

Architecture for Multi-Stage Network Attack Traceback

W. Timothy Strayer, Christine E. Jones, Beverly I. Schwartz,
Joanne Mikkelson, and Carl Livadas

*BBN Technologies
10 Moulton St.
Cambridge, MA 02138*

{strayer|cej|bschwartz|jmmikkel|clivadas}@bbn.com

Abstract

Attacks can originate from anywhere in the network but there is little the network can tell operators about where the attacker is located. Packet traceback techniques have been proposed to find the source of one or more IP packets, but some attackers use multiple remote login sessions, or stepping stones, to increase obfuscation. IP packet traceback can only find the source of one of the several connections in the stepping stone connection chain.

Stealthy Tracing Attackers Research Light TracE (STARLITE) is a customization and significant extension to BBN's Source Path Isolation Engine (SPIE.) The goal of STARLITE was to construct a prototype to integrate single packet traceback with stepping stone detection. The resulting prototype traces a packet to an ingress router, and then discovers if the flow of that packet is related to a flow in another connection. A successful correlation can then be continued until an ultimate source is located.

1. Introduction

It is well known that the Internet is vulnerable to malicious attacks. In spite of the development of many sophisticated defense mechanisms such as intrusion detection systems and firewalls, attacks continue to increase [1,2]. This is due in part because of a lack of accountability: The anonymous nature of the Internet and its protocols makes it difficult to accurately identify the source of a network attack when the perpetrator wishes to conceal it. In fact, an attacker can generate attacks that appear to have originated from anywhere or nowhere.

Attack tracing is a matter of determining the source of the attack packets and, when those attack packets are generated as a direct result of some application level activity, tracing back across those applications. This effectively divides the tracing problem into two parts: (1) finding the source of a flow of attack packets, or IP traceback, and (2) discovering which sources are acting to launder the attack, or stepping stone correlation.

1.1. IP Traceback

The ability to identify the immediate source of packets is a necessary first step in identifying the originating source of an attack, and several schemes have been proposed [3-7]. Yet identifying a packet's source is complicated by both legitimate actions taken upon the packet by the routers as well as the always-present possibility of malicious actors along the packet's path.

IP packets may be modified during the forwarding process. In addition to the standard decrementing of the time to live (TTL) field and checksum recomputation, IP packets may be further changed by intermediate routers. Packet *transformation* may be the result of valid processing, router error, or malicious intent. Packet transforms resulting from errors or malicious behavior need only be traced to the point of transformation, since the transforming router either needs to be fixed or can be considered a co-conspirator. However, traceback systems should trace packets through valid transformations back to the source of the original packet. This is especially important since attackers may further conceal their identities by engineering attack packets to be legitimately transformed by network protocols.

Also, an IP traceback system must not be confounded by an attacker that subverts a router with the intent to confuse the tracing system. If any router along the path may be co-opted to assist in concealing a packet's source, it becomes obvious that a traceback system must attempt to discern not only the packet's source, but also its entire path through the network.

The result of a packet traceback is an *attack path*, as shown in Figure 1. This attack path consists of each router traversed by the packet on its journey from source to the victim. Multiple paths may be identified. This can happen for two primary reasons. First, a compromised router can fabricate trace information such that multiple (but incorrect) sources are named. Second, multiple indistinguishable packets may be injected into the network from different, but legitimate, sources, possibly to confound the traceback system.

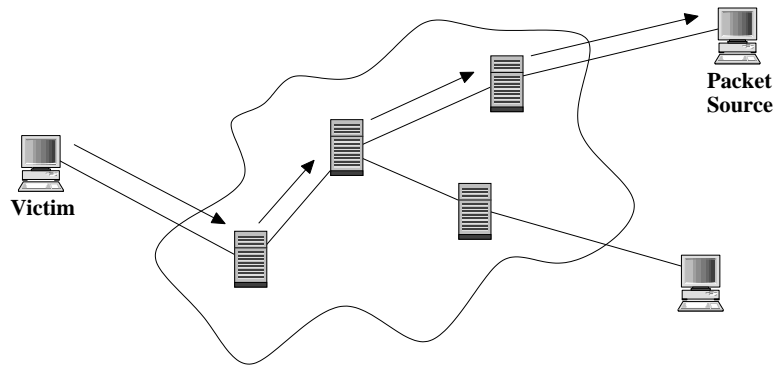


Figure 1—IP Traceback

In the presence of subverted routers, an attack graph may incorrectly identify a host as being the source of malicious packets; that is, an attack graph may contain *false positives*. This is an unavoidable consequence of admitting the possibility of subverted routers and tradeoffs between accuracy and resources. While attempting to minimize false positives, a traceback system must produce no *false negatives*; that is, it must never exonerate an attacker by not including the attacker in the attack graph.

