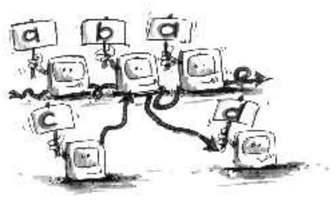
Connection-Chains: A Review and Taxonomy

Ahmad Almulhem and Issa Traore



ISOT Research Lab
www.isot.ece.uvic.ca

Electrical and Computer Engineering Department
University of Victoria, CANADA

December 2005

Technical Report ECE-05.4, 2005

University of Victoria

Connection-Chains: A Review and Taxonomy

Ahmad Almulhem and Issa Traore
ISOT Research Lab, ECE Department, University of Victoria, CANADA
{almulhem, itraore}@ece.uvic.ca

(Technical Report ECE-05.4, 2005)

***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.**

***Index Terms*—Connection chain, Stepping stone, Tracing, Traceback, Network forensics, Network security**

I. Introduction

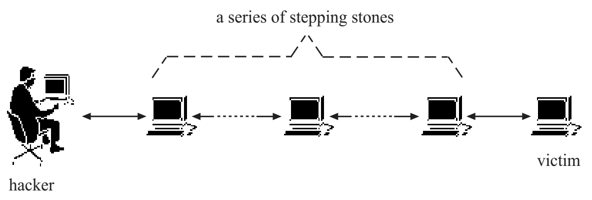


Fig. 1. Using a connection-chain to hide an attacker's origin.

The increase in hacking activities over the Internet is attributed to a number of factors. 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*. The chain is established by recursively logging into different hosts (known as *stepping-stones*) before attacking the target machine as shown in figure 1. Effectively, the connection-chain constitutes a channel that connects the attacker on one side with a victim machine on the other side. It gives the attacker an anonymous mean to manually interact with the victim machine without revealing the attacker's origin. The victim will only see packets coming from the last host in the chain, when in fact; the attack is hidden behind a list of possibly unrelated hosts.

Tracing connection-chains is a challenging yet important task for a number of applications. The following is a sample:

- Network Forensics: Tracing connection-chains plays a crucial role in network forensics applications. Particularly, it has the potential of revealing an attack's path as well as the involved hosts. Investigation then typically proceeds by isolating affected hosts and collecting data from them. Ideally, such tracing also leads to the origin of an attacker especially insiders. Coupled with collected evidences, the attacker can also be prosecuted in a court of law.
- Liability: If a host owned by an organization were exploited as a stepping-stone, the attack would appear to be originating from this organization. As a result, they may be held liable for such attack. Detecting connection-chains can help to enforce policies of transit traffic.
- Deterrence: Anonymity is a main concern of serious attackers. In fact, it is the whole purpose of establishing a connection-chain in the first place. An effective tracing tool will deter some attackers in fear of exposing their true origin.

Historically, connection-chains have been used repeatedly by attackers to hide their true origin. For instance, they have been used in a spy chase documented in the popular book: *the cuckoo's egg* [1]. Yet, Staniford-Chen and Heberlein are first to actually address the problem within a network security context and propose a solution [2]. They also coined the term *connection-chain*. The term *stepping-stones* was later coined by [3]. Since then, different approaches for detecting and tracing connection-chains have been proposed in the computer security literature.

In this review, we survey several approaches for detecting and tracing connection-chains. We also classify these approaches according to a proposed taxonomy. The review focuses on the *technical* issues. Specifically, we try to show how an approach works and highlight some shortcomings as well. There are some *non-technical* issues, for instance legal and societal ones. Such issues are not discussed here, but the interested reader is advised to reference the paper by [4].

The rest of the review is outlined as follows. In section II, we address some related and subtle issues surrounding connection-chains. In section III, relevant terminology and background material are presented. Then, taxonomy of connection-chains approaches is briefly described in section IV. The taxonomy is then explored in further details in the succeeding sections. First, network-based approaches are discussed in section V. Then, host-based approaches are discussed in section VI. At last, system-based approaches are discussed in section VII. In section VIII, we evaluate the reviewed approaches against a set of criteria. Finally, we conclude the review and present possible open challenges in section IX.

