A Survey of Connection-Chains Detection Techniques

Ahmad Almulhem and Issa Traore
Electrical and Computer Enginerring Department, University of Victoria, CANADA
e-mails: {almulhem, itraore}@ece.uvic.ca

***Abstract*—A *connection-chain* is a set of connections created by sequentially logging into a series of hosts, known as *stepping-stones*. It provides an effective scheme for attackers to manually interact with a victim machine without disclosing their true origin. The victim will only identify the last host in the chain, while the true origin is hidden behind a series of stepping-stones. Addressing connection-chains poses challenges for researchers in the field of computer security. Accordingly, several approaches have been proposed in the literature. In this paper, we review those approaches and classify them according to a proposed taxonomy.**

I. Introduction

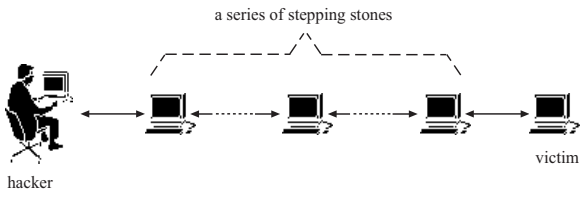

Fig. 1. Using a connection-chain to hide an attacker's origin.

The increase in hacking activities over the Internet is attributed to a number of factors. Probably, one important factor is the lack of *accountability*. Attackers have plenty of tricks and techniques that help them to stay *anonymous* during their attacks. A very effective anonymity technique is to indirectly attack a victim machine via a series of intermediary hosts; a scheme that is often called a *connection-chain* [1]. The chain is established by recursively logging into different hosts (known as *stepping-stones* [2]) before attacking the target machine as shown in figure 1.

In this paper, we survey and classify approaches for detecting connection-chains that have been proposed in the literature. In general, these approaches can be classified according to the location where the analysis takes place, into *network-based*, *host-based*, or *system-based*. Network-based approaches operate on packets at the network level; host-based approaches function inside hosts; system-based approaches employ both host-based and network-based components.

The rest of the review is outlined as follows. First, network-based approaches are discussed in section II. Then, host-based approaches are discussed in section III. Then, system-based approaches are discussed in section IV. Finally, we conclude the review and present possible open challenges in section V.

II. Network-Based Approaches

Network-based approaches operate at the network level by examining packets for signs of connection-chains. They are further divided into two main categories; namely *similarity-based* and *anomaly-based*. The difference lies at the scale of analysis. While, similarity-based approaches operate on a set of connections, anomaly-based approaches operate on only a single connection. Each category will now be explained in further details.

A. Similarity-Based

If two connections c_i and c_j belong to the same connection-chain C, then they are more likely to share some common *invariant* features. Therefore, one could devise a *similarity measure* to compare the connections, and flag similar ones as candidates for being part of the same connection-chain.

Two classes of similarity measures have been proposed in the literature: *content-based* measures and *timing-based* measures. A content-based measure computes similarity by analyzing the packets' contents (payload), whereas the timing-based measure computes similarity by analyzing the packets' timing characteristics.

If the connections are not encrypted, then a content-based measure works well. Basically, a character appearing in a given connection is assured to appear later in time in either a downstream or upstream connection of the same connection-chain. A naive measure is to simply perform a brute-force text matching between packets' contents [2]. Another simple measure is to trace *unique* strings, for instance a *login* greeting message [2]. Yet, another simple measure is to compute *frequencies* of characters traveling through connections [1].

In addition to introducing the connection-chain problem, Staniford-Chen and Heberlein are first to propose a content-based similarity measure, which they referred to as *thumbprint* [1]. In essence, a thumbprint is a real vector that is computed based on frequencies of characters traversing a connection during a specified time period . It serves as a condensed signature that can be used to differentiate (or relate) two connections.

Content-based similarity measures are limited, because they only work if the packets' contents are not encrypted nor modified as they flow through the connection-chain. A more general approach is to correlate connections based on the packets' timings instead of their contents. Collectively, such approaches employ *timing-based* similarity measures.

