Deleon: Network Activity Replay

Erin L. Johnson, Dr. Jack Tan
Department of Computer Science
University of Wisconsin – Eau Claire
Eau Claire, WI
{ johnsone, tanjs } @uwec.edu

Abstract

Deleon is an open-source network activity playback tool that aids computer incident responders in both analyzing attacks and reducing the window of vulnerability for systems.

1. Introduction

From July 1, 2006 to December 1, 2006 Symantec Systems, creator of Norton Antivirus, documented 2526 new software vulnerabilities with 73% of those vulnerabilities considered moderate to high in severity [1]. As with all new vulnerabilities there is a window of time where no countermeasures exists [2]. Thus, across the academic, corporate, government, and home sectors, systems were not protected against these 2526 exploits until computer incident responders could detect, analyze, and develop both a countermeasure and a recovery tool.

The reason for the lag between exploit detection and the creation of a viable counter measure is that incident response is a two-step process. First, the systems administrator must respond to an identified compromised machine, analyze it, recover data, and, if possible, return the system to its last known good state. Second, an incident responder must take the necessary actions to not only prevent the incident from reoccurring on the system, but also protect other systems vulnerable to the attack [3]. Incident analysis is a vital part of incident response but it is time-consuming; thus, responders must use tools that expedite the process as much as possible without missing possible incident vectors [4].

Initially, the author of the TCPReplay suite had taken under development a program, flowreplay, to parse network activity logs and replay that data [5]. Such a program would grant incident responders the ability to replay attacks in a known isolated environment in real or pseudo real time. By containing replay activity within an isolated test environment further harm to the originally compromised machines is prevented. This functionality augments both phase one and phase two of the incident response life cycle in the following ways.

First, by providing an environment to study the attacks "live", incident responders can filter out the dataflows as little or as much as they want. This process allows them to either narrow the focus of their study to a few packets or review the big picture of the logged network activity at the time of incident.

Second, by creating a reproducible attack vector, the utility can be part of a testing methodology during the development of countermeasures. In addition, a pseudo real time feature lets attacks be replayed in quarter or half time thus narrowing the time frame between incident detection and neutralization.

In 2007, however, the author of flowreplay abandoned the project. Thus, the goal of the Deleon was to develop an open-source tool that could be used as a corner stone for a flowreplay replacement.

2. Development and Test Environments

During the course of Deleon's development and testing several elements, such as virtual vs. physical environments, base operating system, and test environment, have stayed constant. Conversely, other aspects of Deleon, like the deployable format and programming language used for development, have gone through several iterations. That said, in all cases, two factors guided the choices made.
The first factor was cost. Deleon is an undergraduate research project with minimal funding; as such free solutions or solutions that came with minimal cost were necessary. The second factor that guided development decisions was the general availability of a solution. During Deleon's evolution, from its original conception as a plug-in to its final implementation as a stand alone project, it became clear that the program would not be able to implement a vast number of protocols due to time constraints. As such, barring the initial choice of Erlang as a programming language, all of the explored development vectors were chosen with the ease by which an outside developer could modify the code and utilize the framework that drives Deleon to meet their needs.

2.1. Virtualization

Virtualization was chosen over physical machines for developer convince as well as a budget that made purchasing hardware for a development and test environment cost prohibitive. The most accessible hardware available during development was the developer's MacBook Pro and since the operating system of choice for Deleon was Ubuntu, VMWare Fusion, a system virtualization program, was selected [6].

2.2. Operating System

Ubuntu 8.10, Intrepid Ibex, was the operating system chosen for both development and testing [7]. This decision was due to Ubuntu being a Linux variant and, thus, the developer would have use of all protocol header files native to Linux, much like the TCPReplay suite. Because of this decision Deleon utilizes known good structures rather than containing many reinvented wheels.