II. Related Issues

A. Anonymity: Connection-Chains Vs Spoofing

Serious attackers may have different motives, but definitely share a common concern; i.e. *anonymity*. They strive to hide their true origin during their attacks by employing different tricks and techniques. Generally, these techniques belong to one of two major schemes. One of them is using connection-chains, which is the focus of this review. The other one is to carry an attack using spoofed traffic; i.e. traffic where packets have forged source `ip` addresses.

From a tracing perspective, the settings of the two schemes are very different. As a result, researchers have treated them almost in parallel. *IP traceback* or just *traceback* [1] is coined for tracing spoofed traffic, while plain terms like *tracing* or *tracking* are used in the context of connection-chains. In what follows, we highlight some of the main differences between the two schemes.

- A connection-chain is established to create a bidirectional channel for an attacker to manually interact with a victim machine. As a result, every connection in the chain must have correct source and destination `ip` addresses. On the other hand, spoofed traffic is used for flooding denial of service attacks (DOS/DDOS). An attacker is not interested in receiving any traffic from the victim machine. Therefore, source `ip` addresses can be anything.
- In connection-chains, anonymity is achieved because a victim can only trace traffic to the last host in the chain. The true origin is laundered by several intermediary hosts. In spoofing attacks, however, an attacker stays anonymous because the packets' source `ip` addresses are fictitious.
- The type of traffic underlying the two schemes is very different. Connection-chains carry interactive traffic that reflects an attacker's typing dynamics. Therefore, the packets typically carry few (mostly one) bytes with timings of a typewriting human. On the other hand, spoofed traffic is generally massive and only limited by the dynamics of the network.

B. Legitimate Connection-Chains

Many users establish connection-chains on a daily basis either for convenience or necessity. A typical example is accessing restricted machines. These types of connection-chains are legitimate, because they conform to typical usage policies. The focus of this review, however, is the "bad" type of connection-chains. Distinguishing one from another is a subtle issue that may cause some confusion. In this section, we try to highlight what makes a connection-chain bad.

One distinguishing criteria is the *objective* of creating a connection-chain in the first place. As mentioned earlier, legitimate connection-chains are established for either convenience or necessity. On the other hand, illegitimate ones are established by someone to hide her true origin; i.e. to stay anonymous. Clearly, the objective is very different in the two cases. Unfortunately, there is no easy way to identify the objective of a connection-chain. However, the next criteria can help.

Another distinguishing criteria is the association of a connection-chain with an attack. A connection-chain by itself is not an attack. In fact, it is just a channel to carry an attack. Therefore, legitimate connection-chains should not carry any attacks. On the other hand, illegitimate ones should carry signs or signatures of some attacks. Accordingly, "bad" connection-chains are those that can be associated with an attack.

C. Progressive Difficulty

From a computer security perspective, connection-chains are troublesome since they are quite easy to establish and use for attacks. At the same time, they are hard to trace as they may span different autonomous systems (AS). Additionally, neither the standard TCP/IP suite nor standard operating systems adapt protocols or mechanisms to deal with them.

The difficulty level 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 [6].

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.

Using the above models, the challenges in tracing connection-chains progressively increase as one moves from one model to the next one.

III. Background Knowledge

A. Terminology and Definitions

In TCP/IP suite, applications like `telnet` [7], `rlogin` [8] and `ssh` [9] are used to log in a host and acquire a *virtual terminal* (or simply a *terminal*) on that host. The terminal (also called *console* or *shell*) is useful to execute commands and other programs *interactively*. For convenience, we refer to such applications as *terminal applications*.

If a user runs a terminal application on host h_0 to log into another host h_1, a terminal on host h_1 is obtained and a `tcp` *connection* [10] (or simply a *connection*) c_0 is established. The user then may use the terminal at host h_1 to log into another host h_2. This procedure may be repeated as many times as the user wishes [2] creating a series of connections as follows:

$$|h_0| \longleftarrow c_0 \longrightarrow |h_1| \longleftarrow \;\cdots\;\cdots\; \longrightarrow |h_{n-1}| \longleftarrow c_{n-1} \longrightarrow |h_n|$$

This series of connections is called a *connection-chain* [2], whereas the intermediary hosts are called *stepping-stones* [3].

Definition 3.1: A *connection-chain* C is a list of `tcp` connections created by recursively logging into a series of hosts; $C = \langle c_0, \dots, c_i, \dots, c_{n-1} \rangle$.

