by specific parts of an attack.

In order to provide users with the most up-to-date information we include attack examples that appear to be most relevant. This relevance will be determined in part from data provided by the Symantec Corporation [4].

In general our main goal is to provide users with an easy to use, searchable attack traffic library. This goal has been met in that our library currently includes nine attack examples. Each of these entries are searchable by CAPEC definition ID and include captured, sanitized network traffic files as well as a corresponding SIDSr to detect the attacks contained in the traffic files.

2. Related Work

The Computer Science Department at Indiana University of Pennsylvania relies heavily on the hands on experience [5]. Utilizing downloads from www.cert.org and www.insecure.org they integrate several common security tools which are free or open source. These tools include Ethereal, Tripwire, Nmap, Nessus, logsentry, logwatch, GnuPG, Nutcracker, John the Ripper, and Crack. The tools are deployed on Linux. Student challenges include keeping the systems updated, appropriate tool selection, and the installation process of the tools [5]. This example highlights the importance of providing students with a hands-on experience to reinforce theoretical material.

In order to reinforce learning the computer Science Department at Texas A&M University also makes use of hands on exercises. The goal of this program is to teach students concepts of computer security [6]. Students are divided into two teams and work as a member of either a black or gold team. The goal of the black team is to break into other team's computers. The goal of the gold team is to defend their network. Based on four years of this structure it has been determined that "persistent cooperative groups and active learning are effective approaches for teaching network security and are preferred over a lecture-based course." [6] Lab exercises reinforce concepts introduced in lecture as well as help with effective thesis research. There are many benefits to implementing hands-on learning including the ability to create contrived situations, the ability to place administrative limitations on the systems, and the ability for black team members to utilize tools or platforms which are not allowed on the school network.

The utilization of scenario driven problems which force students to think outside of the box and provide non-traditional formats of teaching can lead to higher student learning and satisfaction levels [7].

The use of hands on labs can provide benefits for testing student comprehension. Rather than using traditional paper based tests, students are asked to reproduce an activity based on a previous lab [8]. Labs can be configured to produce competency or exploratory labs. Competency labs are goal oriented and provide the student with instructions but are not step-by-step. When the student feels they have mastered the material they are required to answer a series of questions based on the lab exercise. Exploratory labs require the students to create their own experiments based on a set of parameters and questions. When implemented together these types of labs help students develop a solid area-specific knowledge base and increase their ability to think independently [8].

It is not uncommon to find students who excel at taking exams but are labeled by employers as lacking real-world experience and skills [9]. Students need practical experience to augment their theoretical skill sets [10]. The incorporation and utilization of hands on material provides benefits to both students and employers. Students benefit from the real-world experience which can supplement traditional theory based lectures. Employers benefit because students have a deeper understanding and more experience utilizing in their networking and security skills [11].

3. Building the PCAP Attack Library

In order to catalog the data that results from the capture of attack traffic, some basic network infrastructure must be assembled and tested. Recall that one goal of the PCAP Attack Library is to produce a simple, unpolluted example of the given attack. To accomplish this, our attack and network set up was intentionally minimalistic.

Initially the capture set up consisted of one pc dedicated to attacking, one pc dedicated to sniffing data sent across the network, and one machine to host various vulnerable services to be attacked. These machines were manufactured by Dell and are model E-6100 Desktops. They were connected via a Cisco switch. Figure 1 introduces the capture environment.

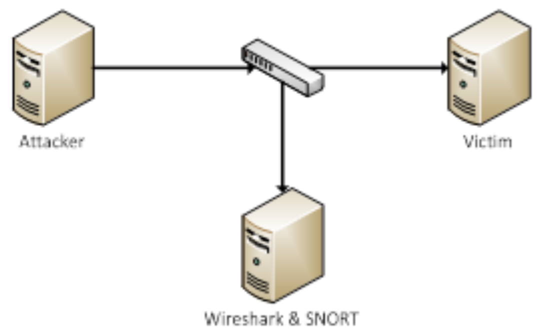

Figure 1. Isolated network for PAL creation.

.

Virtual machines were also added to our capture infrastructure. In many instances, the simple network outlined in Figure 1 still produced additional and unnecessary network "chatter" including STP, ARP Requests, and various network broadcasts. The use of two virtual machines on a single host helped to eliminate these issues and often produced a cleaner PCAP attack file. In these situations the attacking virtual machine also served as the sniffing machine.

