Evading stepping stone detection by using enough chaff

Michael Shell
Georgia Institute of Technology
someemail@somedomain.com

Homer Simpson
Twentieth Century Fox
homer@thesimpsons.com

James Kirk
and Montgomery Scott
Starfleet Academy
someemail@somedomain.com

***Abstract*—Bla**

I. Introduction

Malicious actors on the Internet frequently use chains of compromised hosts to relay their attack, in order to obtain access to restricted resources and to reduce the chance of being detected. These hosts, called **stepping-stones**, are used by the attacker as relay machines, to which they maintain access using tools such as SSH or telnet reference.

Accessing a server via multiple relayed TCP connections can make it harder to tell the intruder's geographical location, and enables attackers to hide behind a long interactive stepping-stone chain. Furthermore, it is often required to relay an attack via privileged hosts in a network that have access to restricted resources.

However, detecting that a host is used in a stepping-stone chain is a clear indication of malicious behaviour. If a stepping-stone intrusion can be detected during the attack stage, the connection can be terminated to interrupt the attack. Stepping-stone detection (SSD) primarily looks at network traffic, with many approaches aiming to identify potential correlation between two connections going from or to a particular host. To hide correlation between relayed packets, intruders can impose delays on packet transfer and inject additional *chaff* packets into the connections.

There are a number of approaches to detect stepping-stones, with the earliest one having been proposed by Zhang and Paxson in 2000 citation needed?. However like many intrusion attacks, stepping-stones are rare and there exists no public data representing real stepping-stone behaviour, and researchers have to rely on synthetic data. Some attempts have been made to create publicly available stepping-stone testbeds, but most researchers evaluate their SSD methods on self-provided data. The underlying generating implementations often differ significantly, which makes their direct comparison of the achieved results impossible. Additionally, we find that implemented evasive behaviours are often too simplistic and thus increase detection rates.

In this work, we provide an independent framework to generate data representing stepping-stone behaviour .... We use this data to provide an in-depth evaluation of current passive SSD methods and compare their results. Furthermore, we show that by inserting enough chaff perturbations in the right form, an intruder can evade all current SSD methods successfully

II. Background

When a person logs into one computer and from there logs into another computer and so on, we refer to the sequence of logins as a connection chain[18]. In an interactive stepping-stone attack, an attacker located at the origin host, which we call *host O* sends commands to and awaits their response from a target, *host T*, typically by using terminal emulation programs like TelNet or SSH. These commands and responses are proxied via a chain of one or more intermediary hosts, the stepping-stones, which we call *host* S_1, ..., S_N, such as depicted in Fig. insert image.

Stepping-stone detection (SSD) is a process of observing all incoming and outgoing connections in a network and determining which ones are parts of a connection chain. SSD typically either aims at classifying a host as a stepping stone if two connections involving this host appear to be correlated, or at classifying a connection as part of a chain if its behaviour differs from regular connections. In addition, several SSD methods aim to estimate the overall length of the stepping-stone chain to help tracing the intruder by following the connection chain. A traffic collection sensor is typically placed in the vicinity of the examined host to provide the necessary data.

insert citation identify another form of stepping-stones called *store-and-forward*, which transfer data within files in a non-interactive manner. Though harder to detect than interactive connections, this procedure limits the attackers ability to explore the target, which is why SSD research has been primarily concerned with interactive stepping-stones.

A. Packet delays

B. Chaff perturbations

III. Related work

1) Evasion tactics: Padhye et al. [11] in 2010 have proposed a packet buffering method to decorrelate packets in the input flow from the output flow. Using this technique and enough chaff packets, the authors generate a constant-rate flow that resembles multimedia streams such as Voice over IP. However, the authors only test their framework against watermark-based SSD methods.