2.3. Test Environment

A honeynet was used for the test environment due to the developer's previous experience in deploying a honeynet [8]. Honeynets are networks of apparently unsecured computers ripe to be attacked and in actuality serve no other purpose but to be attacked [9]. Because of this, all traffic can be considered malicious and enables Deleon to be tested using real traffic logs while at the same time preventing the abuse of benign network traffic.

3. Deployables

During the exploration phase of Deleon's development, several deployment methods were explored. Even though the TCPReplay suite is a collection of utilities, those utilities create and read from a static cache file. Due to the dynamic nature of resending dataflows it was decided early on that a database approach would be much more appropriate and give applications greater flexibility when handling target computer responses.

It became clear that Deleon could be developed as a replacement for flowreplay "in spirit" and integrate with other applications rather than be a true replacement for flowreplay in the TCPReplay suite. As such, developing a plugin for Metasploit and Wireshark were explored. Ultimately, however, writing a stand-alone application was chosen

3.1. Metasploit

Initially the development of Deleon as a Metasploit Framework (MSF) module was investigated. MSF is a widely used, extensive exploit development platform for research and testing purposes [10]. As an MSF module, incident responders could more easily utilize Deleon; however, due to MSF being a framework to send out known patterns of behavior rather than digest and then reproduce known behavior, this route was ruled out.

3.2. Wireshark

Next, creating an extension to Wireshark, an open source network protocol analyzer, was researched. Again, due to Wireshark already being a commonly used tool, it would have yielded a wider audience as well as minimal re-invention of the wheel so-to-speak where packet analyzing is concerned [11]. After studying the code base this route looked to be viable, however, far too ambitious for the time constraints for this project. However, it is reasonable to view Deleon as tool to be used in conjunction with Wireshark.

3.3. Stand-alone Tool

In the end it was decided that Deleon would be a stand-alone open source toolto be used in conjunction with network activity loggers and viewers such as Wireshark, much like the TCPReplay Suite. Due to the number of protocols in existence as well as the time constraints for the project it was also decided to focus on developing a framework that would require external developers to modify as little source code as possible in order to add support for other protocols.

4. Languages

During the project planning phase two languages were explored as development candidates: Erlang and C++.

4.1. Erlang

The first language to be explored was Erlang, a functional programming language, along with its native database Mnesia. Erlang was considered because it is a functional language as well as its ability to perform complex bit masking and create hyper threads called processes.

Erlang by nature is made for lightweight threads where system derogation does not occur until around the 20 000 mark [12]. By being able to create thousands of threads the original design included implementing computer process for each sending IP/MAC address pair. Thus, each process would be independently responsible for sending out its data based on a global timer.

Unfortunately, Erlang's limitations, including inconsistent handling of character arrays (e.g.: treating strings as character arrays in one instance and then representing character values as integers the next) proved to be too erratic to be chosen as a development language. In addition, due to Erlang's lack of wide use, choosing Erlang would have limited the accessibility of Deleon's source code [13].

4.2. C++

After Erlang was discarded, C++ was chosen even though TCPReplay was written in C because of the object-oriented programming paradigm (OOP) that was implemented in the design of Deleon. One advantage to OOP is the reusability of code in separate parts of the project. As such, code used to parse network activity logs could be used again later to reform packets to replay. In addition, by selecting C++, known libraries such as libnet and libpcapthat could be used for packet manipulation. In addition, the developer's previous low-level network programming had been in C++ due to aiding in developing and writing an FTP program for UWEC's Networks course.

Once C++ was selected, four main libraries were chosen to aid in implementing Deleon:

- libpcap: headers for libpcap files and packets
- libnet: packet formation and sending
- mysql: interaction with the MySQL database
- ncurses: basic user interface

4.2.1. Libpcap 0.0.5

Libpcap is a library that provides headers and functions for network traffic capture and was developed by the Network Research Group at Lawrence Berkeley Laboratory [14]. While Deleon does not have a capture component, many network activity loggers output their captures as PCAP files. As such, there are two main libpcap structures used within Deleon: pcap_hdr and pcap_pkthdr.