The framework we created to build the .pcap Attack Library (PAL) is broken into two separate phases; the first phase is used to capture the actual attack while the second phase is used to test and complete the capture file. The first phase of the framework is presented in Figure 2.

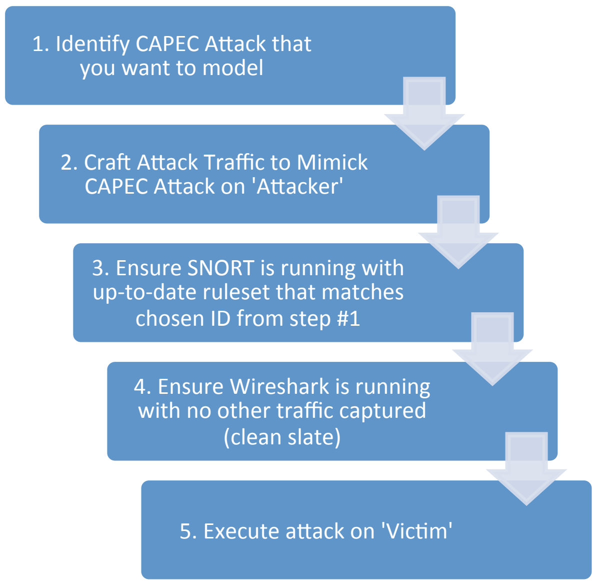
1. Identify CAPEC Attack that you want to model
2. Craft Attack Traffic to Mimick CAPEC Attack on 'Attacker'
3. Ensure SNORT is running with up-to-date ruleset that matches chosen ID from step #1
4. Ensure Wireshark is running with no other traffic captured (clean slate)
5. Execute attack on 'Victim'

Figure 2. Process used to capture the attack traffic

Each PAL capture corresponds directly to an attack in the CAPEC dictionary. The steps to accurately capture and document each .pcap Attack file are introduced in Figure 3.

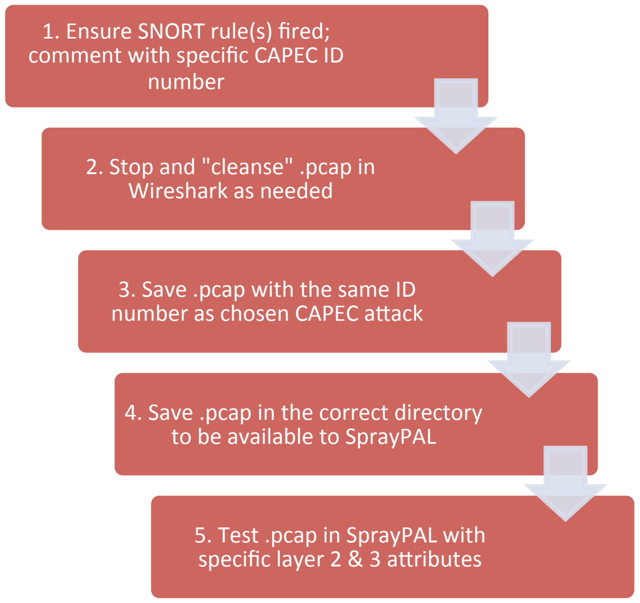
1. Ensure SNORT rule(s) fired; comment with specific CAPEC ID number
2. Stop and "cleanse" .pcap in Wireshark as needed
3. Save .pcap with the same ID number as chosen CAPEC attack
4. Save .pcap in the correct directory to be available to SprayPAL
5. Test .pcap in SprayPAL with specific layer 2 & 3 attributes

Figure 3. Steps to cleanse and document attack traffic.

The goal of mapping real attack traffic captured in .pcap files to corresponding CAPEC definitions and Snort IDS rules, is to provide instructors and network administrators a way to quickly sort and search through the growing attack library. Users can identify a CAPEC definition for the attack that they want to perform. The result is a framework that saves both time and effort for instructors and system administrators wanting to teach defensive techniques (rule set creation) or test their current configurations.

The first step in identifying candidates for the PCAP attack library was to choose common attacks which were easy to setup or configure. This process reduced the amount of time required to fully understand the details of a given attack. After initially reviewing the CAPEC attack dictionary we identified nearly 100 attacks and added them to the potential capture candidates list. This list was further narrowed down to a more manageable subset of 12 attacks.