Network and Distributed Systems Security (NDSS) Symposium 2020
23-26 February 2020, San Diego, CA, USA
ISBN 1-891562-61-4
https://dx.doi.org/10.14722/ndss.2020.23xxx
www.ndss-symposium.org

Yang et al. [20] have recently

A. Datasets

In 2006, Xin et al. [16] developed a standard test bed for stepping-stone detection, called *SST*. The main objectives in the development of SST were to enable a reproducible evaluation of stepping stone chain detection algorithms with easy configuration, management and operation. The tool allows for an arbitrary number of intermediate hosts and generates scripts to mimic interactive SSH and TelNet connections. To insert time delays and chaff perturbations, the authors modified the OpenSSH protocol on the intermediary hosts. Delays can be drawn from a uniform distribution while chaff can be drawn from a Poisson or Pareto distribution. To our knowledge, SST has only been used for evaluation by Zhang et al. [21], and is not available anymore. say something why evasion not sufficiently implemented

Another approach to use publicly available data comes from Houmansadr et al. insert citations. The authors use the well-known CAIDA 2016 anonymized data traces insert citation to evaulate different watermarking methods for SSD. To simulate stepping stone behaviour, packet delays and drops are imposed retroactively on selected connections using Laplace and Bernoulli distributions with different rates. While this procedure seems sufficient for the evaluation of watermarking methods, it falls short on simulating an actual connection chain and leaves out chaff perturbations.

Wang et al. [13] recently conducted an extensive survey of stepping stone intrusion detection. The authors group methods according to the respective methodology into

- content-thumbprint,
- time-thumbprint,
- packet counting,
- random-walk-based,
- cross-over packet-based,
- watermarking,
- network-based,
- and software-defined-networking-based,

but do not cover graph-based methods such as [5] or anomaly-based methods such as [2], which are increasing in popularity recently. The authors then proceed to explain the different methods and highlight their benefits and shortcomings. The authors discuss open problems, but do not provide a comparison of detection rates.

Shullich et al. [12] in 2011 also conducted a survey on stepping stone intrusion detection. The authors perform a similar grouping of methods, but also discuss related work in evasion tactics and test frameworks. The authors furthermore give an outlook on areas for future research, such as hacker motivation, the cardinality problem when correlating connection pairs, the difficulty of tracing back chains through firewalls, the lack of real-world data examples, or detection in covert channels or protocols such as UDP.

[1]

Stepping Stone Detection Techniques: Classification and State-of-the-Art, bad though

Metrics: A Study on the Performance Metrics forEvaluating Stepping Stone Detection (SSD) Stepping Stone Detection: Measuring the SSD Capability

IV. Dataset creation

Our goal is to simulate data that reflects the different aspects of interactive stepping-stone behaviour in a reproducable manner. For a fair and thorough evaluation, we want to cover different settings and interactions to incorporate enough variation in the data to highlight strengths and weaknesses of different SSD methods. Furthermore, we need to generate sufficient data to effectively train some of the included methods.

In order to consider all these factors, we rely on the virtualisation framework Docker. Docker enables us to script the repeated creation of steppings-stone chains within a virtual network in a scalable manner. while allowing network different settings.

write about Docker advantages in terms of speed and controllability

A. Simulating stepping stones with SSH-tunnels and Docker

We want to capture data not only from one interaction in a fixed stepping-stone chain, but from many interactions and chains with different settings. For that, we run multiple simulations, with each simulation establishing a stepping-stone chain and controlling the interactions between host O and host T. A simulation and the corresponding traffic capture is *red*described a capture-script.

A simulation begins with the start-up of the necessary containers and ends with their takedown. We represent host O, host T, and host $S_1, \dots, S_n$ with SSH-daemon containers. To establish a connection chain, we connect these containers via SSH-tunnels, with the first tunnel forwarding the required port from host O to host S_1, which is then forwarded to host S_2 by the second tunnel etc. One benefit of SSH-tunnels from an attackers perspective is that they do not just simply forward packets, but act as independent encrypted TCP-connections along with independent packet confirmations. Fig. ... depicts a packet transfer via an exemplary chain.

Traffic is captured both at host T and host S_n, which acts as the final stepping-stone in the chain and is most likely the target of a detection algorithm.

B. Simulating interactive SSH-traffic

In order to generate enough data instances representing interactive stepping stone behaviour, we automatised the communication between host O and host T. For each simulation, we generate a script which passes SSH-commands from host O to host T.