The first 192 bytes of a pcap file contain PCAP specific data such as the PCAP version, the data link type, and the timestamp of when the file was initially created. This provides a context of sorts for the rest of the data flow.

While pcap_hdr provides the context for the log, pcap_pkthdr provides context for the individual packets. It lists the timestamp of the packet, the amount of data that should have been captured, and the amount of data that was actually captured. These captured lengths are used as guards when reading in the raw datablocks.

In addition to providing data not contained within a raw dataflow, both of these structures circumvent the need for conditional statements and multiple reads of the data just to identify the packet length thus making Deleon more efficient.

4.2.2. Libnet 1.1.3.8

Libnet is a high level interface for low level network programming. Through the use of predefined functions packets can be built by hand yet at the same time compute the appropriate check sums. In addition to this, libnet also provides an interface by which said built packets can be released onto the network at the link layer [15].

Both of these functions of libnet kept the developer from having to reinvent the wheel. Much like choosing Ubuntu as the base OS due to its preexisting headers, libnet provides a reliable tested means of packet creation and sending.

4.2.3. Mysql 3.0.0

MySql was chosen as the database due to its documentation as well as the preexisting C++ mysql library. This library facilitates connections between a C++ program and a MySQL database as well as formatting and parsing out data from query results [16].

4.2.4. Ncurses 5.6

One of the personal frustrations of using the TCPReplay suite is the command line interface and use of chaining flags. Thus, due to the logic required for mapping computers existing in the logs to known test boxes, it was decided that something more robust than a command line user interface was necessary [17]. Ncuses was used to implement a terminal style user interface. This library was chosen due to its low overhead and the ease by which it could be implemented.

5. Database Design

After studying TCPReplay and its use of cache files to create static dataflows and replace computer information (e.g. IP addresses), it was decided that incorporating a database into Deleon's design would provide much more dynamic and pervasive data storage. By utilizing a database, network activity logs can be parsed out by packets and then those packets can be used to create computer specific dataflows. This ability becomes important when a dataflow is replayed in a test environment and the sender must adapt the flow to dropped or corrupt packets.

After Erlang and its native database Mnesia were ruled out, MySQL 5.0.67 was chosen [18]. MySQL's selection was based on its price (free), as well as the aforementioned MySQL C++ library.

The database schema, as seen in Figure 1, is fairly simplistic yet yields both the flexibility needed to create dataflows and remap logged computer data to test computer data. This schema also contains enough human readable and raw packet data to reform packets.

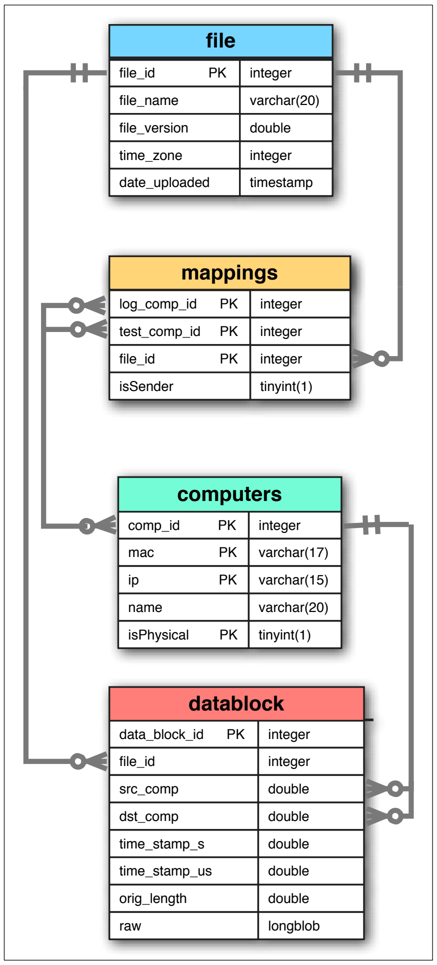
5.1. Computers and Mappings Tables

In Deleon a computer object is defined as a MAC Address and IP pairing. These pairings are either parsed from network activity log files or entered manually by the user. While parsed computers are linked to packets and data flows, entered computers are used as targets when replaying the network activity logs.