Zhang et al. proposed a simple yet effective timing-based measure [2]. The measure exploits the distinctive ON/OFF patterns observed in interactive traffic. Specifically, observing an interactive connection reveals a pattern of alternating ON and OFF periods. An ON period is when the user is typing on her keyboard, while an OFF period is when she is idle. The authors devised a similarity measure that computes coincident transition from OFF periods to ON periods among a set of connections. Using this measure, connections with similar transitions are correlated.

Deviation is another timing-based measure proposed by [3]. The measure relies on the following idea. As packets flow through a connection, the total size of transferred bytes tends to increase monotonically in time. Therefore, if two connections belong to the same connection-chain, then their total size of transferred bytes should grow at a similar rate. Obviously, this measure only works if the packet sizes are not altered at the stepping-stones. The authors formally developed this concept and used it to correlate connections.

Wang et al. proposed a timing-based measure that correlates connections based on the inter-packet delay (IPD) in forward flows [4]. In interactive traffic, IPD is a reflection of the typing dynamics of a user. Hence, the authors propose that they are unique and preserved through a connection-chain. They developed a similarity measure to compute and compare connections' IPDs.

Blum et al. proposed a timing-based measure that correlates connections by *counting* packets observed in a time interval [5]. They also showed how many packets are needed to declare whether two connections belong to a connection-chain or not.

He and Tong adapted a signal processing approach to detect connection chains [6]. In this approach, a connection is modeled as a *point process* [7], where the points represent the stream of packets in the given connection. Two connections are part of a connection chain, if their corresponding processes can be shown to exhibit a *casual* mapping (bijection). On the other hand, two connections are not part of a connection chain, if their corresponding processes are shown to be independent. To search for casual mappings between connections, the authors proposed two algorithms. One of the algorithms is timing-based, one that employs a delay constraint to search for possible mapping between incoming and outgoing packets. The other one uses a memory constraint to perform the same task.

Wang and Reeves proposed an active timing-based measure [8]. The idea is to embed a specially designed *watermark* into the flow of packets. If such watermark reappears later in another connection, then the two connections are part of the same connection-chain. The proposed watermark is essentially a modification of inter-packet timing between some selected packets. Peng et al. studied the secrecy of such watermarks and whether they can be detected [9]. They found out that embedded watermarks can be successfully recovered and duplicated if they are not designed carefully. Also, the existence of watermarks can always be quickly detected.

B. Anomaly-Based

In similarity-based approaches, the theme is comparing connections using some similarity measure. In contrast, anomaly-based approaches conform to a more *local* approach, where each connection is analyzed in isolation of other connections. Specifically, a connection's forward and backward flows are analyzed. The idea is that *direct* terminal sessions behave differently from *indirect* ones like those comprising connection-chains.

This novel approach was first proposed in [10]. He suggested measuring the following two time estimates:

- *Send-Ack time*: The time taken by a *send* packet to travel to the next host and get acknowledged. Basically, this time is an estimate of the *normal* roundtrip time exhibited by a direct terminal session.
- *Send-Echo time*: The roundtrip time for a *send* packet to reach the server side and get echoed back.

In a direct terminal session, the *Send-Ack* and *Send-Echo* times are expected to be similar. In an indirect terminal session (connection-chain), however, the *Send-Echo* time is expected to be larger than the *Send-Ack* time. In fact, the *Send-Echo* time becomes larger as the connection-chain becomes longer. There is, however, a catch. Matching a *send* packet with the corresponding *echo* packet can be tricky especially when encryption is used. Yung developed these concepts formally and provided a heuristic matching algorithm.

Yang et al. proposed another anomaly-based algorithm [11], [12]. Their approach suggests analyzing connections in *real-time* as a connection-chain is being established. They proposed several heuristic algorithms to match a *send* packet with its corresponding *echo* packet, in order to measure the *Send-Echo* time in real-time. When a new connection is appended to the connection-chain, the *Send-Echo* time *jumps* to a higher value. A plot of the *Send-Echo* time reveals a step-like function, where every step corresponds to a newly added connection. As such, a step-like behavior is indicative of a connection-chain.