Once a particular attack has been chosen, the attack is performed while the data on the network is being captured. Upon successful completion of the attack, the network traffic capture is then terminated and the data is saved. For our purposes we do not wish to include any irrelevant packets of data in the capture file. Wireshark allows us the utilization of filtering the captured packets. Filters are built to restrict the packets displayed to only the packets sent from the attacking machine to the victim machine. Once this has been completed, the file is then saved. Wireshark gives the user the option of saving either the entire capture or only the displayed capture. Displayed packets are the packets with filtering enabled, so this is the option that is chosen. This is the portion of the procedure known as "sanitization". This process allows us to more efficiently reproduce the attack since we are not flooding the network with responses from the victim machine, miscellaneous network traffic from normal operation of network infrastructure components, or any other types of traffic on the network at the time of the capture.

Once the network traffic has been captured and saved, we then perform research to locate an appropriate SIDSr for this particular attack. The Snort IDS comes with several pre-constructed rules. This is the first place that is searched for a rule that corresponds to the performed attack. If this initial search fails to provide us with a rule or a good basis for the rule that we need, then an internet search is performed. If these two searches produce no rules then a rule is written from scratch. When creating a SIDSr great care is taken to try and ensure the lowest possible number of false positives result from the rule. In the case of the SIDSr that detects a SYN scan, it also detects a TCP connect scan. This is of course because the SYN scan performs the first 2 parts of the TCP/IP protocols 3-way handshake. Since the TCP connect scan performs a full 3-way handshake, it is logical then for the SYN scan SIDSr to detect the TCP connect scan. Some situations like this cannot be prevented and therefore some false-positives are possible. However, these false-positives can be detected. In the case of the SYN scan SIDSr, if a SYN scan event is triggered and a TCP connect scan is not, then the SYN scan was used. However, if the SYN scan event triggers in addition to the TCP connect scan then we know a reasonable amount of certainty (given a very close temporal relationship between the two events) that the SYN scan was only triggered by the TCP connect scan and thus can be disregarded as a false-positive.

Upon successful completion of the attack, capture of the attack network traffic, and the formation of an appropriate SIDSr, the entire system can be tested. The attack traffic can be replayed from any PC using a pcap replay tool such as TCPReplay. Once the capture process is complete, the attack is then replayed against an IDS and the corresponding SIDSr. If the attack was correctly performed and cataloged, the SIDSr will trigger an event corresponding to this particular attack and generate an alert on the IDS.

Given a successful replay of traffic, the captured network traffic file, a brief README file describing the attack, and the SIDSr file are zipped together into a single file. This allows for easy access to all parts of the cataloged attack data.

4. Attack Selection for the PAL

As previously mentioned, the CAPEC dictionary was used to provide a list of possible attacks for inclusion into the PAL. These attacks were chosen based on an estimation of time requirement and complexity for cataloging the attack. Below we present a brief outline of each of the captured attacks. These attacks were all setup, performed, and successfully cataloged according to framework presented above. These attacks are all completed and available to be downloaded free of charge. The nine attacks and their individual characteristics are as follows:

4.1 Password Brutforcing – CAPEC 49:

Password bruteforcing is the attempt of every possible combination or value for a password. In this scenario an attacker will eventually discover the correct password.

To perform this attack we utilized an instance of the attack tool Hydra to simulate a bruteforce attack against an FTP server.

4.2 SQL Injection – CAPEC 66

SQL Injection attacks are popular as well as XSS attacks [4]. Based on this popularity, its inclusion as a candidate was assured.

SQL Injection attacks were made easy to perform by setting up a vulnerable website. We used the OWASP WebGoat program to create such vulnerabilities and as such were able to perform the SQL Injection attack with relative ease [12]. Because this attack is relatively easy to detect, we were able to catalog this particular type of attack quickly. The majority of time spent on including this attack was learning how it worked and how to perform it.

4.3 Embedding Script (XSS) – CAPEC 86

Cross-site Scripting (XSS) Attacks were a prime candidate based on its growing popularity in the global networking infrastructure [13]. Based on data provided by Symantec Corporation [4], this type of attack has become extremely popular and as a result was a prime candidate for consideration for inclusion in the library.

The majority of the time required to add this attack to the library was used in understanding what a XSS is and how to perform the attack. Once we knew how to perform an XSS, more time was needed to develop a vulnerable web page. This was accomplished using Microsoft's Internet Information Services 6.0 running on Windows XP Professional.