The computers table is completely log file independent because it is reasonable to assume that most log files loaded into Deleon will be from the same network. Thus, having an independent table yields one computer object mapping to many data flows rather than many data flows mapping to many computers.

The mappings table contains a listing of log computer identifiers and test box computer identifiers. Before Deleon relaunches network activity traffic it performs a join on the computers, mappings, and datablock tables and applies the mappings before the raw data is reformed into packets.

In the current version of Deleon, unlike the computers table, the mappings table is log file dependent. In future versions of Deleon, an additional linking table will be added that will contain a list of file identifiers and a reference to a computer mappings identifier to eliminate duplicate mapping entries.



file

file_id	PK	integer
file_name		varchar(20)
file_version		double
time_zone		integer
date_uploaded		timestamp

mappings

log_comp_id	PK	integer
test_comp_id	PK	integer
file_id	PK	integer
isSender		tinyint(1)

computers

comp_id	PK	integer
mac	PK	varchar(17)
ip	PK	varchar(15)
name		varchar(20)
isPhysical	PK	tinyint(1)

datablock

data_block_id	PK	integer
file_id		integer
src_comp		double
dst_comp		double
time_stamp_s		double
time_stamp_us		double
orig_length		double
raw		longblob

Figure 1: Deleon Schema

5.2. File and Datablock Tables

The file table and datablock tables are the backbone of the database. File information, timestamps, and raw packet data are all stored in these two tables regardless of protocol. It is also from here that Deleon pulls the necessary information to begin parsing stored datablocks as well as retrieve the datablocks themselves when building data flows to relaunch at a test environment.

The file table contains mostly historical data such as the log file's name, the date it was uploaded into the database, the PCAP format version, and the timezone in which the original network activity was captured.

Most of the data contained in the Deleon database is stored in the datablock table, however most of the data is not human readable. Due to the decision to make the schema as protocol independent as possible, Deleon maintains an exact copy of the packet in the raw field. By pulling down this information protocols can be parsed and new packets be formed.

6. Processes

With Tcpreplay, network activity log parsing is a two step process. First, the PCAP file is split into sender and receiver flows which creates two cached PCAP files using tcpprep. The second step is to edit packets in those cache files specifically indentifying information such as MAC addresses and IP addresses.

As expected, since Deleon utilizes a database, the log parsing aspect of Deleon is quite different. Firstly the PCAP logs are parsed and uploaded into the database. Then, the mapping of logged computers to test boxes occurs. Due to the design of the database however, the mapping does not need to be updated for each packet. Rather, it is the computer objects that are linked, thus circumventing the need for extraneous data processing.

6.1. Parsing PCAP Files

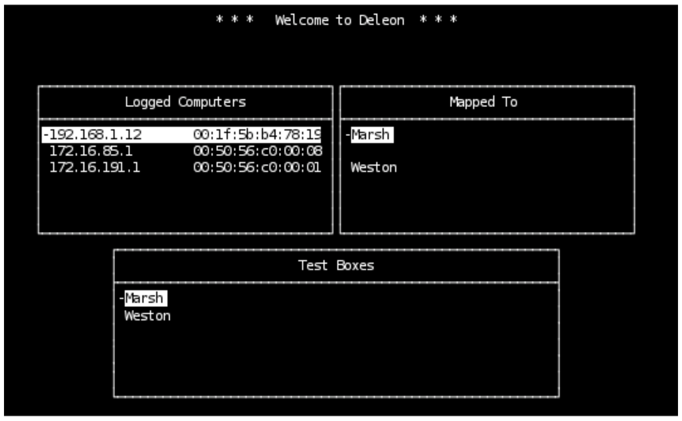
Once a PCAP file has been opened in Deleon, a buffered read of the PCAP file begins following the Factory Method design pattern.

