Evading Stepping-Stone Detection with Enough Chaff

Henry Clausen[1(✉)], Michael S. Gibson[2], and David Aspinall[1,3]

[1] University of Edinburgh, Edinburgh, UK
{henry.clausen,david.aspinall}@ed.ac.uk
[2] BT Applied Research, Ipswich, UK
michael.s.gibson@bt.com
[3] The Alan Turing Institute, London, UK

Abstract. Stepping-stones are used extensively by attackers to hide their identity and access restricted targets. Many methods have been proposed to detect stepping-stones and resist evasive behaviour, but so far no benchmark dataset exists to provide a fair comparison of detection rates. We propose a comprehensive framework to simulate realistic stepping-stone behaviour that includes effective evasion tools, and release a large dataset, which we use to evaluate detection rates for eight state-of-the-art methods. Our results show that detection results for several methods fall behind the claimed detection rates, even without the presence of evasion tactics. Furthermore, currently no method is capable to reliably detect stepping-stone when the attacker inserts suitable chaff perturbations, disproving several robustness claims and indicating that further improvements of existing detection models are necessary.

1 Introduction

The problem of stepping-stones detection (SSD) has been studied for over 20 years, yet the body of literature fails at providing an informative overview of the detection capabilities of current methods. In this paper, we set out to do just that by evaluating and comparing a number of selected state-of-the-art approaches on a new and independently generated dataset.

In a stepping-stone attack, malicious commands are relayed via a chain of compromised hosts, called stepping-stones, in order to access restricted resources and reduce the chance of being traced back. Real-world attacks using stepping-stone chains include Operation Aurora [19], Operation Night Dragon [1], the Black Energy [13] attack on the Ukrainian powergrid, and the MEDJACK [3] attack where medical devices were used as stepping-stones. The European Union Agency for Cybersecurity currently classifies stepping-stone attacks as one of the top ten threats to IoT-devices [8].

The detection of interactive stepping-stones is challenging due to various reasons. Attackers are not constrained to specific proxy techniques and can obfuscate relayed traffic with evasive tactics. Packet-based methods are computationally expensive while false-positives can render a method unusable. Like many

M. Kutyłowski et al. (Eds.): NSS 2020, LNCS 12570, pp. 431–446, 2020.
https://doi.org/10.1007/978-3-030-65745-1_26

intrusion attacks, stepping-stones are rare and there exist no public datasets, leading researchers to evaluate their methods on self-provided private data, which makes a direct comparison of the achieved results impossible.

In this work, we provide the following contributions:

1. We describe a framework to generate data that represents realistic stepping-stone data without bias to particular detection mechanisms. Our framework is scalable and capable of generating sufficient variety in terms of network settings and conducted activity.
2. We release a large and comprehensive dataset suitable for the training of machine-learning-based methods and in-depth performance evaluation. To our knowledge, this is the first public SSD dataset.
3. We re-implemented eight SSD methods that represent the current state-of-the-art and provide a fair evaluation of their capabilities in a number of settings.
4. Our evaluation shows that while most methods can accurately detect command propagation, detection rates plummet when appropriate chaff is inserted. This result disproves the claims made for multiple methods that their detection rates are robust against chaff perturbations.

The rest of the paper is organised as following: Sect. 1 provides an introduction and background to the problem of stepping-stone detection. Section 2 discusses the particular design of the data generation framework. Section 3 presents the dataset arrangement in terms of background and attack data and discusses evaluation methods. Section 4 discusses the selection process, properties, and implementation of the eight SSD methods that we implemented for evaluation. Section 5 discusses the results achieved by the implemented methods on the given data. Section 6 discusses related work.

1.1 Background

Stepping-stones were first conceptualised by Staniford-Chen and Heberlein in 1995 [18]. In an interactive stepping-stone attack, an attacker located at the origin host, called *host O*, sends commands to and awaits their response from a target, *host T*. The commands and responses are proxied via a chain of one or more intermediary stepping-stone hosts, called *host* $S_1, \ldots, S_N$, such as depicted in Fig. 1. Once a host S_i is brought under control, it can be turned into a stepping-stone with simple tools and steps. Some of the most common set-ups are port forwarding via SSH-tunnels, setting up a backpipe with NetCat, or using metasploit to set up a SOCKS proxy [9].

