Network Traffic Exploration Application: A Tool to Assess, Visualize, and Analyze Network Security Events

Grant Vandenberghe

Network Information Operations Section
Defence Research and Development Canada (DRDC)

Abstract. Defence Research and Development Canada (DRDC) is developing a security event / packet analysis tool that is useful for analyzing a wide range of network attacks. The tool allows the security analyst to visually analyze a security event from a broad range of visual perspectives using a variety of detection algorithms. The tool is easy to extend and can be used to generate automated analysis scripts. The system architecture is presented and its capabilities are demonstrated through the analysis of several covert tunnels.

Keywords: Packet Analysis, Network Forensics, Visualization, Covert Tunnels.

1 Introduction

Although many organizations recognize the value of an automated response to a cyber attack, the trust in the response is lacking. False positives are routinely seen in many systems and when a suspicious event occurs, an analyst is required to review the event. The review process determines if the event is malicious. If an alert occurs often enough then the analyst may write a script to validate it. Often, the time required to write a script is considerable and requires the skills of a highly trained analyst.

Described in this paper is a data visualization tool that allows the analyst to investigate network events and automatically create validation scripts that embed the analyst's experience into the security infrastructure. It also allows security teams to progressively transform security analysis from a manual to an automated defensive detection system.

Analysts make use of a range of tools to detect incursions, repel attacks, and eliminate false positives. A taxonomy of these tools is shown in Figure 1. Analysts use attack assessment tools such as vulnerability databases to understand intent and determine what types of systems are vulnerable. Vulnerable systems can be protected using attack abatement tools like a firewall. Although the analyst can limit network exposures using a firewall, some potentially malicious packets may still inadvertently enter the network. To detect malicious packets, the analyst will use either a host or network-based intrusion detection system (IDS). A network-based product examines all traffic crossing a point on the network. Alternatively, a host-based IDS is intended to monitor a single host.

J.R. Goodall, G. Conti, and K.-L. Ma (Eds.): VizSec 2008, LNCS 5210, pp. 181–196, 2008.

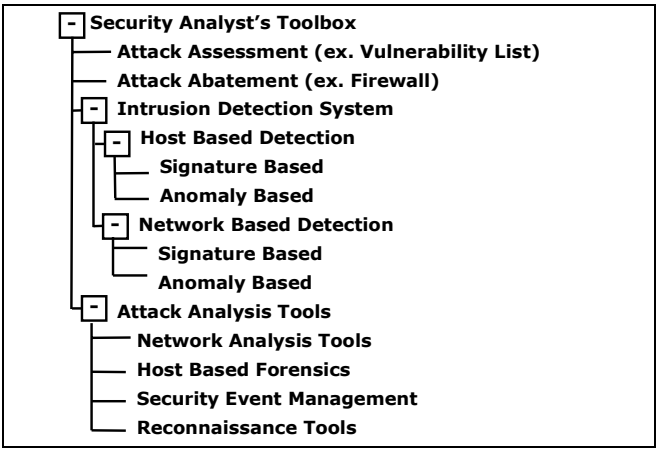

Fig. 1. Security Tool Taxonomy

The host and network-based IDS tools can be further categorized by their detection algorithms. Both anomaly-based and signature-based algorithms are capable of detecting a wide range of attacks but both have a number of weaknesses. Signature-based tools are only effective at detecting known signatures and rely on well-written signatures to reduce false alarms. Anomaly-based IDS's are able to detect unusual patterns in network traffic but often require an analyst to judge the significance of suspicious patterns [1,2]. Often, the resulting text-based security events do not adequately convey the magnitude of an abnormality that is best presented in graphical terms.

Some IDS systems are starting to support both signature-based and anomaly-based algorithms. However, most network-based tools have a fairly rigid architecture that makes adding new anomaly detection algorithms difficult. Even open source signature-based tools like SNORT [3] have relatively few anomaly-based plug-ins available for them.

A typical process used to collect and respond to a security event is shown in Figure 2. With respect to IDS implementation, some security events generated by the IDS are reliable enough to warrant an automated response. In other cases, the security events need to be reviewed by an analyst. All events generated by the IDS are sent to some type of database or security events management (SEM) system. Although SEM systems can correlate events that surround a potential intrusion, the analyst still needs to validate the event and determine a course of action [4].

There is a wide range of network-centric tools that can assist the analyst in analyzing a network-based event. A quick survey showed that there were more than 50 tools freely available [5-10]. Many of these tools complement one another but, in general, are not interoperable and do not provide unencumbered exchange of information among them. Rather than trying to discuss all 50 tools that have already been well covered in [5-10], four tools will be discussed.

SGUIL [11] is a GUI-based tool that combines SNORT [3], Wireshark [12], P0f [13], tcpflow [14] and libpcap [15] / tcpdump [16] into a single environment. Some reviewers have described SGUIL as a tool "by analysts, for analysts" [10]. This tool does not have any script writing capability and has limited graphical visualization options.

OPNET-ACE [17] is a commercial tool that allows an analyst to characterize application interactions. The product is not specifically tuned to security analysis but it does have a very strong visualization component associated with it.

BRO [18] is essentially an enhanced signature-based IDS but it includes a very robust scripting language. The tool significantly extends signature-based analysis and is easier to program than scripts written in PERL or C. However, it does not try to combine signature and anomaly-based methods together into a single environment.