Even though initially the PCAP header and the raw data for a packet is read in via aPcapParser and then passed to Deleon's native packet type, the D_Packet, the heavy lifting of protocol dissection occurs in the ProtocolFactory. As the D_Packet cycles through the various OSI layers of the raw data, the ProtocolFactory is passed the protocol identifier as a key in the appropriate OSI layer map and a new object of that protocol type is returned. From here, a pointer to the memory address of the protocol header is passed to the new protocol object and the header is parsed out.

One unique feature of Deleon is the way computer specific information is handled. Each D_Packet contains two long pointers: src_comp_id and dst_comp_id. These two variables point to memory addresses that are the values contained in a map in the main log parser class. This map's key is a struct containing the computer object MAC and IP paring while the value is set the computer's index within the computers table. These mappings are created so that it is necessary to update a computer index in only one centralized location rather than cycling through the entire datablock set again.

6.2. Computer Mapping

To resend log data safely without risk of re-attacking a production environment, Deleon has the ability to map logged computer objects to known testboxes. By default, each computer object is unmapped and cannot be sent. An override can occur whereby a computer will stay mapped to itself. The mapping mechanism is largely driven by the user interface (UI) written with the ncurses library.



```
                    * * *   Welcome to Deleon   * * *

   Logged Computers                          Mapped To
  -192.168.1.12       00:1f:5b:b4:78:19     -Marsh
   172.16.85.1        00:50:56:c0:00:08
   172.16.191.1       00:50:56:c0:00:01      Weston

                           Test Boxes
               -Marsh
                Weston
```

Figure 2: Mapping ncurses User Interface

When the UI is displayed the computers found in the last log uploaded is displayed as well as the possible test boxes. Once a mapping is made by tabbing and scrolling through the interfaces the mapping object is then stored in a map and passed to the mysql handler to be uploaded to the database. Once the log has been parsed, data is uploaded to the database.

6.3. Traffic Launcher

The traffic launcher portion of Deleon is primitive at this point and was developed to test the parser. That said, like the log parser, the traffic launcher utilizes the Factory Method design pattern and, in fact, uses the same class, ProtocolFactory. Where as before the packet was being deconstructed from bottom to top, the ProtocolFactory constructs a protocol specific libnet context for each layer and builds the packet from the top down. Each libnet function takes a payload context pointer; thus, the appropriate nesting occurs and the correct check sums are calculated.

The traffic launcher knows nothing about the original computer objects; rather, the MySQLer object replaces the original computer data with the mapped computer data before it passes the resultset, which includes the raw data of the packet, to the ProtocolFactory. The Protocol Factory once again looks up the protocol's identifier in a map and returns a new protocol object of that specific type.

Once the bottom layer protocol is returned (with the nested upper layer protocols in it's payload), the D_Packet is put on a queue while its timestamp offset from the first packet is stored in another queue. This information is stored so that when the packets are launched onto the network the original traffic pattern can be emulated. Also, storing this information allows for a quantum of time other than those of real-time to be used so that incident responders and researchers may study the traffic flows at varying speeds.

7. Conclusions

The main objective of the Deleon project is to build a foundation for future work designed to develop a tool that incident responders can use to recreate attacks as well as build and test counter measures. Deleon will be a replacement for the now abandoned flowreplay utility.

Due to a number of factors, Deleon was designed with extensibility in mind rather than as a closed, stand-alone project. This goal was achieved through the use of generalized structures in the code base and database as well as utilizing preexisting headers and protocol identifiers native to the Linux platform.

By developing with these two aspects in mind, Deleon is able to incorporate new protocols by simply modifying a specific Deleon native protocol class and adding an object of that class to a structure within the ProtocolFactory. Because of this design, Deleon is a solid base from which other incident response tools can be built with further research and development.