[1]For *IP traceback* approaches, the interested reader is referred to a review by [5]

[2]In practice, the number of hosts is limited by the maximum delay that a user is willing to experience [11].

Definition 3.2: Stepping-stones H are intermediary hosts that are used in establishing a connection-chain;
$H = \langle h_1, \ldots, h_i, \ldots, h_{n-1} \rangle$.

A related notion to consider is the *relative position* of the hosts and connections within a connection-chain. For this purpose, the following terms are defined.

Definition 3.3: In reference to a given host h_j, an *upstream host* h_i is a host that is closer to the *origin* host (h_0); i.e. $0 \leq i < j$. Conversely, a *downstream host* h_k is a host that is closer to the *target* host (h_n); i.e. $j < k \leq n$. The terms *upstream connection* and *downstream connection* are defined similarly.

Yet, another issue to consider is the notion of *directionality*. Although a connection is *bidirectional*, it has a direction in the sense of the *client/server* paradigm [12]. To distinguish each direction, the following definition is provided.

Definition 3.4: Each connection is made of a *forward flow* and a *backward flow*. The *forward flow* refers to the sequence of `tcp` packets sent by the *client-side* (or simply the *client*), while the *backward flow* refers to the sequence of `tcp` packets sent by the *server-side* (or simply the *server*).

At last, it is worth highlighting the following idea. In a connection-chain, each flowing packet has a *header* part and a possible *content* (or *payload*) part. The header part is unique for every single connection in the chain. It can be described by the following 5-tuple:

$$\langle src.ip, src.port, dest.ip, dest.port, protocol \rangle$$

The abbreviations src and $dest$ stand for *source* and *destination* respectively. They refer to the notion of directionality mentioned earlier.

Unlike the header part, the content part of a packet is relaid across the connections of a connection-chain. Therefore, it should remain the same during its journey through the chain. It, however, can be transformed in its passage depending on the used applications. A common transformation is encryption.

B. Dynamics of Terminal Applications

Terminal applications have distinctive *functionality* and *traffic pattern*, which set them apart from other applications. Functionality refers to the way that the client-side and the server-side interact during an established session. On the other hand, traffic pattern refers to the characteristics of generated packets as observed at the network-level, such as packets' sizes and inter-arrival times.

In terms of functionality, a terminal application exhibits a distinctive *send/echo* activity between its client-side and server-side. When a user establishes a terminal session and starts typing on her keyboard, the following types of packets are exchanged:

1) *Send packet*: The client *sends* every character as it is being typed by the user.
2) *Echo packet*: The server *echoes* back the sent character in order for the client to display it.

This routine is repeated for every typed character until the user hits the key "*return*" which causes executing the typed command. After the command's execution is finished, the server sends the command's output to the client.

In the case of a connection-chain, the above basic *send/echo* model is stretched over the entire chain. Effectively, the connections are equivalent to a single *logical* `tcp` connection where a client is located at one end and a server is located at the other end. The individual connections are glued by in-between stepping-stones. A *send* packet traverses every forward flow, while the corresponding *echo* packet traverses the backward flows.

To stitch the connection-chain, each stepping-stone runs two processes: a server and a client. The server accepts connections from an upstream stepping-stone (or the chain's origin), whereas the client connects to a downstream stepping-stone (or the chain's target). *Send* and *echo* packets are passed over between the two processes inside a stepping-stone. Effectively, *send* packets are pushed downstream towards the target, while the corresponding *echo* packets are pushed upstream towards the origin.

Moving on to the concept of *traffic pattern*, there are generally two classes of network traffic: *interactive* and *bulk transfer*. Terminal applications generate traffic that belongs to the former class. Essentially, the packets flowing in forward flows are dictated by the user's activity instead of the network dynamics. On the contrary, *bulk transfer* sessions (for instance, `ftp` [13]) are limited by factors like `tcp` flow control, *maximum transfer unit* (MTU), network congestion, etc.

In general, several metrics are used to characterize interactive traffic [14], [15]. For the purpose of studying connection-chains, the following metrics are considered: *packet size* and *packet timing*. Packet size refers to the size of the `tcp` payload in bytes, while packet timing refers to the characteristics of a packet's arrival/interarrival times.