CA- eHealth [19] is a network management system which can store and present the state of the enterprises networking equipment however this tool does not allow the user to examine packet activity on the network.

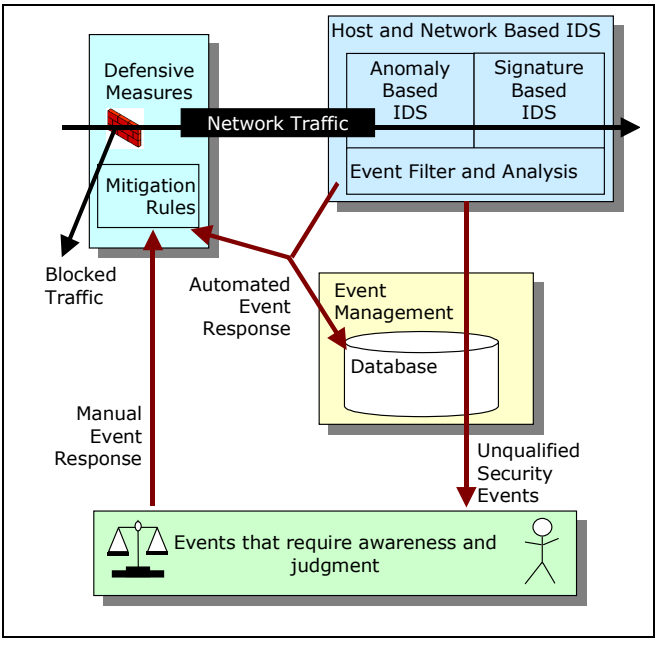

Fig. 2. Typical Cyber Defence Process

When the above tools are applied to the security event analysis process small feature gaps exist. These feature gaps can be summarized as follows:

- Many IDS's allow the analyst to add new signatures but most do not allow the analyst to add new anomaly detection algorithms.
- Most security-based tools deal with events on a text basis and do not exploit the full range of visualization options available to them.
- Scripting tools and analysis tools are highly separated. No existing tools take the experience gained from performing an analysis and use it as a basis for strengthening the defences in the future.
- There are many niche tools available but they are limited in their ability to exchange information.

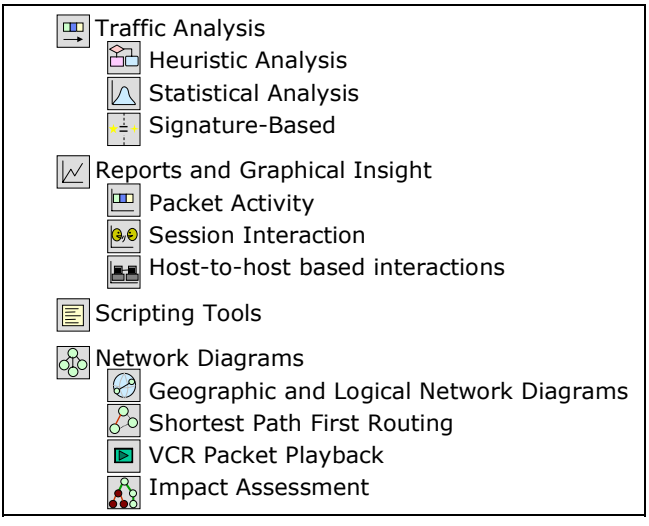

Fig. 3. Core Functional Areas of the NTE

- Most tools deal with the high level network or the low level packet analysis but there is little available to transition between the views. The ability to relate a tiny packet level event to the significance of the network at large is not well addressed.

To address the above limitations, the Network Traffic Exploration (NTE) tool was developed. This in house tool is being used to explore different event analysis approaches and is used internally for a wide range of packet analysis activities. This tool combines six key functional areas into a single package as shown in Figure 3. The tool includes rudimentary intrusion detection and extensive analysis functionality. NTE readily connects to the SNORT signature-based IDS and includes several simple anomaly-based features as well. The tool works with both anomaly detection algorithms and text-based logs. It uses a wide range of visualization features and text-based reports to bridge the gap between the different detection techniques. The tool can generate scripts and is open enough to allow the analyst to easily add new algorithms to any of the core areas.

NTE also provides several methods of traffic analysis. Each method can be used individually or techniques can be combined to provide a stronger overall analysis capability. Consider an ICMP packet exchange with the following abnormal characteristics:

- Unusual timing patterns
- Unusual packet length
- Multiple responses to a single query
- Incorrect ICMP checksums.

Individually, each characteristic is circumstantial but together they point to a potential covert tunnel.

Beyond the straight analysis capabilities, the NTE provides the analyst with the ability to develop and test customized traffic analysis scripts. Although there are many ways to create traffic analysis scripts, most require knowledge of a programming language like C or PERL. The NTE's approach allows an analyst to create scripts as a background activity and doesn't require prior programming knowledge. The analysis process is automatically stored by the NTE as the security event is processed. The NTE can be later asked to create a custom script based on the user's past actions. This custom script can be manually reused or potentially deployed to an unattended computer that would periodically run a series of these scripts in batches. An operational implementation of the design is shown in Figure 4. In this potential implementation, the script's execution would be triggered when IDS alerts matching certain characteristics are detected. The script would be rerun to validate the alert with the results fed back into a centralized event management system.

