Evading stepping stone detection by using enough chaff perturbations

No Author Given

No Institute Given

Abstract.

1 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.

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 Staniford and Heberlein in 1995 citation needed?. 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 traffic generation implementations often differ significantly, which makes a 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 that represents realistic stepping-stone data. Our framework is scalable and capable of generating sufficient variety in terms of network settings and conducted activity. This allows us to generate a large and comprehensive dataset suitable for the training of ML-based methods and in-depth performance evaluation. We use this

data to provide comparison of the capabilities of eight state-of-the-art stepping-stone detection methods that we re-implemented. Furthermore, we show that by inserting enough chaff perturbations in the right form, an intruder can evade all current SSD methods successfully.

The rest of the paper is organised as following: Section 1 provides an introduction and background to the problem of stepping-stone detection, with Section 1.2 discussing related work. Section 2 discusses the particular design of the data generation framework. Section 3 discusses a eight different SSD methods that we selected and implemented for evaluation. Section 3.6 presents the dataset arrangement in terms of background and attack data and discusses evaluation methods. Section 4 discusses the results achieved by the implemented methods on the given data.

1.1 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. 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, SSH, or Netcat reference. These commands and responses are proxied via a chain of one or more intermediary hosts, the stepping-stones, which we call *host* $S_1, \ldots, S_N$, such as depicted in Fig. 1.

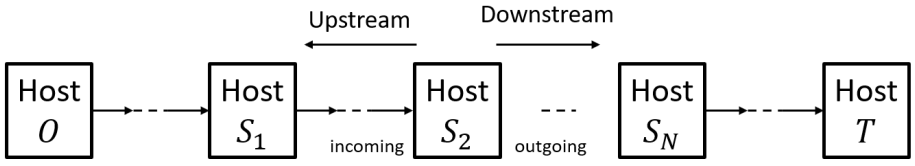

Fig. 1. Depiction of a stepping-stone chain.

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 it shows certain characteristic of relay behaviour. 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.

Coskun et al. [2] 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.

To avoid detection, several evasive flow transformation techniques exist that aim at decreasing observable correlation between two connections in a chain.

Packet transfer delays/drops An easy way for an attacker to destroy flow watermarks and create temporal disparity between connections is to apply artificial random delays to forwarded packets, often also called jitter. Similarly, an attacker can choose to not forward certain packets at all and cause retransmissions. Researchers often assume the existance of a maximum tolerable delay for an attacker.

Chaff perturbations Chaff packets are packets without meaningful content that are added to individual connections in a chain without being forwarded, and cannot be distinguished from real attack packets by a third party. Adding chaff perturbations alters the overall amount of traffic in a connection and can be used to shape the connection profile towards other traffic types.

Repacketisation Repacketisation is the practice of combining closely adjacent packets into a larger packet or splitting a packet into multiple smaller packets to alter observed packet sizes and numbers.

Flow splitting/merging Since SSD is mostly done on singular connections, an attacker can split the flow of packets to two or more connections and merge them at the target.

1.2 Related work

In 2006, Xin et al. [17] 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. The authors give little details about the implemented evasive tactics.

Another approach to use publicly available data comes from Houmansadr et al. insert citations. The authors use the well-known CAIDA anonymized data traces [1] 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 but do not cover graph-based methods such as [6], or anomaly-based methods such as [3].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 direct 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.

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.

2 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 us to emulate different network settings.

can we cite DetGen here?

2.1 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 in 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 $S_1, \ldots, S_n$ 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 and repacketisation. Fig. 2 depicts a packet transfer via an exemplary chain.

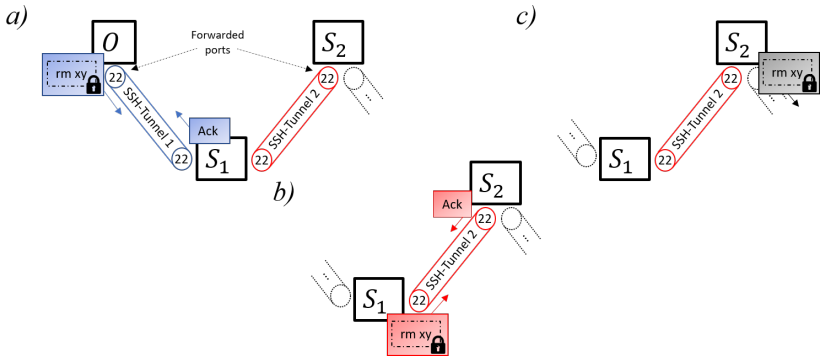

Fig. 2. Depiction of the way a command is packetised, encrypted, and travels through the different stages of the stepping-stone chain via SSH-tunnels.