III. Host-Based Approaches

In a typical network, a host usually has several inbound and outbound connections that correspond to listening servers and talking clients respectively. If the host is exploited as a stepping-stone, then there must be a correlation between some inbound and outbound connections. In other words, a packet arriving at an inbound connection is assured to reappear in an outbound connection.

Most operating systems, by default, do not have a function or a data structure that tells whether an outbound connection has been created by an inbound connection. As a result, one has to actually explore the operating system to find out if such a link exists. In literature, several techniques have been proposed to address this shortage.

One class of techniques employ a process *searching* algorithm based on the following concept. If an outbound connection c_o is created by an inbound connection c_i, then the

processes p_i attached to c_i and p_o attached to c_o are somehow linked. Depending on the operating system, the processes tree can be searched to discover if such link does exist.

Kang et al. proposed a simple search algorithm for a UNIX operating system [13]. Using the notations stated above, the algorithm operates as follows. If c_i and c_o are part of a connection-chain, then p_o (in many cases) is created (`fork`-ed) either directly or indirectly by p_i. Given the fact that in Unix, each process maintains a pointer to its parent process. Then, a simple way to link p_o to p_i is to start at p_o and *recursively* visit its parent process until p_i is found.

The above simple search algorithm fails if the link between p_o and p_i is more involved. For instance, this can be a result of using a pipe or other interprocess communication mechanism. Carrier and Shields proposed a more comprehensive search algorithm to resolve those cases [14]. For each p_o, they proposed walking up the processes tree, exploring a process's parent and all its siblings. They implemented the algorithm for three Unix-like operating systems; Linux, OpenBSD and Solaris.

Buchholz and Shields proposed a different approach, which does not require searching processes [15]. The approach calls for modifying an operating system to support linking an outbound connection to an inbound one. For each process, a new data structure `origin` is stored in its process table. For processes created by a remote connection, `origin` holds the typical 5-tuple information associated with that connection. For locally created processes, `origin` is undefined. When a process `forks` another one, `origin` is, as usual, inherited. The authors also proposed other supporting system calls and data structures.

IV. System-Based Approaches

In the literature, several system-based approaches have been proposed. In general, they employ an arrangement of collaborating components that together cooperate to detect and trace connection-chains. The components are both host-based and network-based.

One of the first proposed systems in this class is called *Distributed Intrusion Detection System* (DIDS) [16]. It consists of distributed host/LAN *monitors* and a centralized analysis module called the *director*. In essence, monitors collect auditing data and send them to the director for analysis.

DIDS has an interesting feature that enables tracing a user as she moves across a monitored network. The idea is to assign every user a unique network identification (NID) when she first logs into the monitored network. An NID is different from the typical user identification (UID). A user may have several UIDs for different hosts and resources, but only a unique NID. Accordingly, a user's activities (including logins) are associated with a single NID by the director. Based on its records, the director can then track a user's movement across the network.

In some respects, DIDS employs a *centralized* paradigm (the director) to trace connection-chains. In contrast, Jung et al. proposed a fully distributed system called *Caller Identification System* (CIS) [17]. The system requires installing two modules at each host: an extended version of *tcp-wrapper* ($ETCPW$) [18] and a CIS server ($CISS$). These modules interact locally and remotely using a distributed protocol to verify the origin of an inbound connection before allowing it in. Connections with inconsistent route information are denied.