References

[1] Symantec, "Symantec Internet Security Threat Report Trends for July–December 06," [Online document], 2007 Mar, [cited 2007 Sept. 8], Available: http://eval.symantec.com/mktginfo/enterprise/white_papers/ent-whitepaper_internet_security_threat_report_xi_keyfindings_03_2007.en-us.pdf

[2] Gonzalez and Radianti, "Understanding Hidden Information Security Threats: The Vulnerability Black Market," in Proc. 40th Annual Hawaii International Conference on System Sciences. (Waikoloa, Big Island, Hawaii, January 3-6, 2007, pp. 156-166).

[3] W. Brown, J. Moira, D. Stikvoort, K.P. Kossakowski, G. Killcrece, R. Ruefle, and M. Zajicek. "Handbook for Computer Security Incident Response Teams (CSIRTs)," [Online document], 2003 Apr., [cited 2007 Mar. 9] Available: http://www.sei.cmu.edu/publications/documents/03.reports/03hb002.html

[4] K. R. van Wyk and R. Forno, Incident Response. Sebastopol, CA: O'Reilly, 2001.

[5] Aaron Turner, "Tcpreplay," [Online Document], [cited 2009 July 15], Available: http://tcpreplay.synfin.net/trac/

[6] VMware, "VMWare Fusion: Run Windows on Mac, for Mac Desktop Virtualization," [Online document], 2009, [cited 2009 June 18], Available: http://www.vmware.com/beta/fusion/

[7] Canonical Ltd, "Ubuntu Home Paeg | Ubunut," [Online document], 2009, [cited 2009 June 18], Available: http://www.ubuntu.com/

[8] E.L. Johnson, J. M. Koenig, and P. Wagner, "Development and Implementation of the Honeynet on a University Owned Subnet," in Proc. 40th The Midwest Instruction and Computing Symposium. (Grand Forks, North Dakota April 20 - 21, 2007), [Online document], [cited 2007 Sept. 5] Available: http://www.micsymposium.org/mics_2007/proceedings.pdf

[9] Spitzner, Lance. Honeypots: Tracking Hackers. Boston: Addison-Wesley, 2003.

[10] The Metasploit Staff, "Metasploit 3.0 Developer's Guide," [Online document], 2007, Feb. 25, [cited 2007 Sept. 5], Available: http://framework.metasploit.com/documents/developers_guide.pdf

[11] Wireshark Foundation, "Wireshark: Go deep," [Online document], 2009, [cited 2009 June 18], Available: http://wireshark.org

[12] Armstrong, Joe. Programming Erlang: Softare for a Concurrent World. Dallas: The Pragmatic Bookshelf, 2007/

[13] T Nagy, AN Vig, "Erlang testing and tools servey," in Proc. 7th ACM SIGPLAN Workshop on ERLANG 2008, [Online Document], [Cited 2009 June 23] Available: http://portal.acm.org/citation.cfm?id=1411277

[14] Luis MG, "TCPDUMP/LIBPCAP public repository," [Online document], 2009, Jan 22, [cited 2009 June 18], Available: http://www.tcpdump.org/

[15] Repura, "Libnet 1.1 Tutorial for Beginners," [Online document], 2008, Sept. 13, [cited 2009 June 18], Available: http://repura.livejournal.com/31673.html

[16] MySQL Ab, "MySQL::MySQL 5.0 Reference Manual :: 20.9 MySQL C API," 2009 [cited 2009 June 18], Available: http://dev.mysql.com/doc/refman/5.0/en/c.html

[17] "NCURSES - New Curses" [cited 2009 June 18], Available: http://invisible-island.net/ncurses/

[18] MySQL AB, "MySQL:: The world's most popular open source database," 2009 [cited 2009 June 18], Available: http://www.mysql.com/