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.

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 [17].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 to fill a file. 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 truncated 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.

Simulating different network settings 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 container network interfaces.

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.

2.2 Evasive tactics

Adding transfer delays To increase detection difficulty as suggested by [11], we add random transfer delays to individual 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 significantly longer packet delays of several seconds to request packets and their answers in order to decrease temporal correlation between active periods in two connections. However, this often leads

to difficulties in the TCP-protocol due to the significant increase of packet reordering and response time-outs. We set the maximum value for δ_d at 1500 ms. As we will see in Section 4, this is enough to render watermarking methods obsolete while flow decorrelation can be achieved more effectively by using chaff perturbations.

Adding chaff perturbation In addition to transfer delays, we insert chaff packets to individual connections in the chain to increase detection difficulty. These chaff packets do not contain actual information and act as noise to decorrelate individual connections in the chain. To add and filter packets in a connection, we open additional ports in each SSH-tunnel that are however not forwarded through the entire chain. This means that each chaff packet only appears in the connection it was inserted to, making the chaff perturbation in two different connections independent of each other. We then use a NetCat client containers to send and receive packets on the additional ports in both directions.

The data sent consists of strings with random size x drawn from a truncated Cauchy-distribution with mean μ_C to mimic video-streaming traffic as suggested by Padhye et al [11]. Similarly, packets are sent in intervals of random length δ_c drawn from a uniform distribution that covers the interval $[\delta_c/2, \delta_c]$ to mimic a relatively constant packet flow, typical for video-streaming. By adjusting δ_C, we can control the amount of chaff sent.

3 Selected SSD methods

The main contribution of this work is to offer a modern evaluation of the current state of SSD methods. A range of underlying techniques exist for SSD, and we try to include approaches from every area to create an informative overview and highlight strengths and weaknesses.

We surveyed publications to create a collection of SSD methods. We with the publications described in the surveys [12, 13] from 2018 and 2011, mentioned in Section 1.2. We expanded this collection via finding publications that cited these papers. Lastly, we found publications by browsing Google Scholar with different combinations of the keywords “connection”, “correlation” “stepping-stone”, “detection”, “attack”, “chaff perturbation”. Our final collection contained 60 publications.

From here, we selected approaches based on the following criteria:

1. Achieved detection rates,
2. robustness against evasion tactics,
3. improvements of previous work.

Below, we describe the selected methods. Table 1 contains a summary of the included methods. We labelled each method to make referring to it in the evaluation easier.

Category	Approach	TP	FP	resistance	Label
Packet-corr.	Yang, 2011 [19]	100%	0%	jitter/< 80% chaff	PContext
Neural networks	Nasr, 2018 [10]	90%	0.0002%	small jitter	DeepCorr
	Wu, 2010 [16]	100%	0%	-	WuNeur
RTT-based	Yang, 2015 [20]	not provided		50% chaff	RWalk
	Huang, 2016 [8]	85%	5%	-	Crossover
Anomaly-based	Crescenzo [3]	99%	1%	jitter/chaff	Ano1
	Huang 2011 [7, 5]	95%	0%	> 25% chaff/ > 0.2s jitter	Ano2
Watermarking	Wang, 2011 [14]	100%	0.5%	< 1.4s jitter	WM

Table 1. Summary of included SSD-methods along with the true positive and false positive rates and evasion resistance provided by the authors.

3.1 Packet-correlation-based approaches

Packet-correlation approaches attempt to identify correlations between two connections by identifying packets that appear in both connections. Since connections can be encrypted, this is often done by comparing sequences of interarrival times and packet sizes.