1.2. Stepping Stone Correlation

IP traceback systems are oriented to packet tracing. They aim to produce an attack graph showing the path of one or more packets through the network. The resulting attack graph for a packet trace generally consists of routers that a packet traversed; the only hosts involved are the source of the packet and the victim. A traceback that reaches a host has identified a potential source of a packet. However, this does not mean that the source of the attack has necessarily been determined. The host may actually be a laundering host mid-stream of the actual attack path.

Tracing an attack path through laundering hosts, or *stepping stones*, requires the discovery of an association between two connections with endpoints at that host such that these connections act as consecutive links in a chain of connections. The chain of connections between an originating (or attack) source and the victim form what is called an *extended connection*, as shown in Figure 2.

An *extended connection* is constructed by a series of individual connections linked in a pairwise fashion and related by carrying the same content (plaintext or encrypted) serially across each connection from the originating source to the ultimate destination. A *connection pair* is any two adjacent connections in the extended connection—that is, of the four endpoints of the two connections, two endpoints are the same host.

Stepping stone detection techniques attempt to determine the correlation between all pairs of connections such that the pairs related by being part of the same extended connection will have correlation scores that are higher than those that are not related. Here are some of the notable algorithms found in the literature.

- Staniford-Chen and Heberlein [10] introduce the idea of *thumbprints* as short summaries of the contents calculated over a particular interval of connections, and use these thumbprints as the basis for correlating two connections at a stepping stone.
- Wang and Reeves [11] consider the inter-packet delays as the means of encoding a *watermark* signal into the packet stream that can be detected at points throughout the network.
- Zhang and Paxson [12] describe representing candidate data streams as a series of On/Off periods based on when connections go idle, and these periods are compared to find stepping stone pairs.
- Yoda and Etoh [13] describe a method to detect stepping stones based on the *deviation* between two connections, where deviation is defined as the difference between the average propagation delay and the minimum propagation delay between any two connections.
- Donoho et al. [14] present an approach that addresses directly the issue of countermeasures by assuming that the attacker has a *maximum delay tolerance*. They conclude that there are theoretical limits on the ability of attackers to disguise their traffic for sufficiently long connections.
- Blum *et al.* [15] also used maximum delay tolerance to develop a packet counting algorithm to provide bounds on the number of false positives.
- Strayer *et al.* [16] developed a correlation algorithm, called State Space, based on the probability that a

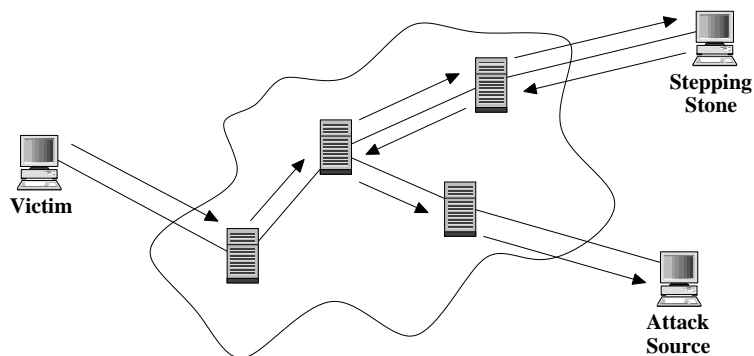


Figure 2—Trace across a stepping stone to create an extended connection

reaction (sent packet) on one connection was due to an incoming packet on another.

2. Integrated Attack Trace Architecture

The identification of the originating host of an attack requires being able to trace the attack back across not only the initial path of the attack packet or packets—giving the immediate source—but also back across any stepping stones that link several connections into the extended connection used to effect the attack. Since stepping stones link connections, which are often interactive, it is tempting to believe that stepping stone correlation algorithms alone are sufficient to trace a multi-stage attack back to its originating source. This is not necessarily true, largely due to two factors.