The primary consideration for its inclusion in the library was its popularity. There are many different ways to perform an XSS attack, which makes detecting all possible attacks quite difficult. However, since a simple XSS attack is not that difficult to perform, it was selected as a final candidate. Using the Snort IDS's ability to use Perl Compatible Regular Expressions (PCRE's) detecting many types of XSS attacks with one rule was possible. This was a secondary consideration when choosing this rule for inclusion in the library

4.4 Web Server Application Fingerprinting – CAPEC 170

Web server application finger printing requires the attacker to send network traffic to the target in an attempt to elicit a response from the web server. The goal is to identify the specific software version or type based on the unique response to the initial traffic.

To accomplish this capture we utilized Httprint. This tool sends a series of probes to the target in order determine software and version information.

4.5 TCP SYN Scan – CAPEC 287

SYN scans were included because they are easy to produce and represent a popular choice of attack. SYN Scans are also easy to detect and there are pre-existing SIDSr's. One issue of concern when capturing the SYN Scan was that most SYN Scan SIDSr's are unable to distinguish between traffic resulting from a SYN scan and traffic from a TCP Connect scan. This problem is due to the fact the SYN Scan is the first two parts of what is known as the three-way handshake. Since the TCP Connect scan performs all three parts of the handshake it will naturally trigger the SYN scan event due to its performance of the first two portions of the three-way handshake.

4.6 IMCP Echo Request Ping– CAPEC 288

During the initial testing of our infrastructure we decided that one of the easiest ways to test was to initiate a simple ping scan against our host. Given that a SIDSr already existed [14], we set up the sniffing machine with the simple ping scan rule and ran our scan. The SIDSr indeed triggered an event. However, when we replayed the captured traffic using a replay tool and the SIDSr failed to trigger an event. After changing the layout of our systems and verifying that the live simple ping scan triggered an event, we attempted to replay the traffic again, this time using TCPReplay. This time the SIDSr triggered an event.

This particular scan was initially chosen because it is quite simple to implement. This made it a prime candidate for our process as well as a great candidate for testing our infrastructure.

4.7 TCP Connect Scan – CAPEC 301

Given that we cataloged the SYN Scan, we felt it necessary to be able to distinguish between the SYN Scan and the TCP Connect scan. The best way to accomplish this was to add a SIDSr that detects the TCP Connect scan. Doing so would allow us to know when the SYN event was triggered, if the result was truly a SYN Scan by examining whether or not the TCP connect scan event was triggered as well. If both events triggered, this indicates a TCP Connect scan and not a SYN Scan. Therefore any previous event showing a SYN scan can be ignored.

The primary reason for choosing the TCP connect scan as a final candidate was that it was both easy to perform and easy to detect. Since we needed this scan to differentiate between the SYN scan and the TCP connect scan it further added to its need for inclusion in the library.

4.8 TCP XMAS Scan – CAPEC 303

The XMAS Tree scan was chosen as a final candidate due to our inability to detect various types of Denial of Service (DoS) attacks. We were unable to correctly implement a SIDSr to alert when such an attack occurred. As a result we replaced the DoS with the XMAS tree scan.

Nmap makes performing this scan very simple. This allowed us to add another scan to the library with minimal time invested. An additional reason that this scan was selected as a final candidate included the fact that is was simple to detect. Pre-existing SIDSr were readily available [14]. Since so much time was invested in learning how to perform a DoS attack, it was a good idea to then replace that attack with one that was easy to perform.

4.9 TCP NULL Scan – CAPEC 304

Null scans were chosen primarily because the infrastructure, capture configurations, and tools required to perform the attack were already in place. Nmap had been used to capture several other scans. As a result, NULL scans were easy to perform. In addition, NULL scans were readily detectable using a preexisting SIDSr.

5. Attacks Not Included in PAL

5.1 Denial of Service Attack

One of our original primary candidates for inclusion in the library was the Denial of Service attack. This attack is still quite popular since it can be easy to perform using the right tools. At the time the decision was made to no longer include this attack in the initial proof of concept library, we could not find an appropriate Snort rule set. It was decided that, based on our time constraints, we would delay the inclusion of this attack until a later date.

5.2 Sniffing Attack

The majority of our time was spent researching network sniffing as a type of attack. Many experiments were performed in order to attempt to detect sniffing. Our current efforts were unable to detect a change made to the regular network traffic on the wire upon the beginning of a sniffing attack. Our experiments focused on continuously putting a NIC into promiscuous mode while sniffing traffic on our network. We were unable to find traffic to indicate that a card had switched to promiscuous mode. Furthermore, it was discovered that there are tools freely-available to users that allow for detection of sniffing attacks. This software works by sending out ARP packets and then examining the machine's response to the packets. Further research will be done on this attack to include this attack in future releases of PAL.