Correlating TCP/IP Packet contexts to detect stepping-stone intrusion, 2011 Yang et al. [19] 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.

DeepCorr: Strong Flow Correlation Attacks on TorUsing Deep Learning, 2018 A more recent example comes from Nasr et al. [10], who train a deep convolutional neural network to identify correlation between two 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.

3.2 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. Since information is relayed over one or more hosts in a stepping stone chain, this has an effect on the overall RTTs which can be observed within individual connections in the chain.

Neural networks-based detection of stepping-stone intrusion, 2010 A notable initial example of neural network applications to SSD came from Wu et al. [15], and which was later improved by the authors [15]. The designed nerual network model 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 of the chain. The network itself only contains one hidden layer and is relatively small. RTT-based methods achieves good results only if RTTs are small, i.e. the stepping-stone chain is completely contained within one LAN-network.

RTT-based Random Walk Approach to Detect Stepping-Stone Intrusion, 2015 This model by Yang et al. [20] combines packet-counting methods and RTT mining methods to improve detection results from [18]. 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 insertion 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.

Detecting Stepping-Stone Intruders by Identifying Crossover Packets in SSH Connections, 2016 This method by Huang et al. [8] improves the detection methods proposed by Ding et al. [4]. 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 connection 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.

3.3 Anomaly-based approaches

Two authors have proposed algorithms to detect the insertion of time delays and chaff perturbations in a connection as deviations from typical TCP-behaviour to indicate of suspicious behaviour.

Detecting Anomalies in Active Insider Stepping Stone Attacks, 2011 Crescenzo et al. [3] have proposed an 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 comparing packet sequences ON-OFF intervals in upstream and downstream in a connection, which should be similar to each other in a benign connection. The assumption is that chaff insertation is often independent in the two directions, leading to dissimilarities.
3. The causality-based method is verifying that the number of ON-intervals in upstream and downstream direction match each other to detect chaff.

Detecting Chaff Perturbation on Stepping-Stone Connections, 2011, Detecting Stepping-Stones under the Influence of Packet Jittering, 2013 Huang et al. [7] 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 [5] 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.

3.4 Watermarking

Watermarking is the only active SSD mechanism and 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 packet interarrivals, 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.

Robust correlation of encrypted attack traffic through stepping stones by flow watermarking, 2010 Flow watermarking is the most effective way at correlating connections with low false positives. However, 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 $1.4s$.

3.5 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.

3.6 Evaluation data and methods

The quality of both the stepping-stone and background data is crucial for a fair evaluation.

Different approaches require different amounts of packets to make predictions. To create a fair playing field, we only look at connections that exchange

more than 1500 packets and exclude shorter connections from both the background data. This number seems like a suitable minimal limit for a successful interactive stepping-stone chain, and there were no connections with less packets in the stepping-stone dataset add validation about how this is sufficient for all approaches? . The first 1500 packets are then passed to each SSD method to make a prediction.

The initialisation of the SSH-tunnels usually follows a distinct pattern that can be learned and consequently boost detection rates for ML-based methods. Since this pattern is not necessarily representative of actual stepping-stone behaviour, we remove the first thirty packets of all captured connections. If necessary, we also remove the last thirty packets to avoid the same issue for connection closures.

3.7 Stepping-stone data

We generate stepping-stone data following the steps described in Section 2. We create a main dataset using a chain of four stepping-stones S_1, S_2, S_3, and S_4. We subdivide this main dataset according to the two discussed evasion tactics, transfer delays and chaff perturbations. We first capture data without either of these methods. We then capture data once with added transfer delays with varying δ_d to control delays, and once with added chaff perturbations of varying δ_C.

We furthermore create four smaller datasets with differing numbers of stepping-stones to evaluate this effect on RTT-based methods.

Table 2 provides a summary of the connection pair numbers in each part of the dataset.

4 jumps	# connection pairs
No evasion	30.000
Delays	30.000
Chaff	30.000
1 jump	1.000
3 jump	1.000
5 jump	1.000
8 jump	1.000

Table 2. Number of connection pairs in each part of the stepping-stone dataset.

3.8 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.

We include real-world traffic traces, taken from the *CAIDA 2018 Anonymized Internet Traces* dataset [1]. insert citation and description.