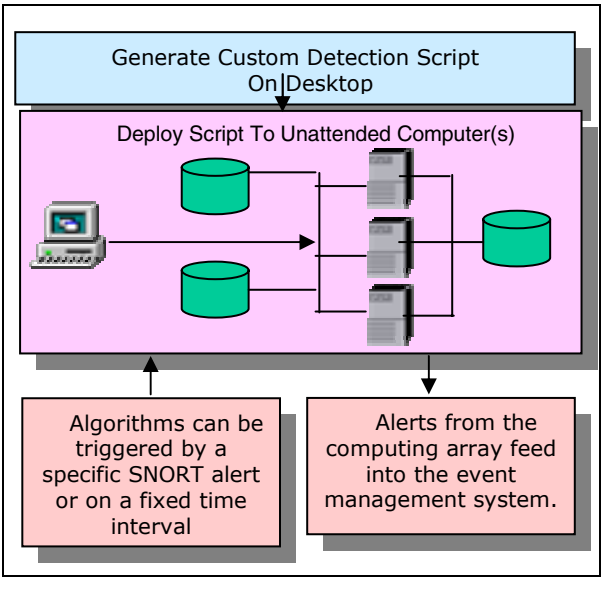

Fig. 4. NTE Process Workflow

2 Architecture

The previous section discussed the limitations of the current cyber defence tools. This section explains how NTE is structured to overcome these limitations using a layered software architecture and interwoven feature set. With respect to the software architecture, NTE has three layers. The software uses MATLAB as a development environment, a low level packet analysis library to perform specific tasks, and finally the NTE provides the unified application front end.

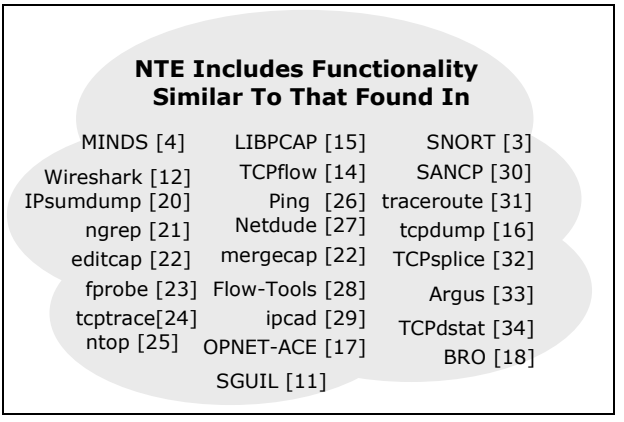

Fig. 5. NTE Scope (with references)

The choice of MATLAB as a development environment is somewhat unusual but has a number of advantages over a traditional high-level language. In a language like C or JAVA, all variables disappear (unless the data is stored to disk) when the program finishes executing. In MATLAB, the output of each algorithm is retained as variables in the desktop environment. This allows the operator to maintain a highly interactive interface with the data and makes portability between different algorithms much easier. In a C coding paradigm it would be the equivalent trying to code from within the debugger.

The packet analysis library was created by DRDC to provide a series of commonly used traffic analysis functions. The functions are largely written in MATLAB's internal language but there are some functions written in C and JAVA as well.

The NTE software is built on top of the packet analysis library. The tool extends well beyond the scope of the base library by adding an interactive GUI environment, additional data structures, data query capabilities, traffic profiling, help system and reporting capabilities.

The NTE has a broad suite of functions that are similar to that found in a broad range of analysis tools as shown in Figure 5. It interfaces directly to some tools such as SNORT and Wireshark. In other cases, the functions needed for general analysis happen to directly overlap with the other tools. NTE provides an environment where performance, visualization, statistical analysis, session analysis, and protocol analysis functions can all exchange data on an unencumbered basis.

The general architecture of NTE is shown in Figure 6 and provides these functions:

- Interface, manage, control and exchange data with a series of external tools.
- Place information in a series of arrays that follow a common storage philosophy.
- Analyze data from a variety of intrusion detection models
- Present information in text and graphical form to communicate the findings.
- Support analyst activity with a central management framework to hold the surrounding functionality together.