6. Conclusions and Future Work

PAL is available to download and utilize free of charge. We encourage the community to help grow the PAL by completing the steps outlined in section 3 above. The process is simple and straight-forward. Captures can be completed with minimal network infrastructure and resources or the through the use of virtual machines. The use of an isolated network ensures only traffic from "Attacker" to "Victim" is captured, thus less cleansing of the .pcap files needs to be done.

"Attacker" software is at the discretion of the user. We encourage adopters to make user of widely available tools so others can validate your .pcap files with the same toolset. Each attack must be mapped to an individual CAPEC Attack Pattern and be documented as such in the .pcap file. These attack .pcap files should have as much unnecessary traffic as possible removed. Once attacks are captured, various tools can be used to manipulate layer 2 and 3 parameters during replay. Traffic can be captured and .pcap files created in Wireshark (http://www.wireshark.org/). Utilizing Wireshark on a Windows machine requires the installation of WinPCAP as the packet capture and filtering engine (http://www.winpcap.org/). Attack traffic can be monitored and alerted on with SNORT (http://www.snort.org/).

Our work is significant from the standpoint of current academic and industry professionals looking for an easy way to teach and test intrusion detection systems. Instructors and administrators no longer need to possess the technical tools or know the details of performing various offensive security attacks. Each pcap file provides the user with the ability to quickly and accurately recreate various attacks. Our attacks also serve as a valuable addition to the current CAPEC attack library. When paired with the CAPEC dictionary, our attacks provide viewable, replayable, instances of each attack description.

Furthermore, this work also produces a framework for future collaboration and growth. This framework is important as it will serve to keep the PAL standardized and usable across professions and industries.

These files will be indexed by CAPEC ID for searches. The entire library exists currently as a compressed folder whose contents are .pcap files containing each entry in the library. The process has been documented which allows for future attacks to be cataloged and added to the PAL.

.

ACKNOWLEDGMENT

This work was supported in part by a grant from the National Science Foundation (NSF) under grant CNS 1004843 and by the National Center for the Protection of the Financial Infrastructure at Dakota State University.

References

1. mitre.org, " Common Attack Pattern Enumeration and Classification," http://capec.mitre.org July 23, 2010
2. Cho, K, Mitsuya K, Kato, A, "Traffic Data Repository at the WIDE Project," 2000 USENIX Annual Technical Conference, June 2000
3. Lee, C, Copeland, J, "FlowTag: A Collaborative Attack-Analysis, Reporting, and Sharing Tool for Security Researchers," ACM VizSEC '06 November 3, 2006
4. Fossi, M, Turner, D, Johnson, E, Mack, T, Adams, T, Blackbird, J, Enstwisle, S, Graveland, B, McKinney, D, Mulcahy, J, Wueest, C, "Symantec Global Internet Security Threat Report Trends for 2009," Volume XV, April 2010
5. S. Bhagyavati, "Agyei-Mensah, Rose Shumba, Iretta BC Kearse, Teaching hands-on computer and information systems security despite limited resources," 2005.
6. J. Hill*, et al.*, "Using an isolated network laboratory to teach advanced networks and security," *ACM SIGCSE Bulletin,* vol. 33, pp. 36-40, 2001.
7. M. Bishop, "Teaching context in information security," *Journal on Educational Resources in Computing (JERIC),* vol. 6, 2006.
8. S. Perez-Hardy, "A unique experiential model for teaching network administration," 2003, pp. 119-121.
9. M. Motsidi*, et al.*, "New Approach in Teaching Network Security Subjects," presented at the International Conference on Information (ICI9), Kuala Lumpur, 2009.
10. B. Hartpence and L. Hill, "Wireless carts: an inexpensive education and research platform," presented at the Proceedings of the 6th conference on Information technology education, Newark, NJ, USA, 2005.
11. B. Hartpence, "Teaching wireless security for results," 2005, pp. 89-93.
12. OWASP, "WebGoat Project," http://www.owasp.org/index.php/Category:OWASP_WebGoat_Project April 18, 2010
13. McKinney, D, "Cross-site Scripting Vulnerabilities," Symantec Connect (http://www.symantec.com/connect/blogs/cross-site-scripting-vulnerabilities) , July 4, 2006
14. "Snort IDS Rules," https://www.snort.org/downloads/snortrules-snapshot-2853.tar.gz July 22, 2010