First, the source address of a connection is not always valid just because TCP connections require bidirectional communication. In general an attacker does not effectively use a connection, especially an interactive one, with a spoofed source address¹, but this does not mean that the network is preserving the source address. The source address is actively modified by the network when the source is part of a privately addressed network, as one connected to the Internet via a network address translator (NAT) [8]. In this case, the source address is an address from the non-routable private address space as defined by RFC 1918 [9]. The attacker establishes a connection from his private network through a NAT to a compromised system in the Internet, possibly also within a privately addressed or corporate network where the IP addresses are not routable or not known.

The second factor is that connection pairs are not always easily stitched together to form the fully reconstructed extended connection. Consider three connections between four hosts, $H_1 \leftrightarrow H_2$, $H_2 \leftrightarrow H_3$, and $H_3 \leftrightarrow H_4$. The pair ($H_1 \leftrightarrow H_2$, $H_2 \leftrightarrow H_3$) can be detected at some point where all traffic into and out of H_2 is seen. Likewise, the pair ($H_2 \leftrightarrow H_3$, $H_3 \leftrightarrow H_4$) is detected near H_3 . However, matching the two pairs into an extended connection assumes that, given one pair, the stitching algorithm knows where to find the other. The obvious clue is H_3 's address, but this suggests that there must be some global mapping of each host's address onto the data collection point handling the detection of stepping stones for that host. Outside of maintaining such a mapping, the easiest and most effective way to find the data collection point is if those points are closely tied to the routing infrastructure since the mapping of hosts onto routers is fairly well understood.

Integrating stepping stone detection with IP traceback addresses these two problems. A good IP traceback

¹ It is feasible for an attacker to spoof a connection; for instance, the response of the spoofed connection may transit the attacker, or the attacker may attempt to confuse the detection system itself. However, an IP traceback of a spoofed connection packet should identify the packet's (and thus the connection's) path, which includes the attacker.

system does not rely on the source address in the packet during traceback. Since a packet can be easily associated with a connection, tracing any one of the packets in a connection will find the origin of the connection, even if that origin is obscured in some way. Further, a good IP traceback system will assemble the entire path of a packet, specifically each router that a packet passed through. This implies that the traceback system is closely integrated with or somehow relies upon the routing infrastructure of the network.

The stepping stone detection techniques described in the literature to date, however, focus more on the effectiveness of the offered algorithm and much less on how that algorithm can be practically deployed in a network. Generally the algorithms are tested using traffic traces to measure their results against ground truth. They are rarely if ever placed in multiple locations in live networks to measure how well detected stones can be stitched together to trace back across the extended connection.

Closest to a deployable system, perhaps, is the Bro work by Paxson [17]. Bro is an intrusion detection system that passively monitors network traffic for certain types of events at various levels. One type of event Bro can look for is the presence of stepping stones based on the algorithm by Zhang and Paxson [12]. Bro uses policy scripts made up of event handlers that specify what to do whenever an interesting event occurs. These event handlers can draw from a rich set of reactions to events, from setting alerts to calling functions. From these building blocks it is possible to envisage a set of cooperating Bro installations that piece together the components of an extended connection, but this functionality is beyond the Bro distribution and would require considerable software construction.

3. STARLITE Traceback Architecture

BBN has integrated several of the stepping stone detection algorithms with SPIE to make an integrated multi-stage traceback system called STARLITE, or the Stealthy Tracing Attackers Research Light Trace.

3.1. IP Traceback using SPIE

An example of a good IP traceback system is the Source Path Isolation Engine (SPIE) [7]. SPIE is a log-based traceback system that uses efficient auditing techniques at network routers to support the traceback of individual IP packets. The tasks of packet auditing, query processing, and attack graph generation are dispersed among separate components in the SPIE system. Figure 3 shows the three major architectural components of the SPIE system.