If the connection is not encrypted (as in `telnet`), a send (and an echo) packet normally has a size of one byte that corresponds to a character typed by a user. With encryption (as in `ssh`), however, a send (and an echo) packet carries an encrypted version of the typed character. Hence, a packet's size depends on the encryption algorithm used. There are exception cases where few characters may be combined. For instance, `telnet` has a *line* mode where a client sends lines of text instead of individual characters.

For packets' timing, send packets in a forward flow are generated one by one as the user types on her keyboard. As a result, the packets' arrival (and inter-arrival) times are faithful reflections of her typing dynamics rather than the network dynamics. In particular, the inter-arrival times reflect how fast a user can type. There are empirical and statistical models that describe these times rigorously [14], [15]. Also, there are simulation tools to simulate them [16]. We refer the interested reader to the cited publications.

C. Model and Assumptions

Fortunately, it is not necessary to address the connection-chain problem in a general manner. In practice, there are several *domain knowledge* constraints, which are generally stated as either explicit or implicit assumptions. This section addresses these assumptions.

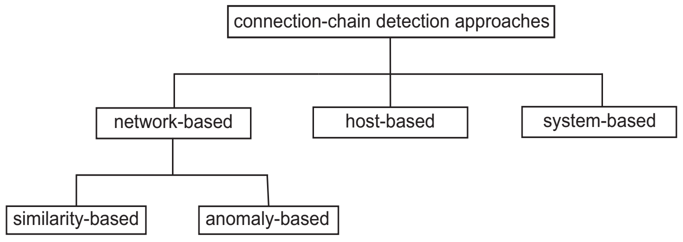


Fig. 2. Taxonomy of proposed approaches for detecting and tracing connection-chains.

In principle, connection-chains are solely used for *interactive* attacks. In other words, an attacker uses a connection-chain as a bidirectional channel to interact with a victim machine. This entails two main consequences. First, the `tcp` protocol is the transport protocol used in connection-chains. Secondly, the underlying traffic is interactive.

It is conceivable for an attacker to use another transport protocol or an unusual covert channel. This, however, requires a great deal of activities such as installing special modules on the stepping-stones. Potentially, such interactions are very loud and easier to detect and trace.

The above discussion leads to the following points:

- `tcp` [10] is the transport protocol employed in establishing connection-chains.
- A connection-chain can be modeled as a single logical TCP connection that connects a *human* agent at one side to a *victim* machine on the other side.
- Stepping-stones act as relay machines. They however may modify the relayed traffic. Possible modifications include encryption, embedding fictitious packets and adding random or intentional delay. Nonetheless, incoming traffic is related to outgoing one. This sets stepping-stones apart from other types of attack relays; namely *zombies* and *reflectors*.

IV. Taxonomy of Approaches

In the literature, many approaches have been proposed to detect and/or trace connection-chains. Those approaches integrate numerous interesting ideas and techniques. Before we start exploring them in details, we present a taxonomy that encompasses them. The taxonomy should render a “big picture” of the proposed approaches.

Generally, connection-chain approaches align with the taxonomy shown in figure 2. The top level of classification is based on the familiar *deployment* criteria. In particular, approaches are classified based on the *location* where analysis takes place. Accordingly, approaches that operate on packets at the network level are classified under *network-based* (section V). Also, approaches that function inside hosts are classified as *host-based* approaches (section VI). Finally, those approaches, which employ both host-based and network-based components, are classified as *system-based* approaches (section VII).

Noticeably, network-based approaches have received most of the research effort. Therefore, we further classify them into either *similarity-based* (section V-A) or *anomaly-based* (section V-B) approaches. In this case, the classification is based on the *nature* of analysis employed in each class.

V. 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 the larger scale of a set of connections, anomaly-based approaches operate on the smaller scale of 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 features. This is especially true for interactive traffic. 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. Such measure is typically a function of some *invariant* features that are relayed by the stepping-stones.

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 [3]. Another simple measure is to trace *unique* strings, for instance a *login* greeting message [3]. Yet, another simple measure is to compute *frequencies* of characters traveling through connections [2].

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* [2]. 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.

Regrettably, 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. In fact, the majority of connection-chain research belongs to this particular class.

Zhang et al. proposed a simple yet effective timing-based measure [3]. 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 such measure, connections with similar transitions are correlated.