We group packets to connections according to the usual 5-tuple consisting of {Src. IP, Dst. IP, Src. port, Dst. port, IP protocol}. We select and pair TCP-connections at random to make 100.000 connection pairs.

The CAIDA dataset provides suitable general background traffic. However, to sufficiently test for false-positive, we need to include benign traffic that has similar characteristics to the attack traffic and was generated in a similar network environment, which the CAIDA dataset not necessarily provides. For that, we created a set of interactive SSH-connections that are conducted directly between the client and the server instead of via a stepping-stone. We follow the same procedure as described in Section 2.1 with the same randomised network congestion settings as described in 2.1. We then proceed to pair connections by random.

Additionally, we include a third type of benign traffic to test false-positives of methods aiming to spot chaff perturbations. Since we followed the findings of Padhye et al. [11] to generate chaff perturbations that resemble multimedia streams, we want to include actual multimedia streams. For that, we captured traffic from a server streaming video to a client. Video content is randomised and multiple values for the streaming quality are used. The server and client are hosted in the same virtual Docker network with randomised network congestion settings. Again, we pair generated connections by random. However, for each pair the direction of the transfer has to be either to or from the common host to either resemble a client requesting two videos or a server sending to videos.

We merge the three datasets to create our benign background dataset according to the following ratios:

	# pairs
CAIDA	60.000
SSH	20.000
multimedia	20.000

Table 3. Ratio of traffic in the benign dataset.

We are aware that the amount of interactive SSH traffic and multimedia streams in this setting is inflated from a realistic setting, but we believe that this setting is more suitable to highlight the strengths and drawbacks of SSD methods. In the evaluation, we will also state individual false positives for each of the three datasets to indicate which type of traffic present the biggest challenge for the selected SSD methods.

3.9 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.

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

3.10 Implementation of selected approaches

add short description how we trained approaches

4 Results

4.1 Data without evasion tactics

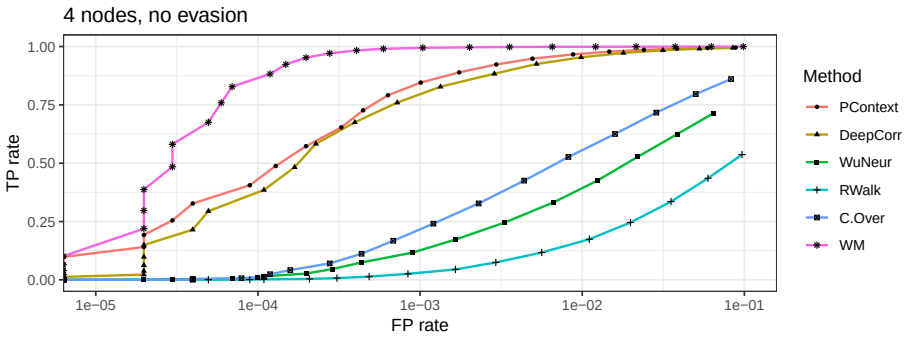

Fig. 3. 4 nodes

First, we look at the detection rates for traffic from stepping-stones that did not use any evasive tactics, i.e. $S_1, \ldots, S_4$ are only forwarding the commands and responses. The successful detection of this tytpe of behaviour with

	PContext	DeepCorr	WuNeur	RWalk	C.Over	WM
AUC	0.998	0.997	0.939	0.856	0.965	0.9998

Table 4. AUC-scores for different methods on stepping-stone data without evasive tactics.

low false-positives should be the minimum requirement for any of the selected methods except for the anomaly-based approaches. Since these aim to detect evasive behaviour, we exclude them from this analysis.

To get clear picture of the capabilities of the selected methods, we look at the ROC-curve in Fig. 3, which plots the true positive rate against the false positive rate for various detection threshold settings. Table 4 depicts the overall AUC-scores.

Unsurprisingly, the watermarking method achieves already high detection results without any false-positives. Both the PContext and DeepCorr models start to yield good detection results of around 80% at a FP rate lower than 0.1%, with the PContext method slightly outpacing the DeepCorr method.