The SPIE Traceback Manager (STM) controls the whole system, providing an interface for intrusion detection systems or other entities requesting a packet trace. When a request is presented to the STM, it verifies

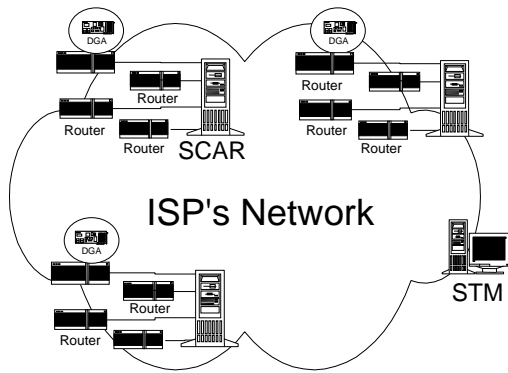


Figure 3—The SPIE Architecture

the authenticity of the request, dispatches the request to the appropriate SCARs/DGAs, gathers the resulting attack graphs, and assembles them into a complete attack graph. Upon completion of the traceback process, the STM replies to the requesting entity with the final attack graph.

SCARs, or *SPIE Collection and Reduction Agents*, are responsible for a particular region of the network, facilitating traceback of any packets that traverse the region. When a trace is requested by the STM, each SCAR produces an attack graph by querying the *Data Generation Agents* (DGAs) for the SPIE-enhanced routers in its region. A DGA produces packet digests of each packet as it is forwarded through a router, and stores the digests in time-stamped digest tables.

If a packet is determined to be offensive by some intrusion detection system (or judged interesting by some other metric), a trace request is dispatched to the system's STM. The initial trace request must provide the STM with a packet P , the victim V , and time of attack T . SPIE places two constraints on the intrusion detection system: the victim must be expressed in terms of the last-hop router, not the end host itself, and the attack packet must be identified in a timely fashion. The first requirement provides the query process with a starting point; the latter stems from the fact that traceback must be initiated before the appropriate packet logs are overwritten by the DGAs. This time constraint is directly related to the amount of resources dedicated to the storage of traffic digests.

A simulated reverse-path flooding algorithm is used by the STM to produce a final attack graph that indicates the packet's ingress router V' , the time T' at which the packet traversed the ingress router, and the packet P' (it may have been transformed, possibly multiple times) that actually traversed the ingress router.

3.2. Integration of Stepping Stone Detection

STARLITE integrates stepping stone detection into the SPIE model since SPIE already has a rich communication infrastructure. SPIE calls for DGAs to be

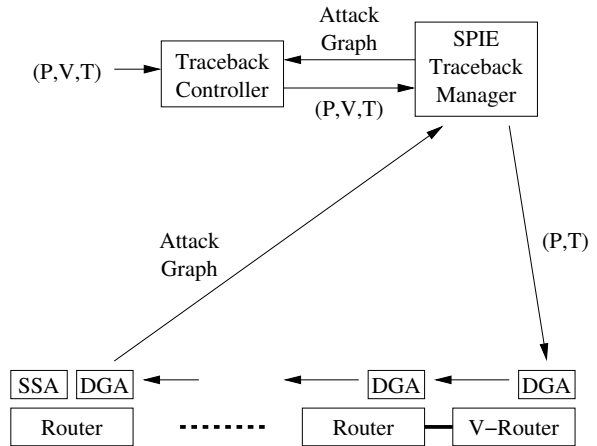


Figure 4—Initiating a multi-stage trace

deployed at every router. In STARLITE, some routers will also have stepping stone detector modules as well. These routers are stub routers, and we assume for now that hosts are not multi-homed. This allows us to use a single identifiable router for all incoming and outgoing traffic to a set of hosts. It means that the router serving a host acting as a stepping stone will see the traffic for both connections in a connection pair. Given that the router is the only sure point of commonality for the traffic, we assert that this is the appropriate location for stepping stone detectors. Further, given an IP traceback system that can determine the ingress router for a given packet, this stub router will be the ingress router for packets associated with the outgoing connection originating at the host behind the router. It will also be the last router to have seen the packets on the incoming connection and, therefore, be the natural starting place for a trace of packets associated with that incoming connection to determine that connection's ingress point (which possibly has evidence of another stepping stone, and, therefore, another connection to trace).

It is assumed, as in SPIE, that there exists some IDS or other entity that has captured a packet of interest, and thus initiates the trace by giving the traceback system a packet, the router nearest the victim, and the time the packet was seen. This is shown in Figure 4 as (P, V, T) being passed to the *Traceback Controller*, or TBC.