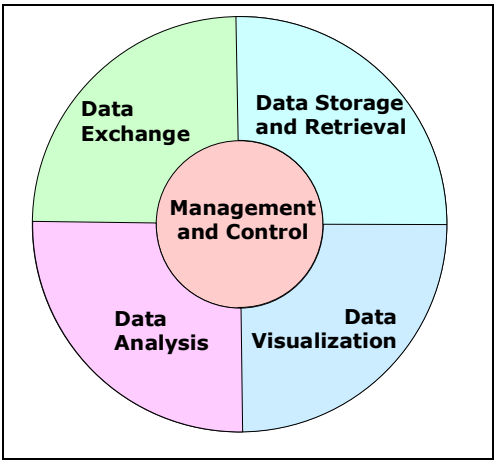

Fig. 6. NTE Architecture

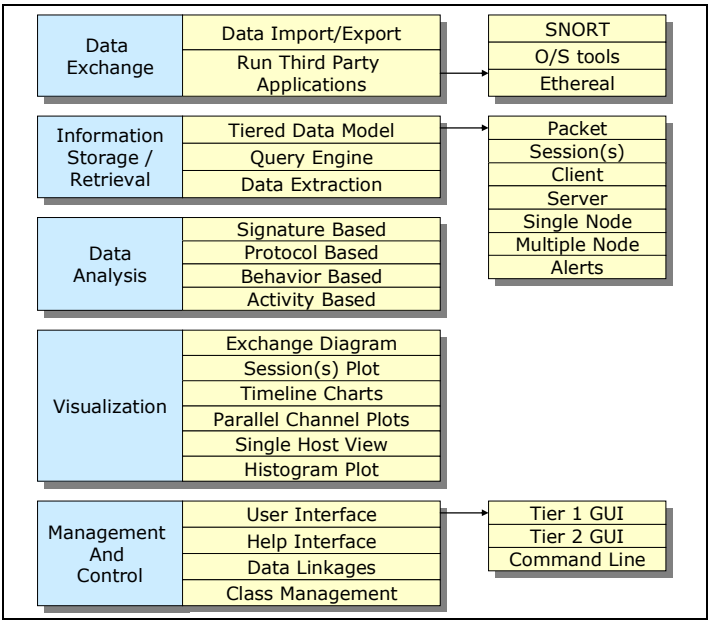

Fig. 7. NTE High Level Diagram

Figure 7 expands on each of the previously described functional blocks. The following subsections describe the functions in this figure in depth.

DATA EXCHANGE: NTE interfaces with a series of tools including Ethereal, SNORT, ping, traceroute, nslookup. NTE also reads the libpcap [15] file format and the fast and full SNORT alert data formats. Within NTE, there is a series of linkage

functions for the comparison or manipulation of data structures. Comparisons can be made between different data types. These features allow the operator to associate SNORT alerts to packets, sessions, or all packets communicated between a pair of nodes. The operator can also correlate, merge, as well as split information from multiple traffic loggers that may be skewed in time or distorted by a non-passive networking device such as a router, or VPN tunnel.

DATA STORAGE AND RETRIEVAL: The information received by NTE is stored in memory in a series of arrays that are similar to database tables. The tables summarize network or SNORT-based alert information into a relatively compact form that can be searched using a string-based query. Individual columns of the tables can be exported to external tools for analysis.

DATA ANALYSIS: NTE includes a wide variety of investigation and detection functions to provide:

- Checks for specific protocol attributes and behaviors
- Analysis of session-based communications
- Analysis of all packets exchanged between two end points
- Protocol profiling and session clustering functions to look for network traffic behaviors that do not conform to the norm
- Activity and usage behavior statistics for clients and servers
- Incoming and outgoing communication activity associated with a single node

VISUALIZATION: At every stage of the analysis, the NTE offers either text or graphical-based feedback. There is a diversified set of graphs and visualization functions. The ability to see an attack signature is useful when planning or developing countermeasures to known types of attacks.

MANAGEMENT AND CONTROL: A wide variety of support, management and control functions are provided. There is a standard model for adding GUI interfaces. The model allows a user to access internal functions through a complete GUI environment, or a command line environment. Each data input field is assigned a class type which allows the tool to know which variables on your desktop fit a given function parameter field. The NTE tracks a user's activity and recommends potential analysis options.

3 Environment

Many security products have a closed environment. Most vendors do not allow end users to add their own anomaly detection algorithms to a product. In contrast, the NTE tool is an open and flexible environment. It has several hundred functions for helping an analyst recognize network-based abnormalities, attacks and intrusions. These functions were written in a way that allows the analyst to "surf" through large volumes of data and display the results in a graphical window. By providing a suite of visualization functions, the end user can understand the nature of an attack and the validity of a given security event.

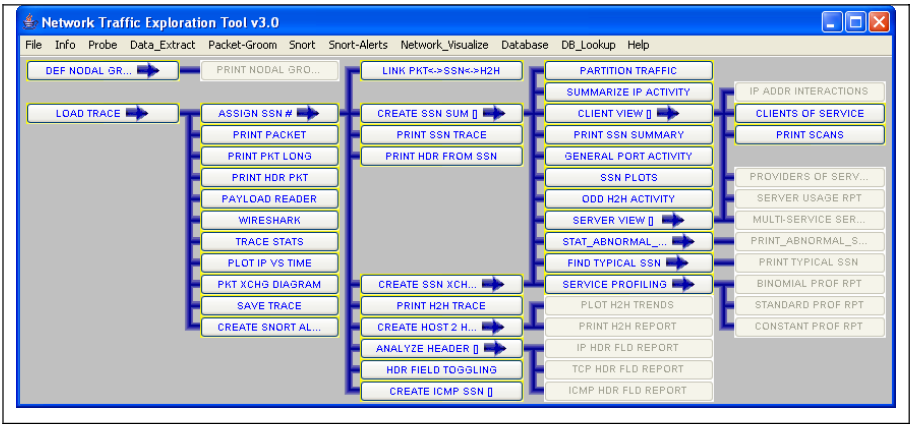

Fig. 8. NTE User Interface

NTE does not require that the analyst learn the hundreds of function calls. To facilitate use, an interactive GUI environment was created as shown in Figure 8. The buttons on the GUI show the options available to the user to solve a given problem. It is important to note that the GUI environment is optional and the user can enter or leave it at will. If there is a test or function that is not available, the user can swap back and forth between the GUI and the MATLAB command line environment without losing environment context or desktop variables.