RTT-based methods seem to not perform as well compared to the other included methods, however the observed ROC curve seems to be in general in agreement of the stated detection rates check this again.

4.2 Delays

We now consider the effect of transfer delays added by the attacker to packets on the detection rates. For that, we pick detection thresholds for each SSD methods corresponding to a FP rate of 0.4% as most methods are able to achieve some detection results at this rate.

We look at delays added to only to outgoing packets on S_4, the last stepping stone in the chain. Fig. 4 depicts evolution of detection rates in dependence of the maximum delay δ_d.

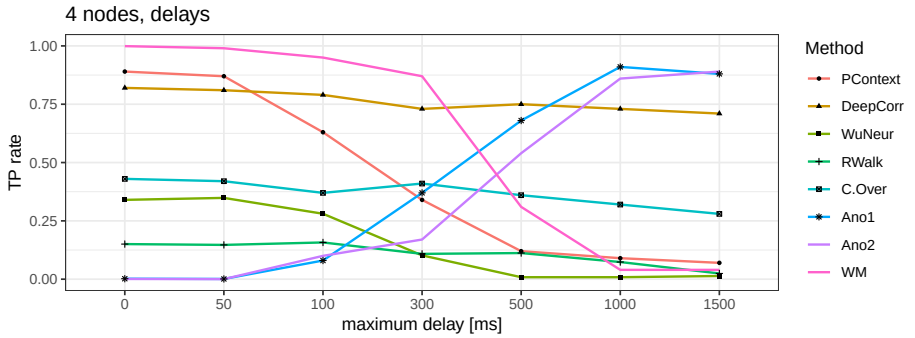

Fig. 4. ...

As visible, both anomaly-based methods are capable of detecting added delays relatively reliably. Furthermore, both the detection rates of DeepCorr and the rtt-based C.Over only decrease slightly under the influence of delays. Detection rates for all other methods decrease significantly to the point where no meaningful predictions can be made. This is also reflected by the AUC-scores for traffic with $\delta_d = 1000ms$, given in Table 6.

	PContext	DeepCorr	WuNeur	RWalk	C.Over	Ano1	Ano2	WM
AUC	0.636	0.995	0.614	0.642	0.953	0.997	0.996	0.561

Table 5. AUC-scores with added transfer delays at $\delta_d = 1000ms$.

4.3 Chaff

We now consider the effect of chaff perturbations added by the attacker to individual connections on the detection rates. Again we pick detection thresholds for each SSD methods corresponding to a FP rate of 0.4%.

Chaff packets are added to both the connection between S_3 and S_4 as well as between S_4 and host T as described in Section 2.2. Fig. 5 depicts evolution of detection rates in dependence of the ratio of number of chaff packets to packets from the actual interaction.

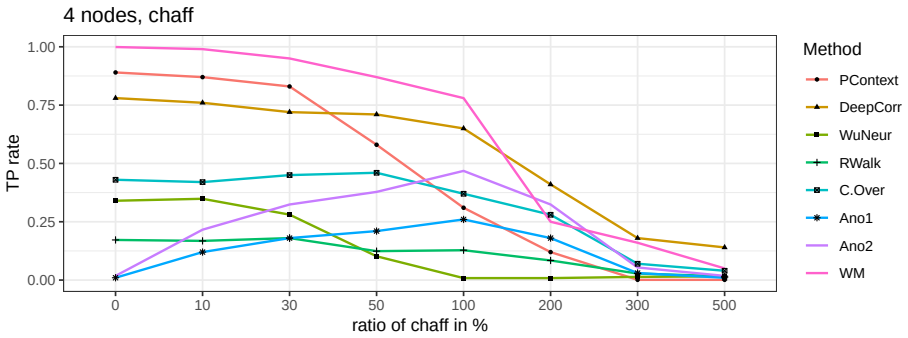

Fig. 5. ...

As visible, all methods struggle to detect stepping stones once the chaff packets become the majority of the transferred traffic. This is also evident from the AUC-scores given in Table 6. Several approaches claimed to be resistent to chaff perturbations, however prior evaluations never looked at chaff ratios that exceed 100%.