For script-based session creation, several measures have been taken to make them realistic. First, each session tries to mimic a real user's action. We compiled a command database which consists of common commands and their usage frequency, similar to [16].Commands are drawn randomly according to their usage frequency and concatenated to a script. Commands can either be atomic, such as ”ls-la” or ”pwd”, or compound. Compound commands need additional input such as the directory and name of a specific file that is transferred or input text for a ..... The content and sometimes length of these inputs as well as transferred files are randomised appropriately when a compound command is drawn.

To simulate human behaviour that is reacting to the response from host T, all commands are separating by *sleep*-commands for time t, which is drawn from a Cauchy-distribution. By using a Cauchy-distribution, we allow for a small number of large t-values that simulate longer thinking periods.

Scripts are of varying length and end once the *End*-command is drawn from the command catalogue.

show exemplary script.

1) Adding network congestion: Hosts in a stepping-stone chains can be separated by varying distances. Some may sit in the same LAN, while others may communicate via the Internet from distant geographical locations. The type of separation between two hosts influences the round-trip-time, bandwidth, and network reliability.

Docker communication takes place over virtual bridge networks, so the throughput is far higher and more reliable than in real-world networks. To retard the quality of the Docker network to realistic levels, we rely on the emulation tool Netem. Netem add reference is a Linux command line tool that allows users to artificially simulate network conditions such as high latency, low bandwidth, or packet corruption/drop in a flexible manner.

We apply Netem commands to the network interface of each container, which adds correlated delays to incoming and outgoing packets that are drawn from a normal distribution with mean μ, variance σ^2, and correlation ρ_1. We furthermore apply correlated packet loss and corruption drawn from a binomial distribution with probability p and correlation ρ_2. Lastly, we apply an overall limit B on the bandwidth of containers container network interfaces. bandwith limiting.

To allow for different types of host separation, we set the network settings and bandwidth limit for each host container individually before each simulation, and draw each of the given parameters from a suitable distribution. This allows for some hosts to experience very fast and reliable communication while others experience more congested network communication. (should I specify which one for each? Seems a bit much...) We store the set parameters along with the collected traffic for each simulation to include the effect of network congestion in the evaluation.

2) Adding delays: To increase detection difficulty as suggested by [11], we add transfer delays to packets forwarded by stepping stone hosts. This method, often called *jittering*, can destroy time-based watermarks in packet flows and help decrease observable correlation between two connections.

We add transfer delays to forward packets again by using NetEm. We draw delays for departing packets on a hosts from a uniform distribution, as suggested by add reference, covering the interval $[0, \delta_d]$, with zero packet correlation. The value of δ_d is fixed before each simulation and can be varied to allow for different degrees of packet jittering.

insert citation suggested to use long packet delays to decrease decorrelation between packet streams. However, this often leads to difficulties in the TCP-protocol due to the significant increase of packet reordering and response time-outs. We therefore set the maximum value for δ_d at $1s$. As we will see in Section ..., this is enough to render watermarking methods obsolete while flow decorrelation can be achieved more effectively by using chaff perturbations.

3) Adding chaff perturbation: We furthermore insert chaff packets to individual connections in the chain to increase detection difficulty citation?. These chaff packets do not contain actual information and act as noise to decorrelate connections in the chain. To add and filter out packets in a connection, we add additional ports in each SSH-tunnel that are not forwarded through the entire chain. We then use a NetCat client containers to send and receive packets on these additional ports in both directions. Figure ... depicts this setup.