To create scripts using the NTE the analyst must indicate the general approach to be taken in analyzing the problem by pressing the various GUI buttons. As the buttons are pressed, the tool builds the actual working program in the background. When the user is finished, the script can be saved to disk or displayed to screen for verification. The custom scripts can be reused in subsequent analysis sessions by entering the script name or command line or pasted into larger programs.

4 Preliminary Evaluation of Covert Tunnel Detection Techniques

Covert tunnels or covert channels allow an attacker to communicate with another system through a means that it was not intended or designed to do [35]. The most common tunnels today are HTTP, but ICMP and DNS tunnels also exist [36-38]. To demonstrate the NTE, a series of HTTP and ICMP tunnel applications were downloaded from the Internet. Some of the signatures have been described [39-47] and the tools to detect some of these covert tunnels are described in [39, 43]. The emphasis here is to demonstrate the breadth of analysis and the way in which the tunnel can be spotted using the built in visualization functions of the NTE.

EXAMPLE 1 – ICMP TUNNEL: The ICMP protocol typically follows a query-response (ping) or is used to send a single packet error message. There are a variety of ways to detect an ICMP covert tunnel, namely:

- Checksum problems between query and response
- Unusual byte patterns in the payload
- Sequence numbering that increment in an unusual way
- Responses without queries
- Queries with multiple responses
- Unusual packet lengths
- Unusual timing characteristics (pings typically occur once a second)

The packet exchange for an ICMP covert tunnel application downloaded from the Internet [48] is shown in Figure 9. In this case, the number of arrows in each direction does not match, the sequence numbers are not incremented by one, the inter-packet timing is greater than the 1 second, and the packet length has an unusual size.

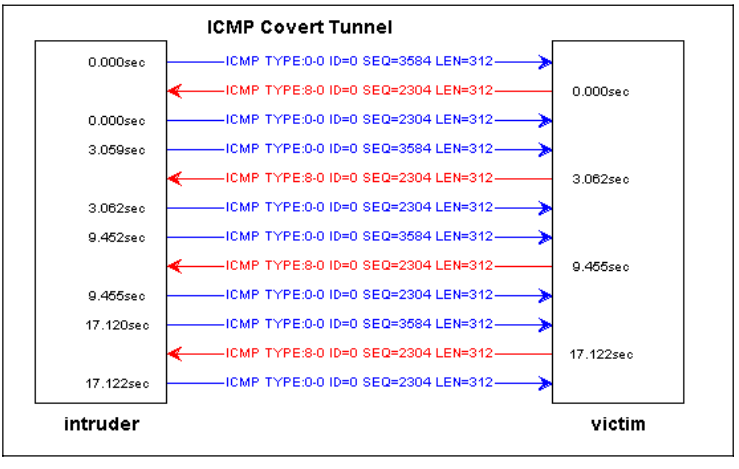

Fig. 9. Packet Exchange Diagram for a Sample ICMP Tunnel - Graph shows the packets exchanged between the attacker and the victim

EXAMPLE 2 – SPYWARE: Spyware is a form of covert tunnel that involves leakage of data to an outside source. This type of tunnel often uses the HTTP protocol and may have the following signatures:

- Repeated access to a number of HTTP sites
- Improper use of the HTTP protocol (binary data)
- Sequential access to a number of web sites on a fixed time interval
- Repeated session exchange patterns over multiple websites

An example of spyware activity is shown in Figure 10. This packet size versus time graph shows transmitted packets as a positive value and received packets as a negative value. Each unique communication sessions is shown in a different color. The graph shows that one computer is involved in repeatedly attempting to communicate with several targets. The variation of the timing characteristics are due

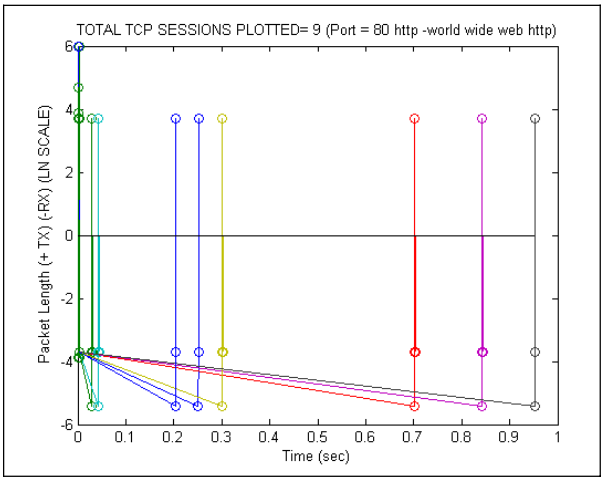

Fig. 10. Overlay Packet Exchange Diagram for a sample spyware - Data points above the black horizontal line represent data transmitted and points below are data received