The TBC then employs the SPIE system to trace the packet starting at the router nearest the victim, the V-Router in Figure 4. SPIE traces this packet to the router nearest to the immediate source, and terminates, since this is the ingress router for the packet. The attack graph is returned to the STM, and the STM gives the graph to the TBC.

If the packet is not part of a connection, then it cannot be part of an extended connection as we have defined extended connection here. However, if the packet is part of a connection, then it may also be part of an extended connection, and the stepping stones module

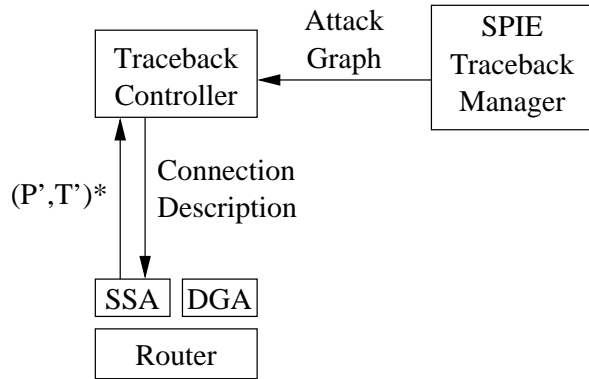


Figure 5—Querying for Stepping Stones

must be queried to see if the connection containing this packet is itself one part of a connection pair.

Consider any given packet that is part of a connection. It is easy to associate that packet with all other packets in the same connection by matching the source and destination IP addresses, ports, and protocol; all packets that exactly share these values are, for some period of time, packets within the same connection. We caution that this is for some period of time because connection identification material can be reused after it has been held unused for an amount of time far exceeding the lifetime of any packet within the network. Nonetheless, any packet that is part of a connection can be definitively identified as part of that connection. Figure 5 shows how the TBC takes the attack graph, determines the ingress router, extracts the connection description from the traced packet. The TBC then makes a query to the *Stepping Stones Aggregator* (SSA) to determine if the connection is part of an extended connection.

Since we assume that the hosts are not multi-homed, any connection that is part of a stepping stone connection pair must also have come through this router. The stepping stone detection algorithms (in the SSA), therefore, are deployed at or near these (and all other) routers which serve as the first hop for any hosts. The stepping stone detection algorithms produce a table of connection pairs such that, given an outgoing connection, the table indicates the correlated incoming connection. In order to continue the trace across the stepping stone, STARLITE must query this connection pair table to determine the next connection to trace.

Since the SPIE trace already has a representative packet within the outgoing connection, it is easy to look up the outgoing connection in the table. The table returns the corresponding incoming connection. But SPIE requires a specific example of a packet to conduct the trace; it cannot construct such a packet given only the connection identification material (IP addresses, ports, and protocol). Therefore, we must add another field in the connection pair table—a representative packet. Now,

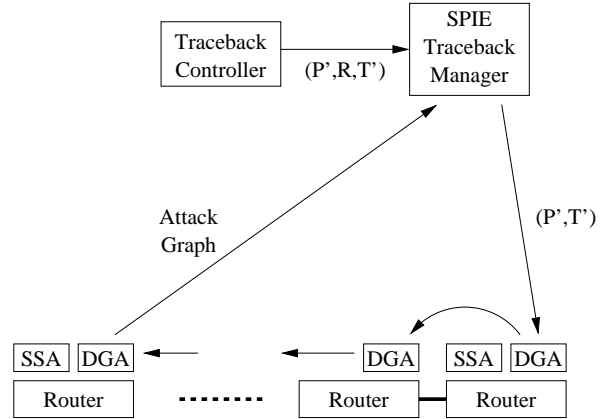


Figure 6—The trace continues by employing SPIE again

when SPIE consults the connection pair table, the result will be the incoming connection and one packet from the connection. The new packet, P' , and the time that packet was seen, T' , are returned to the TBC.

Now the TBC returns to the STM and asks SPIE to construct a new query based on the packet from the incoming connection, and continue the trace, as shown in Figure 6. Recall that SPIE requires three pieces of information for a trace: the packet, the time the packet was seen, and the last router known to have seen the packet. The last router known to have seen the packet is the current router since this must be the last router used by the incoming connection.