The data sent by the client i consists of strings with random size x drawn from a Cauchy-distribution with mean xx_i, and is sent in intervals of random length δ_c drawn from a truncated paretonormal distribution with mean yy_i. Huang et al. [6] have pointed out that packet interarrivals in benign connections often follow a paretonormal distribution, so it seems to be a good choice to make the inserted chaff appear as benign traffic. By adjusting yy_i, we can control the rate of chaff sent. explain more

V. Selected approaches

Researchers have so far proposed two main approaches: passive monitoring and active perturbation. In the latter

A. Packet-correlation-based approaches

1) Correlating TCP/IP Packet contexts to detect stepping-stone intrusion 2011: Yang et al. [18] compare sequences of interarrival times in connection pairs to detect potential stepping-stone behaviour. For that, the context of a packet is defined as the packet interarrival times around that packet. The context of packets is extracted from each connection, and their respective contextual distance is estimated using the Pearson correlation. Packets with high correlation are defined as ‘matched’, and two connections are classified as relayed if the ratio of matched packets exceeds a threshold. The authors propose to only collect interarrival times from *Echo*-packets instead of *Send*-packets to resist evasion tactics such as chaff and jitter, as the sending of *Echo*-packets is subject to more constraints and less easy to manipulate.

The authors evaluate their results on connection pairs with up to 100% chaff ratio, with the model being able to successfully detect connection relays in all cases.

B. Neural networks

Similar to other areas in intrusion detection, researchers have recently begun to train artificial neural networks to identify stepping stones.

A notable initial example came from citation, who ... Performance of neural networks in stepping-stone intrusion detection 2008 however, the authors concluded that the achieved results did not improve existing detection rates. but no good results, better in Neural networks-based detection of stepping-stone intrusion

The authors present two networks to identify stepping stones. The first method uses eight packet variables of an individual packet as input to predict the number of stepping-stones.

The second method is based on sequences of RTTs. For this, a packet matching algorithm is used to compute RTTs, which are then fed as a fixed-length sequence into a feed-forward network to predict the downstream length. The network itself only contains one hidden layer and is relatively small. The results from the packet-based method are however inconclusive, and the RTT-based methods achieves good results only if RTTs are small, i.e. the stepping-stone chain is completely contained within one LAN-network.

A more recent example comes from citaton, who train a convolutional neural network to identify correlation between two simultaneous connections from the upstream and downstream interarrival times and packet sizes in each connection. The trained network is large, with over 200 input filters and consists of three convolutional and three feed-forward layers. The trained model was initially applied to a dataset of Tor-connections as well, where the authors achieved strong results. The authors achieve a 90% detection rate with 0.02% false positives.

C. RTT-based approaches

Another prominent approach to detect stepping stones is based on *Round-trip/times* (RTTs). The RTT of a connection is the time it takes for a packet to be sent to the receiver plus the time it takes for an acknowledgement of that packet to be received. For a normal connection, the measured RTTs should be centered closely around one value. However, since information is relayed over one or more hosts in a stepping stone chain, the assumption for RTT-based detection is that the responses from different hosts within the chain generate multiple RTTs. Observing multiple RTTs is therefore a clear indication of relaying behaviour.

Yang et al. [19], [17] and Huang et al. [7], [3], [8] both have proposed multiple approaches for estimating and employing RTTs for stepping stone detection. We have selected two papers that depict the state-of-the-art...

1) RTT-based Random Walk Approach to Detect Stepping-Stone Intrusion : This model by Yang et al. [19] combines packet-counting methods and RTT mining methods to improve detection results from [17].

A widely-used approach is to compare the number of incoming packets in one connection with the number of outgoing packets in another connection to determine if the pair represents a stepping stone relay. However, the insertation of chaff can separate these numbers substantially. To resist intruders evasion, the authors propose to use the number of round-trips in a connection to determine if the connection is being relayed. Packet pairs representing a round-trip for each connection are estimated using a combination of packet matching and clustering, and counted as N_{in} and N_{out}. The authors then claim that the value of $N_{in} - N_{out}$ is only bounded if the two connections are relayed.

2) Detecting Stepping-Stone Intruders by Identifying Crossover Packets in SSH Connections, 2016: This method by Huang et al. [7] improves the detection methods proposed by Ding et al. [3]. The authors target specifically relayed interactive SSH communication at the end of a connection chain. They build their detection model on the fact that in a long connection chain, the round-trip time of a packet may be longer than the intervals between two consecutive keystrokes. Normally after sending a request packet, a client will wait for the server response before sending another request. However, TCP/IP allows a client to send a limited number of packets to the server without having to wait for the response. In a long connectin chain, this will result in cross-overs between request and response, which causes the curve of sorted Upstream RTTs to rise more steeply than in a regular connection. A stepping stone is detected if the maximum increase in the curve exceeds a threshold. The authors do not state a universal threshold value and instead suggest a method to estimate the appropriate value for a given setting.