Stepping-stone detection (SSD) is a process of observing all incoming and outgoing connections on a particular host h_i and determining whether it is used to relay commands. This is generally done with no prior information about any other stepping-stone hosts $S_1, \ldots S_N$ or the endpoints O and T. A popular approach to SSD is to compare connections pairwise to identify whether they carry the same information. To avoid detection, several evasive flow transformation

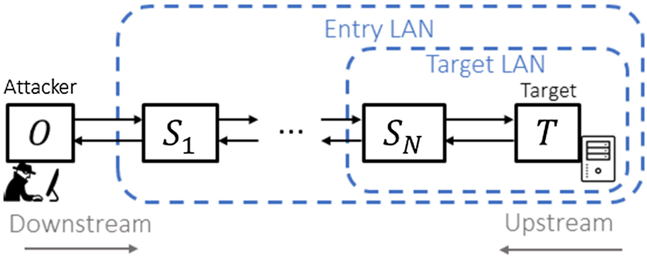

Fig. 1. Depiction of an exemplary stepping-stone chain.

techniques exist that aim at decreasing observable correlation between two connections in a chain.

- **Packet transfer delays/drops**: An attacker can choose to apply artificial delays to forwarded packets, or drop certain packets to cause retransmission, in order to create temporal disparity between connections. Researchers often assume the existence of a maximum tolerable delay [7].
- **Chaff perturbations**: Chaff packets do not contain meaningful content and are added to individual connections in a chain without being forwarded. Adding chaff perturbations 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, splitting a packet into multiple smaller packets, or altering the packet content to change observed packet sizes and numbers.

In our evaluation, we set out to understand the effect of different evasive methods on detection rates.

2 Data Generation Setting

2.1 Containerisation

To ensure reproducibility, we rely on containerisation. A container is a standard unit of software that runs standalone in an isolated user space in order to remove platform dependencies and ensure repeatability. The use of containerisation for this project follows a traffic generation paradigm designed for machine learning, introduced by Clausen et al. [4].

2.2 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 begins with the start-up of the necessary containers and ends with their takedown. We simulate 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 a port from host O to host S_1, which is then forwarded to host S_2 by the second tunnel etc. As mentioned by Gordon Fraser [9], this is one of the most common pivoting methods for attackers. Traffic is captured both at host T and host S_n, which acts as the final stepping-stone in the chain. Figure 2 depicts a packet transfer via an exemplary chain.

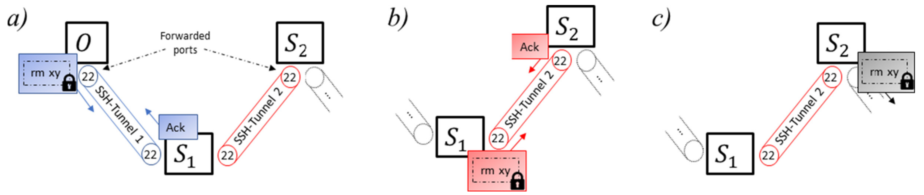

Fig. 2. Depiction of the way a command is packetised, encrypted, and travels through the different stages of the stepping-stone chain via SSH-tunnels.

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.

To mimic a user's actions, we compiled a command database which consists of common commands and their usage frequency, similar to [24]. 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 commands such as inputting text to a file. Command inputs are randomized appropriately when a compound command is drawn. A scripts ends once the *End*-command is drawn at random from the command catalogue.

To simulate human behaviour that is reacting to the response from host T, all commands are separated by *sleep*-commands for time t, which is drawn from a truncated Pareto-distribution. Paxson et al. [16] have shown that interpacket spacings corresponding to typing and "think time" pauses are well described by Pareto distributions with a shape parameter $\alpha \approx 1.0$.

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, which influences the round-trip-time, bandwidth, and network reliability.

To retard the quality of the Docker network to realistic levels, we rely on the emulation tool NetEm, which allows users to artificially simulate network conditions such as high latency, low bandwidth, or packet corruption/drop [10]. We set the network settings and bandwidth limit for each host container individually before each simulation to allow hosts to experience different settings.