Since SPIE is sensitive about the age of the packets it traces—the older the packet, the less likely evidence of the packet's path is still kept in the SPIE DGAs—the packet that gets installed into the connection pair table should be the last packet seen on the incoming connection, along with the time that packet was seen. This gives SPIE the best chance of completing the trace with the incoming connection's representative packet since it is the newest packet on that connection.

SPIE then conducts a normal trace to once again find the ingress router for the connection. Again, the connection pair table is consulted, and additional packet traces may be launched (even tracing through NAT boxes, if necessary), until at some point the table does not report a correlated connection. At this point, the trace terminates with the last ingress router, which is as far as SPIE can go. However, since the traced packet is part of a connection and, therefore, must have a valid source address that can be used for returning packets such as acknowledgments, the host that is the actual originating source or the NAT box fronting for it can be easily determined.

4. Prototype System

The SPIE traceback system [7] has been shown to perform effectively as reported in previous work;

therefore, the successful performance of the STARLITE system is largely dependent on the ability of SSAs within the system to accurately detect stepping stones.

The STARLITE architecture allows for the use of multiple stepping stone detection and correlation algorithms. There are three axes that define the space within which the algorithms fit: how much data is required for the algorithm to produce a usable result, whether the algorithm is content based or timing based, and how resistant the algorithm is to evasive techniques. In our prototype implementation, we chose On/Off, CLT (from Blum), and State Space because they are all reasonably efficient and tolerant of encrypted connections such as *ssh*. We also implemented Thumbprints, but it is content-based, so it doesn't perform well with encrypted connections.

4.1. Performance Results

BBN performed extensive experimentation and evaluation of the four stepping stone detection algorithms—On/Off, CLT, State Space, and Thumbprints—selected to be integrated within the STARLITE prototype system. These extensive experiment results are reported separately in [18]. We provide here a brief discussion of one of the experiment scenarios that demonstrates the general performance of the stepping stone detection algorithms.

An SSA was deployed at BBN's RSA SecureID authentication server. This server is used by BBN employees to securely access BBN's intranet from the public Internet by setting up stepping stone connections through the authentication server. This allowed us to test the performance of the stepping stone detection algorithms when correlating numerous concurrent stepping stone connection pairs.

The results presented here represent stepping stone traffic observed for roughly one half hour during the middle of a business day. There were 106 *ssh* connection halves that were active—that is, at least one packet was transmitted. (A connection half is either the upstream or downstream portion of a connection, which are independently correlated.) Of those connection halves, only 38 were part of a known stepping stone, resulting in a total of 19 stepping stone connection pairs. All of the stepping stone pairs were legitimate; none involved active evasion on the part of an attacker.

Table 1 presents the correlation results for the On/Off, CLT, and State Space algorithms in terms of the number of correctly identified stepping stone connection pairs and the number of false negatives and positives. The Thumbprints detection algorithm relies on connection content to accurately detect stepping stone pairs, and therefore, is not applicable to the encrypted *ssh* connections that comprise this experiment scenario.

The On/Off algorithm, which was run using the original authors' suggested default parameters, performed

Algorithm	Number of Correctly Detected Stepping Stone Pairs	Number of False Negatives	Number of False Positives
On/Off	15	4	0
CLT	5	14	0
State Space	18	1	21

Figure 7—Experiment Results

quite well, accurately detecting 15 out of the 19 stepping stone connection pairs. Four of the stepping stone connection pairs failed to be detected. Note however, that the On/Off algorithm did not produce a single false positive. Our experiments found that the On/Off algorithm effectively detects unmanipulated stepping stone pairs by comparing the ending time of connection idle periods. However, evasive techniques available to an attacker—timing manipulation and chaff insertion—alter the timing characteristics of a connection pair such that idle periods are not necessarily preserved. Consequently, the On/Off algorithm fails to detect manipulated stepping stone connections.