Under CIS, a connection-chain $< h_0, \ldots, h_i, \ldots, h_n >$ is recursively traced as follows. When a new connection arrives at a host h_i, the local $ETCPW$ intercepts it and contacts the local $CISS$ to verify its origin. The local $CISS$, in turn, contacts the $CISS$ at host h_{i-1} requesting route information about the new connection. The remote $CISS$ replies with a list of the previous hosts in the chain; i.e. $< h_0, \ldots h_{i-2} >$. The local $CISS$ then contacts every host in the returned list to verify its integrity. If the integrity test is passed, it saves the list for future requests by the next hosts in the connection-chain; i.e., $< h_{i+1}, \ldots, h_n >$. It, finally, replies back to the local $ETCPW$ to allow the connection in.

Session Token Protocol (STOP) is another fully distributed system that allows to recursively trace connection-chains [14]. In essence, STOP is an enhanced version of the standard *Identification Protocol* (IDENT) [19]. It adds forensics and tracing functionality to IDENT in two essential ways. First, a STOP server is capable of saving user-level and application-level data associated with an outbound connection upon the request of a downstream host. The data is kept locally for future forensic investigation. Secondly, a request can be recursively propagated back to upstream hosts allowing tracing of connection-chains. This latter feature is somewhat similar to the recursive operation in CIS.

Wang et al. proposed another distributed system that calls for installing special modules at routers as well as modified servers at hosts [20]. This system employs an active approach. Basically, the servers (like `telnetd`) are modified to inject a *watermark* into backward flows upon request. Modules at the routers detect a watermark and respond appropriately. A watermark is a specially designed string of characters that depends on the modified server.

V. Conclusion and Open Directions

In this review, we surveyed and classified approaches for detecting connection-chains that have been proposed in the literature. We conclude this review by highlighting some possible directions in connection-chains research. Taking this review as a starting point, the following are some possibilities of where to go from here.

- Simulation Environment: Studying connection-chains would benefit from a powerful and flexible simulation environment. Existing simulation packages (for instance ns-2) need nontrivial modifications to simulate connection-chains. Along this direction, Xin et al. have recently described a promising testbed to simulate and evaluate connection-chains [21].
- Quantitative Comparative Study: A valuable study could be to conduct a quantitative comparative study of the proposed approaches. Such study requires a framework

where different approaches can be compared according to a set of quantitative criteria. In particular, similarity-based approaches are a good candidate for such study.

- Evasion: It is a possibility for a careful attacker to evade detection and/or tracing. Technically, these evasions are targeting the detection and/or tracing process itself. A valuable study would be to enumerate various scenarios under which proposed techniques can be evaded
- New Approaches: This area of research is relatively new. The review may inspire creating new approaches or combining existing ones. The solution space is quite large, and many ideas are still undiscovered.
- The Internet Model: The difficulty of tracing a connection-chain is related to its environment. Specifically, the difficulty is inversely proportional to the ability of controlling the hosts and the network. This progressive difficulty can be demonstrated with the following three reference models [22].
  1) *Closed Model*: Both hosts and network are under the control of a central authority.
  2) *Academic Model*: A central authority controls the network, but not the hosts.
  3) *Internet Model*: Neither the hosts nor the network are controlled by a central authority.

  It should be apparent that none of the proposed approaches applies to the Internet model, because they require control of hosts, network or both. As such, the ultimate challenge is to propose a solution for tracing connection-chains in the Internet model.
- Network Forensics: Tracing connection-chains is important for network forensics applications. In particular, it has the potential to help uncover compromised stepping-stones, the attack's path and even the attacker's origin. Therefore, tracing connection-chains could be projected as an infrastructure for a network forensics system.

REFERENCES

[1] S. Staniford-Chen and L. T. Heberlein, "Holding intruders accountable on the internet," in *Proceedings of IEEE Symposium on Security and Privacy*, May 1995, pp. 39–49.

[2] Y. Zhang and V. Paxson, "Detecting stepping stones," in *9th USENIX Security Symposium*, Aug 2000, pp. 171–184.

[3] K. Yoda and H. Etoh, "Finding a connection chain for tracing intruders," in *ESORICS '00: Proceedings of the 6th European Symposium on Research in Computer Security*. London, UK: Springer-Verlag, 2000, pp. 191–205.