Deviation is another timing-based measure proposed by [17]. 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 [18]. 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 [19]. 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 [20]. In this approach, a connection is modeled as a point process, 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 [21]. 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 [22]. 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.

At last, timing-based measures avoid using packets' contents, hence they can be used even if packets are encrypted. They, however, may fail if packets' timing characteristics (for instance arrival times) are disturbed either deliberately or due to the dynamics of the network and hosts.

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. In other words, connection-chain manifests a deviation from the *normal* direct terminal session.

This novel approach was first proposed in [23]. 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 gets 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 gets 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 [24], [25]. 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.

At last, it is worth mentioning that anomaly-based approaches are not suitable for tracing purposes. They are inspired as a tool to only detect connections that are part of a connection-chain.

VI. 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. In network-based approaches, such correlation is ascertained using different similarity measures (see section V-A). In host-based approaches, however, the story is different.

Ironically, all 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 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 [26]. 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 means. Carrier and Shields proposed a more comprehensive search algorithm to resolve those cases [27]. 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 [28]. 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.

At last, it is worth mentioning that host-based approaches are useful to detect stepping-stones with high accuracy. However, they are not useful by themselves to trace connection-chains. To do so, such approaches have to be employed within a system in order to recursively reveal the whole chain. Also, host-based approaches suffer from an obvious drawback. Specifically, they rely on the *integrity* of the stepping-stones. Such trust can not be established because stepping-stones are compromised hosts by definition.

VII. 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) [29]. 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 in 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) [30]. The system requires installing two modules at each host: an extended version of *tcp-wrapper* ($ETCPW$) [31] 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 [27]. In essence, STOP is an enhanced version of the standard *Identification Protocol* (IDENT) [32]. 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 connection-chains. This latter feature is somehow 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 [33]. 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.

At last, system-based approaches are projected as a comprehensive solution. They are also meant to attain the best of two worlds; network-based and host-based. They are however expected to be more costly in terms of installation, operation and maintenance.

VIII. Assessment of Approaches

We have reviewed several connection-chain approaches and classify them into several categories. In this section, we evaluate those approaches against a set of criteria. Particularly, the following criteria are considered:

- Domain: The different approaches are applicable to certain domains but not to others. For concrete treatment, we assess their applicability against the following three reference network models suggested in [6] (see section II-C): *closed model*, *academic model* and *Internet model*. As indicated earlier, the challenges progressively increase as one moves from one model to the next one. Hence, ideally, a technique should work in the most general model; i.e., the Internet model.
- Scalability: An important aspect of a given approach is its ability and flexibility to meet growth demands. We refer to such aspect as scalability, and evaluate the different approaches as *good*, *average* or *poor*. In principle, network-based approaches have better scalability than host-based ones. Additionally, a central entity is an indication of poor scalability.
- Tracing Ability: Some approaches are designed to only detect connection-chains. Others, however, are capable of tracing as well. In theory, a detection module is needed within a tracing system to initiate a tracing task. We use this criterion to indicate whether a given approach is designed for tracing. We use *yes* or *no* for that.
- Detection Accuracy: Detecting connection-chains is an important aspect of any approach. We use this criterion to rate the detection accuracy[3] for each approach. High accuracy is obviously a desired feature. The following scale is used in this rating: *high*, *average* and *low*.
- Evasion: This criterion refers to the ability of an attacker (who is possibly aware of the system) to escape detection and/or tracing. Ideally, an approach should be immune to evasion attempts. Depending on how hard it is to evade a given approach, the following scale is used: *hard*, *average* and *easy*.
- Cost: Different approaches incur varying costs. Such costs are the result of different factors such as installation, operation, and maintenance. This criterion is an estimate of the aggregate costs incurred by a given approach. The following scale is used: *high*, *average* and *low*.

Using the above criteria, the assessment is summarized in table I. We use descriptive labels such as high and low to establish a relative comparison between the different approaches. It is important to note that the comparison is largely subjective due to the lack of any quantitative comparison studies in the literature.

For the application domain, we notice that none of the proposed approaches applies to the Internet model. This is because all of them require installing special modules in the hosts and/or the network infrastructure. Such access is not granted in the Internet model. We also notice that network-based approaches are more general than other approaches, because they apply to the academic model for which controlling hosts is not required.

[3]Detection accuracy is is analogous to the *true-positive* rate in intrusion detection systems (IDS).