The CLT algorithm, which was run with a maximum tolerable delay of 5 packets and a maximum false positive rate of 5%, was largely unsuccessful at detecting the stepping stone connection pairs. The algorithm detected only 5 of the 19 stepping stone pairs. Although the algorithm provides proven guarantees with respect to false positive and negative rates, it fails to be a practical solution for detecting stepping stones because the algorithm assumes a one-to-one interleave (or exchange) of packets over stepping stone connections. The CLT algorithm is too sensitive to legitimate variations in interleave, let alone artificial increases in interleave due to manipulation on the part of an attacker.

The State Space algorithm, with a λ parameter of 1, detected the most stepping stone connection pairs—18 out of a possible 19. However, the algorithm also produced a large number of false positives. The number is not quite as bad as it seems. Each of the 21 false positives involved one of two non-interactive connections. These non-interactive connections transmitted a fairly constant flow of packets, possibly a bulk data transfer or something similar in nature. The State Space algorithm inaccurately correlates such connections (and connections to which attackers have introduced large and steady amounts of chaff traffic) to every other connection because as the number of packets increases on a connection the inter-connection delay between the connection and all others decreases. This is a weakness of the algorithm, but false positives due to large-flow, non-interactive connections can be identified and possibly filtered from the final correlation results.

In general, though, the State Space algorithm, which correlates connections by exponentially weighting the inter-connection delay between packets observed on connections, is the most promising of the stepping stone detection algorithms. This is because the algorithm is effected to a lesser degree to active evasion by an attacker than the other algorithms.

4.2. Discussion

Since the architectural approach considered a stepping stone detection system that comprises several algorithms, each with its own scoring values and mean, it was necessary to have one *master function* that monitors each of the individual algorithms and aggregates their results into a composite score. In this way, we could exploit the advantages and mitigate the disadvantages of the selected algorithms which make them more or less suitable for certain types of networks or network traffic.

Such a master function must aggregate the results of these algorithms into a single score to be used to determine the correlated pairs of connections passing through this detection point. No stepping stone detection algorithm can issue a result with complete certainty; there is some degree of probability associated with each answer. Further, each of these algorithms uses its own method for determining a likely connection pair. Some use thresholds, some use probabilities, and some make declarative statements. The master function must be able to assimilate these answers, give them the proper weight, and construct an accurate representation of the consensus.

In general, the master function has to handle the problem of combining values from two or more different scales, metrics, or semantics. For scales, it's fairly easy: All of the values must be normalized with each other. For example, adjust all algorithm results so that they fit on a scale of 0 to 100.

Some algorithms may use different metrics than just the probability that two connections are related via stepping stones. To normalize this, the master function would have to first convert one metric into another, so that they are both in the same metric space. For example, one may be a probability, the other a score of the number of positive indications over a period of time. The master function can convert the second to the first by knowing the total possible number of indications (something in the algorithm that states something positive about the relationship) and divide that into the total number of positive indications, turning the score into a percentage. This can be normalized with the probability and the two can be combined.

Semantic differences can be handled the same way if the meaning of the scores is similar. But consider one algorithm that measures a feature that shows up for some connections but not others, and another algorithm that measures those not picked up by the first. Comparing them is very hard. In fact, if one pair of connections is

really a connection pair (related via the stepping stone), one algorithm may score it very high and the other very low because of emphasis on a particular feature. The concrete example in stepping stones is correlation based on the "content" within the connections. Clearly content is all scrambled if encryption is used, so any algorithm measuring correlation of content such as the Thumbprints algorithm, is going to score very low for encrypted connections. But the two connections may in fact score high for a timing-based algorithm. So the master function needs to pick up clues from the connections to determine if the algorithm is even applicable, and discount those that aren't, while normalizing those that are.

Now comes the act of combining. Multiplying probabilities is the normal way to combine two independent events, but these are not events, nor are they independent. So the master function should average the scores somehow. Our experiments have shown that an actual average is probably sufficient.

While testing the correlation of the forward packet streams of the connections, our experiments showed that the reverse packet streams also have strong correlation. Since a connection's forward and reverse packets can easily be identified, we feel that correlation on *both* the forward path and the reverse path adds to the overall strength of the correlation result. Consequently, the master function should take this into consideration when identifying stepping stones.