D. Anomaly-based approaches

Since the insertion of time delays and chaff perturbations is so far relatively successful at evading detection, two authors have proposed algorithms to detect these two activities in a connection as deviations from typical TCP-behaviour to indicate of suspicious behaviour.

E. Detecting Anomalies in Active Insider Stepping Stone Attacks, 2011

Crescenzo et al. [2] have proposed a assembly of three methods to detect time delays and chaff perturbations in a selected connection.

1) The response-time based method is targeted at detecting packet time-delays. It estimates the RTT of a connection, and then flags acknowledgement packets if their response-time exceeds (RTT+δ_{RT}). The authors claim that an attacker would have to impose delays on more than 70% of all packets to evade this method.
2) The edit-distance based method is targeted at detecting

F. Detecting Chaff Perturbation on Stepping-Stone Connections, 2011, and Detecting Stepping-Stones under the Influence of Packet Jittering, 2013

Huang et al. [6] proposed a method to detect chaff perturbations in individual connections based on their assessment that interarrival times in regular connections tend to follow a Pareto or Lognormal distribution whereas chaffed connections do not. To detect whether a connection contains chaff perturbations, the authors extract packet interarrival times and fit a probability distribution to the data using maximum likelihood estimation. Afterwards, the goodness-of-fit is tested using two statistical tests, which yield a *disagreement-of-fit score*. If this disagreement score exceeds a threshold, which was determined from a set of know regular connections, the connection is seen as being subject to chaff perturbations. The authors generate a small set of chaffed interactive SSH stepping-stone chains, where they achieve a 95% detection rate on connections which are subject to a 50% chaff ratio while retaining zero false positives. For lower chaff ratios, the detection rate decreases significantly.

A similar approach from the same authors [4] is directed towards the detection of packet jittering. However, instead of estimating a disagreement-of-fit score, the authors use the estimated distribution parameters as input to train a support vector machine.

G. Watermarking

Watermarking consists of two complementary processes: embedding the watermark and decoding the watermark. A watermark is simply an unique binary string.The process of embedding one bit of this string consists of changing some property, usually ... of a traffic flow such that the change represents a bit. Decoding the watermark involves capturing candidate flows that might match the water-marked flow and looking for the bits in the flow characteristics. The bits of the watermark should have enough redundancy to ensure that they are decoded correctly with high probability.

Flow watermarking is, as found by [9], usually not very robust against timing and chaff perturbation attacks. We have selected the approach developed by Wang et al. [14] to be included in our evaluation because the authors state at least some resistance against timing perturbations. The authors assume some limits to an adversary's timing perturbations, such as a bound on the delays. The authors divide a sequence of packets into two groups of length m, and match packets from both groups into pairs. Each pair interarrival time is then perturbed using a watermarking function in dependence of an overall perturbation value s. The introduced watermark is invisible for third-parties and can be decoded in packet sequences longer than 600 packets using $m = 12$ when $s = 400ms$. The authors achieve 100% TP with 0.5% FP and claim resistance against timing perturbations of up to $1.4s$.

H. Behaviour-based approaches

A different direction in the detection of stepping stones focuses more on the general communication behaviour of selected hosts rather than individual connections. Features include the timely correlation of connecting IP-address on a selected host or unusual paths of simultaneously existing connections within a computer network.

Emulating these features realistically in network traffic datasets is difficult, and there exist little research on how strongly actual attack behaviour influences these features. Since our dataset focuses only on individual connections and corresponding evasion, we are not able to include behaviour-based approaches in our evaluation.