For scalability, network-based approaches obviously have an advantage. Host-based approaches are expected to have a poor scalability, because modifications are required for every added host. Yet, system-based approaches are expected to have an average scalability, since they employ both host-based and network-based components.

For tracing ability, we mentioned earlier that anomaly-based and host-based approaches can only detect a connection-chain. This is because they employ local analysis. On the other hand, tracing a connection-chain requires a global analysis to reveal the whole chain. Such ability exists in similarity-based and system-based approaches.

For detection, host-based modules are expected to produce more accurate results. Network-based modules are inherently less accurate, because they have to deal with issues such as encryption and delay. In fact, anomaly-based ones are expected to be even less accurate. This is because they rely on time estimates (for instance *Send-Ack*) that are hard to measure accurately.

The difficulty of evasion varies among the different approaches. System-based approaches are harder to evade, since they employ different distributed components. Therefore, if one component fails, there is a better chance that another component would resist the evasion. On the other hand, host-based approaches are easiest to evade, because the hosts are already compromised. Finally, network-based approaches are expected to be in between.

In terms of cost, system-based approaches are expected to incur the highest cost, because of the different distributed components involved. Host-based approaches come next, as the cost is expected to be per host. Network-based approaches are expected to come last, since the cost is associated with the network infrastructure.

IX. Conclusion and Open Directions

In this review, we surveyed several approaches for detecting and tracing connection-chains. We also classified them according to a proposed taxonomy. The review focused on the *technical* issues. Specifically, we showed how an approach works and highlighted some shortcomings as well. We also assessed the different approaches against a set of criteria. The review also included relevant background materials and various discussions.

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 [34].
- Quantitative Comparative Study: A valuable study could be conducting a quantitative comparative study of the

Criteria	Network-based (similarity-based)	Network-based (anomaly-based)	Host-based	System-based
Domain	academic	academic	closed	closed
Scalability	good	good	poor	average
Tracing Ability	yes	no	no	yes
Detection Accuracy	average	low	high	high
Evasion	average	average	easy	hard
Cost	low	low	average	high

TABLE I
ASSESSMENT OF THE DIFFERENT CONNECTION-CHAIN APPROACHES.

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: We mentioned earlier that none of the proposed approaches applies to the Internet model. The ultimate challenge is to propose a solution for tracing connection-chains in this model.
- Network Forensics: Tracing connection-chains is important for network forensics applications. In particular, it has the potential to help uncovering 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] C. Stoll, *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*. Doubleday Publishing, 1989.

[2] 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.

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

[4] S. C. Lee and C. Shields, "Tracing the source of network attack: A technical, legal and societal problem," in *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, june 2001, pp. 239–246.

[5] A. Belenky and N. Ansari, "On ip traceback," *IEEE Communications Magazine*, vol. 41, no. 7, pp. 142–153, 2003.

[6] 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.

[7] J. Postel and J. Reynolds, *Telnet Protocol Specification*, RFC 854, May 1983.

[8] B. Kantor, *BSD Rlogin*, RFC 1282, Dec 1991.

[9] C. Lonvick, *SSH Protocol Architecture*, Cisco Systems, Inc., December 2004.

[10] J. Postel, *Transmission Control Protocol*, RFC 793, sep 1981.

[11] D. L. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford, "Multiscale stepping-stone detection: Detecting pairs of jittered interactive streams by exploiting maximum tolerable delay," in *RAID 2002: Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection,*, october 2002, pp. 17–35.

[12] R. W. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1994.

[13] J. Postel and J. Reynolds, *File Transfer Protocol*, RFC 959, Oct 1985.

[14] P. Danzig, S. Jamin, R. Caceres, D. Mitzel, and D. Estrin, "An empirical workload model for driving wide-area tcp/ip network simulations," *Internetworking: Research and Experience*, vol. 3, no. 1, pp. 1 – 26, 1992.

[15] V. Paxson and S. Floyd, "Wide area traffic: the failure of poisson modeling," *IEEE/ACM Trans. Netw.*, vol. 3, no. 3, pp. 226–244, 1995.

[16] P. Danzig and S. Jamin, "tcplib: A library of tcp internetwork traffic characteristic," Computer Science Department, University of Southern California, Tech. Rep. CS-SYS-91-01, 1991.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.