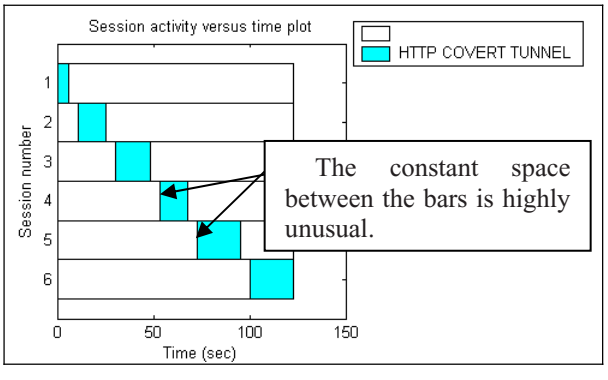

Fig. 11. GANTT Diagram for a sample In-band HTTP Tunnel

to network delays. Note how the packet exchange is exactly the same. Other visualizations show that communication is periodic as one host repeatedly tries to the same messages to three different websites (as the spyware attempts to connect to several home bases simultaneously).

EXAMPLE 3 – IN-BAND HTTP TUNNEL: There are several types of of HTTP tunnels. In-band tunnels hide their data in the payload while out-of-band tunnels communicate using the packet/protocol header. HTTP in-band tunnels are more difficult to detect because the HTTP protocol is very flexible. HTTP in-band tunnels are frequently machine driven, with multiple session exchanges showing little diversity in structure or timing characteristics. A sample HTTP tunnel [49] is shown in Figure 11. Each bar on this gantt chart represents an HTTP session. The tunnel

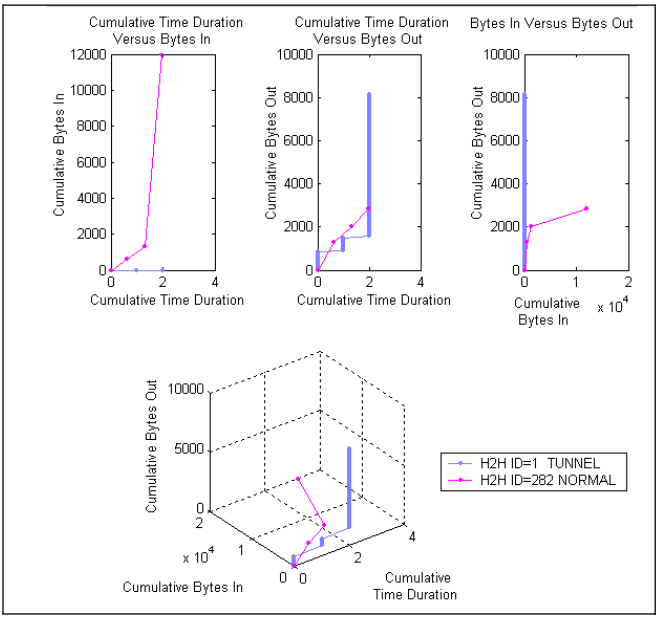

Fig. 12. Host-to-host Diagram for a sample out-of-band HTTP Tunnel

signature is in the constant space between the bars. Typically an end user does not wait X seconds from the end of one HTTP session before initiating the next.

EXAMPLE 4 – OUT-OF-BAND HTTP TUNNEL: HTTP out-of-band tunnels manipulate the packet header fields to send messages in one or both directions. In this example [50], the IP ID field is being manipulated to send commands in one direction (there is no payload, only a protocol header) and HTTP messages are used to send data back. The strange mix of incomplete and complete session patterns result in a very distinctive multi-session profile. Figure 12 shows the multi-session host-to-host graph with a covert tunnel and a single normal HTTP session for illustration. In this case, the tunnel looks like a step function while the regular traffic appears as a wandering line.

EXAMPLE 5 – AUTOMATED DETECTION USING NTE CODE: The signatures in the previous 4 examples were extracted manually by an analyst. In this example, the conversion of that experience to an automated response is demonstrated. Specifically, this example shows how code could be generated by the NTE to detect the out-of-band HTTP tunnel (see Figure 13). The code associated with plotting the data has been removed. The code (including comments) was generated as a standard part of the problem solving process and is executable as a standard MATLAB program. The program can be summarized as follows:

- Load a trace (load_pcap_file_plus)
- Enumerate the communication sessions (assign session number)

```
% Load the the PCAP trace array
[PKT_DATA,IP_OPT,TCP_OPT,PAYLOAD]=load_pcap_file_plus_ev(...
   'C:\DATA \good_bad_2.dmp','TRACE_NUMBER','','BPF','ip');

% Assign session id numbers and host-2-host id numbers
% to the trace array
PKT_IDX=assign_session_number(PKT_DATA,PAYLOAD);

% Create the host to host array
[H2H_SUM,H2H_LIST]=create_h2h_array(PKT_DATA,PKT_IDX);

% Print a summary of the host to host connections
[SSN_NUM,H2H_NUM]=print_host_to_host_details(H2H_SUM, ...
   H2H_LIST,'ALL',5,'TTL_PKTS=1&SERV_PORT=80&PROTOCOL=6',50);

% Convert host to host connections back to packets
[PKT_REF_NUM]=find_pkt_from_h2h(PKT_IDX,H2H_NUM);

% Create SNORT alert
[S_ALERT,S_CPAYLOAD,S_PAYLOAD]=create_snort_alert_ev(...
    PKT_DATA,PKT_REF_NUM(1),'GID',1,'SID',6969,'VER',1, ...
    'MSG','Out of Band Covert Tunnel','PRIORITY',1);

%Store the SNORT alert
save_snort_alert_ev('my_file',S_ALERT,S_CPAYLOAD,S_PAYLOAD);
```