VI. Evaluation methods

All of the above presented methods classify individual connections or pairs of connections as malicious or benign. True stepping stone connections are rare compared to benign ones, making their detection an imbalanced classification problem. As discussed by insert citation, an appropriate measure to evaluate SSD methods are false positive and false negative rates as well as the *Area-under-ROC-curve* (AUC) for threshold-based methods.

Category	Approach	TP	FP	Tested resi
Packet-correlation	Yang, 2011 [18]	100%	0%	Resistance to jitter
Neural networks	Nasr, 2018 [10]	90%	0.0002%	small ji
	Wu, 2010 [15]			50% ch
RTT-based	Yang, 2015 [19]			
	Huang, 2016 [7]	0.85%	5%	
Anomaly-based	Crescenzo [2]	99%	1%	
	Huang/Ding 2011/13 [6], [4]	95%	0%	
Watermarking	Wang, 2011 [14]	100%	0.5%	1.4s jit

TABLE I. Summary of included SSD-methods.

nodes	no evasion	delays	chaff	delays& chaff
1-node	10.000	10.000	10.000	10.000
4-node	10.000	10.000	10.000	10.000

TABLE II.

The false positive rate is defined as

$$\frac{\text{number of benign connections classified as malicious}}{\text{overall number of benign connections}},$$

while the false negative rate is defined as

$$\frac{\text{number of malicious connections classified as benign}}{\text{overall number of malicious connections}}.$$

Some methods additionally try to predict the length of the stepping stone chain.

Write that we grant each method enough packets

VII. Evaluation data

A. Stepping-stone data

We generate data as described in Section IV. We create one dataset using a stepping-stone chain with only one stepping-stone S_1, and one with a longer chain of four stepping-stones S_1, S_2, S_3, and S_4.

For each set, we subdivide the stepping-stone data furthermore according to the two included evasion tactics, transfer delays and chaff perturbations. We first capture data without either of these methods. We then capture data one time with added transfer delays with varying δ_d, and one time with added chaff perturbations of varying rate. Lastly, we capture data with delays and chaff added simultaneously, again with varying δ_d and rate.

Since the initialisation of the SSH-tunnels usually follows a distinct pattern that can be revealing and consequently boost detection rates for ML-based methods, we remove the first twenty seconds of the captured connections.

Table II depicts the number of connection pairs included in each dataset.

	Value	TP deviation from overall			
		C	D	E	H
RTT	10ms				
	70ms				
Packet loss	0%				
	5%				
Bandwidth	no limit				
	50 kb/s				

TABLE III. INFLUENCE OF NETWORK CONGESTION ON DETECTION RATES AT .... THE GIVEN PERCENTAGES ARE DESCRIBING THE CHANGE OF THE DETECTION RATE WHEN COMPARED TO THE OVERALL AVERAGE.

B. Benign data

We need to provide a realistic background of benign data that reflects the heterogeneous nature of regular network traffic, both for evaluation of false positives and for the training of ML-based methods.

First, we created a set of interactive SSH-connections that are held directly between the client and the server. We follow the same procedure as described in Section IV-B with the same randomised network congestion settings as described in IV-B1. We then proceed to pair connections by random and remove the first 20 seconds of each connections to ...

Secondly, we include real-world traffic traces, taken from the CAIDA 2018 dataset insert citation and description. To allow for a fair comparison, we only selected connections that contain at least 1000 packets. Again, we pair connections randomly and remove the first 20 seconds of each connection.

The reason for relying on two different

Must be larger than ... (which excludes ...), trimmed to same number of packets since the algorithm cannot wait until the connection terminates to alarm

C. Implementation of selected approaches

VIII. Results

A. Data without evasion tactics

First, we look at the detection rates for traffic from stepping-stones that did not use any evasive tactics.

B. Combination of delays and chaff

I could insert comparison of prediction accuracy on chain length

C. Influence of network congestion

References

[1] A. Almulhem and I. Traore, “A survey of connection-chains detection techniques,” in *2007 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*. IEEE, 2007, pp. 219–222.

[2] G. Di Crescenzo, A. Ghosh, A. Kampasi, R. Talpade, and Y. Zhang, “Detecting anomalies in active insider stepping stone attacks.” *JoWUA*, vol. 2, no. 1, pp. 103–120, 2011.