2.3 Evasive Tactics

Adding Transfer Delays. To simulate evasive behaviour, we add transfer delays to forwarded packets. This method, often called *jittering*, can destroy time-based watermarks in packet flows and help decrease observable correlation between two connections. The delays are added using NetEm. We draw delays from a uniform distribution, covering the interval $[0, \delta_D]$. This particular choice has been suggested by Padhye et al. [15] in order to mimic the interarrival distributions of streaming services. The value of δ_D is fixed before each simulation and can be varied to allow for different degrees of packet jittering. We explore values for δ_D up to 1500 ms, with values above leading to unstable communication. Results in Sect. 5 show that this is enough to render watermarking methods and most flow correlation methods obsolete.

Adding Chaff Perturbation. We insert chaff packets without actual information to individual connections in the chain using a Netcat client. 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. Padhye et al. [15] suggest to generate chaff that mimics the flow characteristics of streaming services to both spread the added perturbations evenly across the connection and increase the difficulty of detecting the perturbation itself. For this, packet sizes are drawn from a truncated Lognormal-distribution with mean μ_C, while transmission intervals are drawn from a uniform distribution that covers the interval $[\delta_C/2, \delta_C]$ to mimic a constant packet flow. By adjusting δ_C, we can control the amount of chaff sent.

Repacketisation. By design, SSH-tunnels perform repacketisation along with re-encryption and independent packet confirmations.

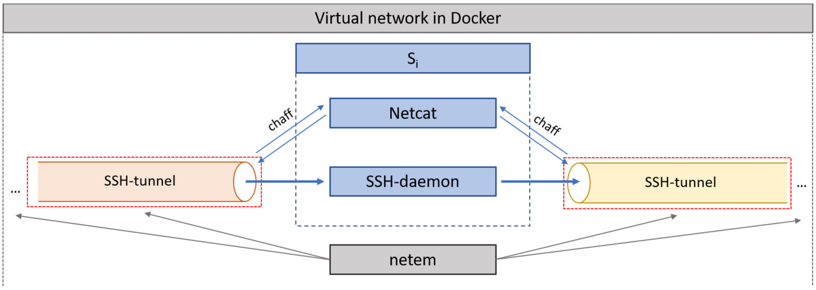

Fig. 3. Depiction the simulation setup for each host in the chain.

3 Evaluation Data

We want to look at a variety of attack scenarios to highlight the strengths and weaknesses of different SSD approaches. We created three main attack datasets that contain different forms and amounts of evasive behaviour, and a smaller dataset to highlight the influence of different chain lengths.

To present a valuable false positive test, we provide three datasets with benign background traffic. The first contains general real-world traffic, while the second and third contain benign data that bears similar traffic characteristics as the generated attack data.

3.1 Stepping-Stone Data

We create our main datasets using a chain of four stepping-stones S_1, S_2, S_3, and S_4. We subdivide into three datasets: We first capture data without transfer delays and chaff perturbations in **dataset BA** (**b**aseline **a**ttack). We then capture data once with added transfer delays with varying δ_D to control delays in **dataset DA** (**d**elay **a**ttack), and once with added chaff perturbations of varying δ_C in **dataset CA** (**c**haff **a**ttack). Each dataset contains 30.000 connection pairs. We furthermore create a smaller **dataset CL** (**c**hain **l**ength) with differing numbers of stepping-stones (1,3,5, and 8 jumps) without transfer delays and chaff perturbations.

3.2 Benign Data

We include real-world traffic traces, taken from the **CAIDA** *2018 Anonymized Internet Traces* dataset [2], as overall background traffic. This data contains traces collected from high-speed monitors on a commercial backbone link, and is often used for research on the characteristics of Internet traffic.

To sufficiently test for false-positive, we also need to include benign traffic that has similar characteristics to the attack traffic and was generated in a similar network environment. We created a set of interactive SSH-connections that communicate directly between the client and the server without a stepping-stone. We follow the same procedure as described in Sect. 2.2.

Since we generate perturbations with multimedia streams characteristics, we additionally want to test for false-positives against actual multimedia stream traffic. For that, we captured traffic from a Nginx-server streaming randomised video to a client.

We merge the three datasets to create our benign background dataset, with the CAIDA part containing 60.000 connection pairs, while the other two each contain 20.000 connection pairs. The amount of SSH traffic and multimedia streams in this setting is inflated from a realistic setting (up to 0.2% of flows for SSH and up 3% for video streaming [20]) to highlight the strengths and drawbacks of SSD methods, which we consider in the evaluation. In Sect. A, we analyze false-positives for each dataset individually. Table 1 summarizes the different parts in our evaluation data.