[4] X. Wang, D. S. Reeves, and S. F. Wu, "Inter-packet delay based correlation for tracing encrypted connections through stepping stones," in *ESORICS '02: Proceedings of the 7th European Symposium on Research in Computer Security*. London, UK: Springer-Verlag, 2002, pp. 244–263.

[5] A. Blum, D. Song, and S. Venkataraman, "Detection of interactive stepping stones: Algorithms and confidence bounds," in *Lecture Notes in Computer Science*, vol. 3224, Jan 2004, pp. 258–277.

[6] T. He and L. Tong, "A signal processing prospective to stepping-stone detection," in *Conference on Information Sciences and Systems 2006 (CISS'06)*, Princeton, NJ, 2006.

[7] D. R. Cox and H. D. Miller, *The Theory of Stochastic Processes*. CRC Press, 1977.

[8] X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2003, pp. 20–29.

[9] P. Peng, P. Ning, and D. S. Reeves, "On the secrecy of timing-based active watermarking trace-back techniques." in *IEEE Symposium on Security and Privacy*, 2006, pp. 334–349.

[10] K. H. Yung, "Detecting long connection chains of interactive terminal sessions," in *RAID 2002, Lecture Notes in Computer Science*, vol. 2516, Jan 2002, pp. 1–16.

[11] J. Yang and S.-H. S. Huang, "A real-time algorithm to detect long connection chains of interactive terminal sessions," in *InfoSecu '04: Proceedings of the 3rd international conference on Information security*. New York, NY, USA: ACM Press, 2004, pp. 198–203.

[12] J. Yang and S.-H. Huang, "Matching tcp packets and its application to the detection of long connection chains on the internet," in *AINA 2005: 19th International Conference on Advanced Information Networking and Applications*, March 2005, pp. 1005–1010.

[13] H. W. Kang, S. J. Hong, and D. H. Lee, "Matching connection pairs," in *Lecture Notes in Computer Science*, vol. 3320, Jan 2004, pp. 642–649.

[14] B. Carrier and C. Shields, "The session token protocol for forensics and traceback," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 3, pp. 333–362, 2004.

[15] F. Buchholz and C. Shields, "Providing process origin information to aid in network traceback," in *Proceedings of the 2002 USENIX Annual Technical Conference*, 2002.

[16] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C. lin Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur, "DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype," in *Proceedings of the 14th National Computer Security Conference*, Washington, DC, 1991, pp. 167–176. [Online]. Available: citeseer.ist.psu.edu/snapp91dids.html

[17] H. T. Jung, H. L. Kim, Y. M. Seo, G. Choe, S. Min, and C. S. Kim, "Caller identification system in the internet environment," in *Proceedings of UNIX Security Symposium IV*, Santa Clara, California, oct 1993, pp. 69–78.

[18] W. Venema, "TCP WRAPPER: Network monitoring, access control and booby traps," in *Proceedings of the 3rd USENIX UNIX Security Symposium*. Baltimore, Maryland: USENIX Association, 14–16 Sep. 1992, pp. 85–92.

[19] M. S. Johns, *Identification Protocol*, RFC 1413, February 1993.

[20] X. Wang, D. S. Reeves, S. F. Wu, and J. Yuill, "Sleepy watermark tracing: an active network-based intrusion response framework," in *Sec '01: Proceedings of the 16th international conference on Information security: Trusted information*, 2001, pp. 369–384.

[21] J. Xin, L. Zhang, B. Aswegan, J. Dickerson, J. Dickerson, T. Daniels, and Y. Guan, "A testbed for evaluation and analysis of stepping stone attack attribution techniques," in *Proceedings of TridentCom 2006*, Barcelona, Spain, Mar 2006.

[22] T. E. Daniels and E. H. Spafford, "Network traffic tracking systems: folly in the large?" in *NSPW '00: Proceedings of the 2000 workshop on New security paradigms*. New York, NY, USA: ACM Press, 2000, pp. 119–124.