[3] W. Ding, M. J. Hausknecht, S.-H. S. Huang, and Z. Riggle, “Detecting stepping-stone intruders with long connection chains,” in *2009 Fifth International Conference on Information Assurance and Security*, vol. 2. IEEE, 2009, pp. 665–669.

[4] W. Ding, K. Le, and S.-H. S. Huang, “Detecting stepping-stones under the influence of packet jittering,” in *2013 9th International Conference on Information Assurance and Security (IAS)*. IEEE, 2013, pp. 31–36.

[5] M. Gamarra, S. Shetty, D. M. Nicol, O. Gonazlez, C. A. Kamhoua, and L. Njilla, “Analysis of stepping stone attacks in dynamic vulnerability graphs,” in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[6] S.-H. S. Huang and Y.-W. Kuo, “Detecting chaff perturbation on stepping-stone connection,” in *2011 IEEE 17th International Conference on Parallel and Distributed Systems*. IEEE, 2011, pp. 660–667.

[7] S.-H. S. Huang, H. Zhang, and M. Phay, “Detecting stepping-stone intruders by identifying crossover packets in ssh connections,” in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2016, pp. 1043–1050.

[8] S.-H. S. Huang, R. Lychev, and J. Yang, “Stepping-stone detection via request-response traffic analysis,” in *International Conference on Autonomic and Trusted Computing*. Springer, 2007, pp. 276–285.

[9] A. Iacovazzi and Y. Elovici, “Network flow watermarking: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 512–530, 2016.

[10] M. Nasr, A. Bahramali, and A. Houmansadr, “Deepcorr: Strong flow correlation attacks on tor using deep learning,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1962–1976.

[11] J. D. Padhye, K. Kothari, M. Venkateshaiah, and M. Wright, “Evading stepping-stone detection under the cloak of streaming media with sneak,” *Computer Networks*, vol. 54, no. 13, pp. 2310–2325, 2010.

[12] R. Shullich, J. Chu, P. Ji, and W. Chen, “A survey of research in stepping-stone detection,” *” International Journal of Electronic Commerce Studies”*, vol. 2, no. 2, pp. 103–126, 2011.

[13] L. Wang and J. Yang, “A research survey in stepping-stone intrusion detection,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, p. 276, 2018.

[14] X. Wang and D. Reeves, “Robust correlation of encrypted attack traffic through stepping stones by flow watermarking,” *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 3, pp. 434–449, 2010.

[15] H.-C. Wu and S.-H. S. Huang, “Neural networks-based detection of stepping-stone intrusion,” *expert systems with applications*, vol. 37, no. 2, pp. 1431–1437, 2010.

[16] J. Xin, L. Zhang, B. Aswegan, J. Dickerson, T. Daniels, and Y. Guan, “A testbed for evaluation and analysis of stepping stone attack attribution techniques,” in *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006*. IEEE, 2006, pp. 9–pp.

[17] J. Yang and S.-H. S. Huang, “Mining tcp/ip packets to detect stepping-stone intrusion,” *computers & security*, vol. 26, no. 7-8, pp. 479–484, 2007.

[18] J. Yang and D. Woolbright, “Correlating tcp/ip packet contexts to detect stepping-stone intrusion,” *Computers & Security*, vol. 30, no. 6-7, pp. 538–546, 2011.

[19] J. Yang and Y. Zhang, “Rtt-based random walk approach to detect stepping-stone intrusion,” in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*. IEEE, 2015, pp. 558–563.

[20] J. Yang, Y. Zhang, R. King, and T. Tolbert, “Sniffing and chaffing network traffic in stepping-stone intrusion detection,” in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. IEEE, 2018, pp. 515–520.

[21] L. Zhang, A. Persaud, A. Johnson, and Y. Guan, “Stepping stone attack attribution in non-cooperative ip networks,” in *Proc. of the 25th IEEE International Performance Computing and Communications Conference (IPCCC 2006)*, 2005.

Appendix