It is surprising that the anomaly detection methods do not perform better at detecting chaff perturbations. Chaff in both approaches was however evaluated with different traffic generation distribution and not compared against a background of traffic following the same generation distribution, which could explain the disagreement between the results we are finding here.

	PContext	DeepCorr	WuNeur	RWalk	C.Over	Ano1	Ano2	WM
AUC	0.637	0.884	0.615	0.644	0.585	0.780	0.737	0.839

Table 6. AUC-scores with added chaff at 300% ratio.

4.4 False positives

Table 7 depicts the relative contribution[1] at $FP = 0.4\%$ of each of the three benign data types to the overall false positive rate. Most methods have more problems with the heterogeneous nature the CAIDA traces, with only PContext and DeepCorr seeing most false positives in the SSH traffic.

The multimedia traffic is causing most problems for the anomaly-based methods, persumably because it follows a similar distribution as the generated chaff perturbations.

	PContext	DeepCorr	WuNeur	RWalk	C.Over	Ano1	Ano2	WM
CAIDA	0.37	0.37	0.51	0.67	0.57	0.49	0.36	0.82
SSH	0.52	0.46	0.22	0.23	0.32	0.08	0.00	0.12
multimedia	0.11	0.17	0.27	0.10	0.11	0.43	0.64	0.06

Table 7. Relative contribution in % of different benign data to FP rate at $FP = 0.4\%$.

4.5 Influence of chain length and probe placement

In this section, we look at the effect of differing chain lengths on the detection rates. We only focus on RTT-based methods here since the other methods do not seem to be significantly influenced. Since RTT-based methods aim to measure the effect of packets travelling via multiple hosts, it is unsurprising that they perform better at detecting longer chains.

add comparison of results at different positions

Of the RTT-based methods, only C.Over was able to yield consistent detection rates under transfer delays. Interestingly, if the C.Over method is applied to connections between S_3 and S_4 instead of between S_4 and the target, detection rates decrease in the same manner as for other RTT-based methods. This is not

[1] after adjusting for their weight

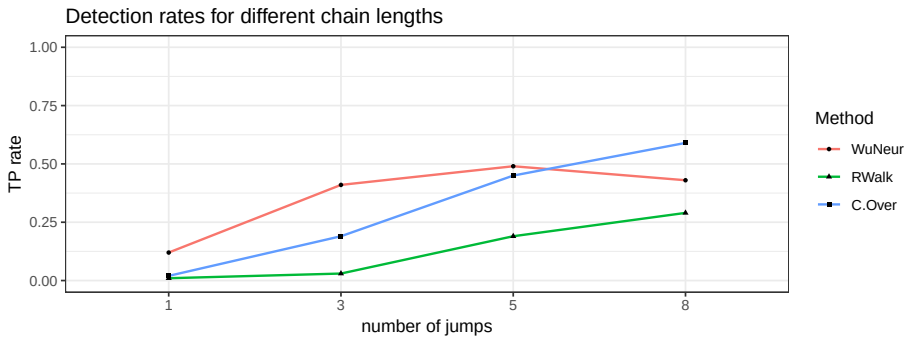

Fig. 6. ...

surprising as the underlying assumption for robustness for this approach relies on Echo-packets not being delayed.

4.6 Influence of network settings

	Value	TP deviation from average				
		DeepCorr	WuNeur	RWalk	C.Over	WM
RTT	5ms	−0.2%	+41.3%	−42.3%	−36%	+0.03%
	70ms	−5.6%	−5.8%	+35.1%	+51%	−2.2%
Packet loss	0%	+1.2%	+1.3%	+2.1%	+4.3%	+0.02%
	7%	−9.1%	−1.1%	−3.1%	−7.3%	−9.7%

Table 8. Influence of network congestion on detection rates at $FP = 0.4\%$. The given percentages are describing the change of the detection rate under the given congestion setting when compared to the overall average.

Finally, we look at the effect of different nework settings. We only show methods that show significant effects and omitted bandwidth from the evaluation as different values do not seem to have any effect on detection rates.

As visible in Table 8, the three RTT-based methods show different responses to small/large average round-trip times. While WuNeur, as expected from prior results, performs better in LAN settings, detection rates of the RWalk and C.Over methods are boosted by larger RTTs.