Table 1. Summary of different components in our evaluation data.

	Label	Nr. of conn.	Purpose
Attack data	Set BA	30.000	Baseline attack data without evasion tactics
	Set DA	30.000	Inclusion of delays with varying δ_D
	Set CA	30.000	Inclusion of chaff with varying δ_C
	Set CL	40.000	Data from chains of different lengths, no evasion tactics
Background data	CAIDA	60.000	General background data
	SSH	20.000	Background data similar to attack commands
	Multim.	20.000	Background data similar to chaff perturbations

3.3 Evaluation Methodology

To create a fair playing field for the selected SSD methods, we only look at connections that exchange more than 1500 packets and exclude shorter connections from both the data. The number of packets necessary for detection should ideally be a low possible to enable early detection. The chosen number of 1500 packets seems like a suitable minimal limit since all of the selected methods are designed to make successful detection with 300–1500 packets. Furthermore, there were no connections with less packets in the stepping-stone dataset.

True stepping stone connections are rare compared to benign ones, making their detection an imbalanced classification problem. An appropriate evaluation measure for imbalanced data are false positive and false negative rates as well as the *Area-under-ROC-curve* (AUC) for threshold-based methods.

4 Selected SSD Methods and Implementation

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 started with the publications from surveys [17,21], and then added impactful recent publications found via Google Scholar[1]. From here, we selected approaches based on the following criteria:

1. The achieved detection and false positive rates claimed by the authors,
2. and whether the model design shows robustness against any evasion tactics as claimed by the authors.
3. We always selected the latest versions if a method has been improved by the authors.

Table 2 contains a summary of the included methods. Especially for traditional packet-correlation as well as robust watermarking and anomaly-based methods, there has been little developments since the early 2010s. We labelled each method to make referring to it in the evaluation easier.

[1] Keywords “connection”, “correlation” “stepping-stone”, “detection”, “attack”, “chaff perturbation”.

Table 2. Summary of included SSD-methods along with the claimed true positive and false positive rates and evasion robustness by the corresponding authors. We added labels to each method for later reference.

Category	Approach	TP	FP	Robustness	Label
Packet-corr	Yang, 2011 [26]	100%	0%	Jitter/$<$ 80% chaff	PContext
Neural networks	Nasr, 2018 [14]	90%	0.0002%	Small jitter	DeepCorr
	Wu, 2010 [23]	100%	0%	-	WuNeur
RTT-based	Yang, 2015 [27]	Not provided		50% chaff	RWalk
	Huang, 2016 [12]	85%	5%	-	Crossover
Anomaly-based	Crescenzo, 2011 [5]	99%	1%	Jitter/chaff	Ano1
	Huang, 2011 [6,11]	95%	0%	$>$ 25% chaff/ $>$ 0.2 s jitter	Ano2
Watermarking	Wang, 2011 [22]	100%	0.5%	$<$ 1.4 s jitter	WM

PContext, 2011. Yang et al. [26] compare sequences of interarrival times in connection pairs to detect potential stepping-stone behaviour. For that, the contextual distance of a packet is defined as the packet interarrival times around that packet. The authors focus on *Echo*-packets instead of *Send*-packets to resist evasion tactics. The authors evaluate their results with up to 100% chaff ratio with 100% detection rate.

WuNeur, 2010. Wu et al. [23] propose a neural network model based on sequences of RTTs , which are fed into a feed-forward network to predict the downstream length of the chain. The network itself only contains one hidden layer and achieves good results only if RTTs are small, i.e., when the stepping-stone chain is completely contained within one LAN-network.

DeepCorr, 2018. Nasr et al. [14] train a deep convolutional neural network to identify connection correlation from the 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. On stepping-stones, the authors achieve a 90% detection rate with 0.02% false positives.

RWalk, 2015. Yang et al. [27] combine packet-counting methods and RTT mining methods to improve detection results from [25]. The model resists chaff perturbation by estimating the number of round-trips in a connection via packet-matching and clustering to determine if the connection is being relayed.

C-Over, 2016. Huang et al. [12] use the fact that in long connection chain, the round-trip-time of a packet may be longer than the intervals between two consecutive keystrokes. 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.