The idea of using multiple stepping stone algorithms with a master function may, in fact, be a bit of a red herring. Our experiments [16] have shown that some algorithms consistently perform better than others, namely State Space and On/Off, and those that perform well do so under the same conditions (that is, are equally good or bad given the presence of chaff or jitter). Equally so, our experiments have shown that several of the algorithms perform quite poorly, and so would not be a positive influence on the result of the master function. It may well be the case that one algorithm is sufficient. This would be fortunate since each algorithm requires substantial amounts of memory.

5. Future Work

Although a considerable amount of effort has been focused on the detection of stepping stones, both for this work and in the research community as a whole, there is still a considerable amount of work needed in developing an accurate and reliable method of detecting stepping stone connection pairs.

BBN is currently working on a mathematical model for the State Space algorithm, and is analyzing this model to gain some insight into what makes algorithms more or less resilient against countermeasures like chaff and jitter. In addition to developing this model, we are also considering non-interactive stepping stone correlation, where timing cannot be used as a means for correlating

streams. Our goal is to develop a system that is able to trace across a wide variety of ways attackers obfuscate their attack path, and to do so with a provably high degree of confidence.

Acknowledgment

This material is based upon work supported by the Space and Naval Warfare Systems Center, San Diego, under contract N66001-04-C-6006. We particularly wish to thank Michael Oehler for his help and support.

References

- [1] Howard, J. D., "An Analysis of Security Incidents on the Internet, 1989 - 1995," PhD Thesis, Apr. 1997.
- [2] Lawrence A. Gordon, Martin P. Loeb, William Lucyshyn, and Robert Richardson, "2004 CSI/FBI Computer Crime and Security Survey," *Ninth Annual Report*, Computer Security Institute, Aug 2004.
- [3] S. Bellovin, "ICMP Traceback," *Message to the IETF ICMP Traceback WG*, <http://www.research.att.com/~smb>
- [4] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *Proc. ACM SIGCOMM '00*, August 2000.
- [5] H. Burch, B. Cheswick, "Tracing Anonymous Packets to Their Approximate Source," *Proceedings of USENIX LISA '00*, December 2000.
- [6] D. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *Proc. IEEE INFOCOM 2001*, April 2001.
- [7] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, B. Schwartz, S.T. Kent, and W.T. Strayer, "Single-Packet IP Traceback," *ACM/IEEE Transactions on Networks*, Dec 2002.
- [8] Egevang, K., and Francis, P., "The IP Network Address Translator," *RFC 1631*, May 1994.
- [9] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and Lear, E., "Address Allocation for Private Internets," *RFC 1918*, February 1996.
- [10] Staniford-Chen, S., and Heberlein, L. T., "Holding Intruders Accountable on the Internet," *Proceedings of the IEEE Symposium on Security and Privacy '95*, May 1995, pp. 39-49.
- [11] Wang, X., and Reeves, D. S., "Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Manipulation of Interpacket Delays," *Proceedings of the ACM Symposium on Computer and Communications Security (CCS)*, October 2003.
- [12] Zhang, Y., and Paxson, V., "Detecting Stepping Stones," *Proceedings of the USENIX Security Symposium '00*, August 2000, pp. 171-184.
- [13] Yoda, K., and Etoh, H., "Finding a Connection Chain for Tracing Intruders," *Proceedings of the European Symposium on Research in Computer Security*, October 2000, pp. 191-205.
- [14] Donoho, D. L., Flesia, A. G., Shankar, U., Paxson, V., Coit, J., and Staniford, S., "Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay," *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (Oct. 2002)*, pp. 17-35.
- [15] Avrim Blum, Dawn Song, and Shobha Venkataraman, "Detection of Interactive Stepping Stones: Algorithms and Confidence Bounds," *Proceedings of the Conference of Recent Advance in Intrusion Detection (RAID) 2004*.
- [16] W.T. Strayer, C.E. Jones, I. Castineyra, J. Levin, and R. Rosales Hain, "An Integrated Architecture for Attack Attribution," *BBN Report 8384*, Dec 31, 2003.
- [17] Paxson, V., "Bro: A System for Detecting Network Intruders in Real-time," *Proceedings of the USENIX Security Symposium*, January 1998, pp. 1-11.
- [18] C.E. Jones, J. Mikkelsen, B. Schwartz, C. Livadas, and W.T. Strayer, "STARLITE Experiment Results," *BBN Technical Memo 2010*, May 2005.