All methods profit from lower packet losses.

5 Conclusion

...

References

1. The CAIDA UCSD Anonymized Internet Traces 2018, 2018. Accessed: 2020-02-10.
2. Baris Coskun and Nasir Memon. Efficient detection of delay-constrained relay nodes. In *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, pages 353–362. IEEE, 2007.
3. Giovanni Di Crescenzo, Abhrajit Ghosh, Abhinay Kampasi, Rajesh Talpade, and Yin Zhang. Detecting anomalies in active insider stepping stone attacks. *JoWUA*, 2(1):103–120, 2011.
4. Wei Ding, Matthew J Hausknecht, Shou-Hsuan Stephen Huang, and Zach Riggle. Detecting stepping-stone intruders with long connection chains. In *2009 Fifth International Conference on Information Assurance and Security*, volume 2, pages 665–669. IEEE, 2009.
5. Wei Ding, Khoa Le, and Shou-Hsuan Stephen Huang. Detecting stepping-stones under the influence of packet jittering. In *2013 9th International Conference on Information Assurance and Security (IAS)*, pages 31–36. IEEE, 2013.
6. Marco Gamarra, Sachin Shetty, David M Nicol, Oscar Gonazlez, Charles A Kamhoua, and Laurent Njilla. Analysis of stepping stone attacks in dynamic vulnerability graphs. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2018.
7. Shou-Hsuan Stephen Huang and Ying-Wei Kuo. Detecting chaff perturbation on stepping-stone connection. In *2011 IEEE 17th International Conference on Parallel and Distributed Systems*, pages 660–667. IEEE, 2011.
8. Shou-Hsuan Stephen Huang, Hongyang Zhang, and Michael 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)*, pages 1043–1050. IEEE, 2016.
9. Alfonso Iacovazzi and Yuval Elovici. Network flow watermarking: A survey. *IEEE Communications Surveys & Tutorials*, 19(1):512–530, 2016.
10. Milad Nasr, Alireza Bahramali, and Amir Houmansadr. Deepcorr: Strong flow correlation attacks on tor using deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1962–1976, 2018.
11. Jaideep D Padhye, Kush Kothari, Madhu Venkateshaiah, and Matthew Wright. Evading stepping-stone detection under the cloak of streaming media with sneak. *Computer Networks*, 54(13):2310–2325, 2010.
12. Robert Shullich, Jie Chu, Ping Ji, and Weifeng Chen. A survey of research in stepping-stone detection. *" International Journal of Electronic Commerce Studies"*, 2(2):103–126, 2011.
13. Lixin Wang and Jianhua Yang. A research survey in stepping-stone intrusion detection. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):276, 2018.
14. Xinyuan Wang and Douglas Reeves. Robust correlation of encrypted attack traffic through stepping stones by flow watermarking. *IEEE Transactions on Dependable and Secure Computing*, 8(3):434–449, 2010.
15. Han-Ching Wu and Shou-Hsuan Stephen Huang. Performance of neural networks in stepping-stone intrusion detection. In *2008 IEEE International Conference on Networking, Sensing and Control*, pages 608–613. IEEE, 2008.
16. Han-Ching Wu and Shou-Hsuan Stephen Huang. Neural networks-based detection of stepping-stone intrusion. *expert systems with applications*, 37(2):1431–1437, 2010.

17. Jianqiang Xin, Lingeng Zhang, Brad Aswegan, John Dickerson, T Daniels, and Yong 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.*, pages 9–pp. IEEE, 2006.
18. Jianhua Yang and Shou-Hsuan Stephen Huang. Mining tcp/ip packets to detect stepping-stone intrusion. *computers & security*, 26(7-8):479–484, 2007.
19. Jianhua Yang and David Woolbright. Correlating tcp/ip packet contexts to detect stepping-stone intrusion. *Computers & Security*, 30(6-7):538–546, 2011.
20. Jianhua Yang and Yongzhong Zhang. Rtt-based random walk approach to detect stepping-stone intrusion. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 558–563. IEEE, 2015.
21. Linfeng Zhang, Anthony Persaud, Alan Johnson, and Yong 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.