Fig. 13. Code generated by the NTE during the problem solving process to spot an out-of-band HTTP tunnel

```
07/09-11:25:00.381048 [**] [1:6969:1] Out of Band
Covert Tunnel [**] [Classification: ] [Priority: 1]
{TCP} 192.168.6.3:1234 -> 192.168.6.2:80
```

Fig. 14. SNORT alert generated by the NTE

- Compile a list of all systems which spoke together (builds the host–to-host array)
- Search the intercommunication arrays for high numbers of unrequited communication sessions between two hosts that communicate using HTTP (print_host_to_host_details)
- Convert activity that passes this test into packets (find_pkt_from_h2h) (note if array is empty it is ignored)
- Convert packets to a SNORT alert which is saved to disk.

The function calls contained in the listing are standard NTE function calls. When this program is run, no GUI appears and the code outputs events to a standard SNORT alert file (see Figure 14). Note that the analyst can still modify this code and add extra features using either the NTA toolbox, NTE or using any of the MATLAB prepackaged function calls.

5 Discussion and Conclusion

The covert tunnel detection demonstration shows that NTE can handle a wide range of situations. The tool's broad range of graphs, reports, and visualization features offers a high level of insight into security events.

The tool allows an analyst to examine a particular type of attack and recognize signatures that distinguish it from routine traffic. Through the process of signature detection and analysis, it is capable of writing the code in the background necessary to detect future attacks or eliminate false positives.

NTE includes a wide range of functions and is easy to use and extend. The tool provides good analyst support via an interactive GUI, detailed user guide, individual help pages for functions as well as a unique tracking and guidance system. There is a detailed technical manual available to guide developers through each step of the function addition process. The tiered GUI system allows custom GUI front-ends for different users while maintaining a single backend.

NTE allows the user to store data in a desktop environment and port data easily between different functions. Linkage tools exist to facilitate comparison between different data types. The data query language allows the analyst to find data quickly and easily. The correlation functions allow the analyst to match multiple rows of data across multiple data sets.

Currently, NTE comes equipped with interfaces to external tools such as SNORT and Wireshark. The tool can read and write PCAP formatted files as well as SNORT's full and fast alerts. Future extensions could include fully automated event analysis and passive host-fingerprinting. These additions would allow an analyst to respond to network events faster and minimize repetitive analysis tasks.

References

[1] Valeur, F., et al.: A Comprehensive Approach to intrusion Detection Alert Correlation. IEEE Transactions on Dependable and Secure Computing 1(3), 146–149 (2004)

[2] Farshchi, J.: Statistical based approach to Intrusion Detection, SANS Institute(2003) (Access date 1 April 2008), http://www.sans.org/resources/idfaq/statistic_ids.php

[3] Roesch, M.P: SNORT (Access date 1 April 2008), http://www.snort.org/

[4] Ertoz, L., Eilerston, E. Lazarevic, A., Tan P. Srivastava, J. and Kumar, V.: Detection and Summarization of Novel Network Attacks Using Data Mining, Techincal Report (2003), http://www-users.cs.umn.edu/~aleks/MINDS/papers/raid03.pdf

[5] Chakchai, S.: A Survey of Network Traffic Monitoring and Analysis Tools, (2006) (Access date 1 April 2008), http://www.cse.wustl.edu/~cs5/567/traffic/index.html

[6] Ranum, M.: Packet Peekers, Information Security Magazine, p. 28 (2003)

[7] Keshav, T.: A Survey of Network Performance Monitoring Tools (2006)(Access date 1 April 2008), http://www.cs.wustl.edu/~jain/cse567-06/ftp/net_perf_monitors1.pdf

[8] Fortunato, T.: The Technology Firm, web page (2007), http://www.thetechfirm.com/reviews/

[9] Lyon, G.: Top 100 Security Tools, Insecure.org (2006), http://www.insecure.org/tools.html
[10] Bejtlich, R.: The Tao of Network Security Monitoring: Beyond Intrusion Detection, pp. 105–344. Addison-Wesley, Boston (2005)
[11] Vissher, R.: SGUIL (2007) (Access date 2 April 2008) , http://sguil.sourceforge.net/
[12] Combs, G., et al.: wireshark (2008) (Access date 2 April 2008), http://www.wireshark.org/
[13] Zalewski, M.: P0f (2006) (Access date 2 April 2008), http://lcamtuf.coredump.cx/p0f.shtml
[14] Elson, J.: tcpflow (2003) (Access date 2 April 2008), http://www.circlemud.org/~jelson/software/tcpflow
[15] Jacobson, V., et al.: Libpcap (2007) (Access date 2 April 2008), http://www.tcpdump.org/
[16] Jacobson, V., Leres, C., and McCanne, S.: tcpdump (2007) (Access date 2 April 2008), http://www.tcpdump.org/
[17] OPNET ACE Application Characterization Environment (2007) (Access date 2 April 2008), http://www.opnet.com/solutions/brochures/Ace.pdf
[18] Paxon, V.: BRO (2007) (Access date 2 April 2008), http://bro-ids.org/
[19] Computer Associates, eHealth (2008) (Access date 2 April 2008), http://www.ca.com/us/products/product.aspx?ID=5637
[20] Kohler, E.: ipsumdump (2006) (Access date 2 April 2008), http://www.cs.ucla.edu/~kohler/ipsumdump/
[21] Ritter, J.: ngrep (2006) (Access date 2 April 2008), http://ngrep.sourceforge.net/
[22] Combs, G., et al.: editcap/ mergecap (2008) (Access date 2 April 2008), http://www.wireshark.org/
[23] Astashonok, S.: Fprobe (2005) (Access date 2 April 2008), http://sourceforge.net/projects/fprobe
[24] Ostermann, S.: tcptrace (2003) (Access date 2 April 2008), http://www.tcptrace.org/
[25] Deri, L.: ntop (2008) (Access date 2 April 2008), http://www.ntop.org/
[26] Postel, J.: RFC 792 - Internet Control Message Protocol, (1981) (Access date 2 April 2008), http://www.faqs.org/rfcs/rfc792.html
[27] Kreibich, C.: netdude (2007) (Access date 2 April 2008), http://netdude.sourceforge.net/
[28] Fullmer, M.: flow-tools (2005) (Access date 2 April 2008), http://www.splintered.net/sw/flow-tools/docs/flow-tools.html
[29] Walkin, L.: ipcad (2007) (Access date 2 April 2008), http://sourceforge.net/projects/ipcad/
[30] Curry, J.: SANCP (2003) (Access date 2 April 2008), http://www.metre.net/sancp.html
[31] Kernen, T.: Traceroute (2008) (Access date 2 April 2008), http://www.traceroute.org/
[32] Fenner, B.: tcpslice (2002) (Access date 2 April 2008), http://sourceforge.net/projects/tcpslice/
[33] Buyllard, C.: Argus, (2008) (Access date 2 April 2008), http://www.qosient.com/argus

[34] Cho, K., Dittrich, D.: tcpdstat (2000), http://staff.washington.edu/dittrich/talks/core02/tools/tools.html
[35] Naval Research Laboratory, "Handbook for the Computer Security Certification of Trusted Systems", Technical Memorandum 5540, 062A (1996)
[36] Temmingh, R.: Setiri: Advances in Trojan Technology (2002) (Access date 2 April 2008), http://www.blackhat.com/presentations/bh-asia-02/Sensepost/bh-asia-02-sensepost.pdf
[37] Smith, J.: Covert Shells (2000) (Access date 2 April 2008), http://www.s0ftpj.org/docs/covert_shells.htm
[38] Kieltyka, P.: ICMP Shell (2002) (Access date 3 April 2008), http://sourceforge.net/projects/icmpshell
[39] Borders, K.: Web Tap: Detecting Covert Web Traffic. In: Proceedings of the 11th ACM conference on Computer and communications security, pp. 110–120. ACM, Washington (2004)
[40] Northcutt, S., Novak, J.: Network Intrusion Detection, An Analyst's Handbook, New Riders, Indianapolis, Indiana, pp. 63–65 (2000)
[41] Northcutt, S., Cooper, M., Fearnow, M., Fredrick, K.: Intrusion Signatures and Analysis, New Riders, Indianapolis, Indiana, p. 137 (2001)
[42] Knight, G., et al.: Detecting covert tunnels within the hypertext transfer protocol (2003), http://www.rmc.ca/academic/gradrech/abstracts/2003/ece2003-2_e.html
[43] Castro, S.: Covert Channel and Tunneling over the HTTP protocol Detection: GW implementation theoretical design (2003), http://www.infosecwriters.com/hhworld/cctde.html
[44] Dyatlov, A.: Exploitation of data streams authorized by a network access control system for arbitrary data transfers: tunneling and covert channels over HTTP protocol (2003) (Access date 2 April 2008), http://www.net-security.org/dl/articles/covertpaper.txt
[45] Feamster, N., Balazinska, M., Harfst, G., Balakrishnan, H., Karger, D.: Infranet: Circumventing Web Censorship and Surveillance. In: 11th USENIX Security Symposium, San Francisco, CA (2002)
[46] Crotti, M., Dusi, M., Gringoli, F., Salgarelli, L.: Detecting HTTP Tunnels with Statistical Mechanisms. In: ICC 2007. IEEE International Conference on Communications, pp. 6162–6168 (2007)
[47] Castro, S.: Cctde - Covert Channel and Tunneling Over the HTTP Protocol Detection (2003) (Access date 2 April 2008), http://gray-world.net/projects/papers/html/cctde.html
[48] Vecna. PacketStorm - 007Shell.tgz (1999) (Access date 2 April 2008), http://packetstormsecurity.org/groups/s0ftpj/
[49] Rowland, C.: Covert Channels in the TCP/IP Protocol Suite (1996) (Access date 2 April 2008), http://www.firstmonday.dk/issues/issue2_5/rowland/
[50] Hauser, V.: Reverse-WWW-Tunnel-Backdoor v1.6 (1998) (Access date 2 April 2008), http://packetstormsecurity.org/groups/thc/rwwwshell-1.6.perl