Ano1, 2011. Crescenzo et al. [5] have proposed an anomaly-based methods to detect time delays and chaff perturbations in a selected connection. Packet time-delays are detected if RTTs exceed a threshold, while chaff detection compares the similarity of downstream with upstream sequences. The authors claim detection for chaff ratios 25% or more, and for delays introduced to up to 70% of all packets.

Ano2, 2011/2013. Huang et al. [6,11] proposed an anomaly-based method to detect chaff and delay perturbations since interarrival times in regular connections tend to follow a Pareto or Lognormal distribution, which chaffed connections supposedly do not. The authors state 95% detection rate at 50% chaff ratio and more while retaining zero false positives using a small set of interactive SSH stepping-stone connections.

WM, 2010. Watermarking typically yields very low false-positives for connection correlation. Wang et al. [22] provide an approach that offers 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 state 100% TP with 0.5% FP with resistance against timing perturbations of up to 1.4 s.

5 Results

5.1 Data Without Evasion Tactics

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 commands and responses. The successful detection of this activity with low false-positives should be the minimum requirement for any SSD method. Since anomaly-based approaches aim to only detect evasive behaviour, we exclude them from this analysis.

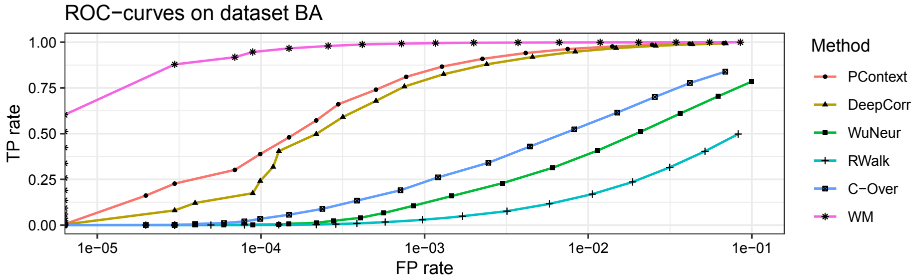

Fig. 4. ROC-curves for different SSD methods on dataset BA (no evasive tactics). Anomaly-based methods are excluded.

Table 3. AUC-scores for different methods on stepping-stone data without evasive tactics.

	PContext	DeepCorr	WuNeur	RWalk	C-Over	WM
AUC	0.998	0.997	0.938	0.853	0.965	0.9998

Figure 4 depicts the calculated ROC-curves, which plot the true positive rate against the false positive rate for varying detection thresholds. Table 3 depicts the overall AUC-scores.

Unsurprisingly, the watermarking method achieves high detection results with very low 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. Overall, the observed ROC curves seem to be in agreement with the stated detection rates of the selected methods except for RWalk.

5.2 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 at least moderate 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. Figure 5 depicts evolution of detection rates in dependence of the maximum delay δ_D.

As visible, both anomaly-based methods are capable of detecting added delays relatively reliably above a certain threshold. 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 $\delta_D = 1000$ ms, given in Table 4.

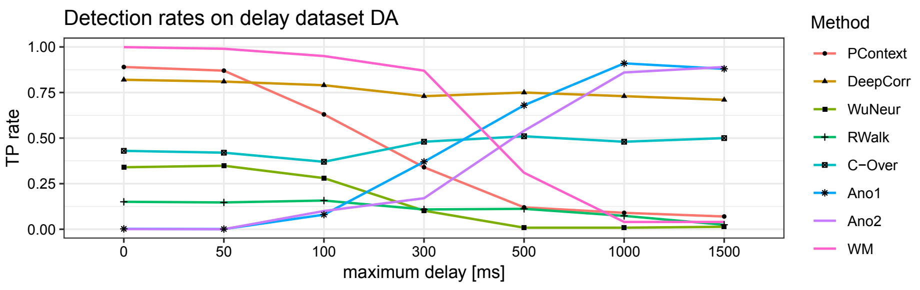

Fig. 5. Detection rates in dependence of δ_D for different methods on dataset DA with a fixed FP rate of 0.4%.

While the WM method is robust against transfer delays up to $\delta_D = 500$ ms, this value is smaller than the one claimed by the authors. This might however be a result of the slightly smaller quantisation step size that we used. It is surprising that the PContext method shows only little robustness against transfer delays, which contradicts the authors claims, potentially due to the incorrect assumption that relying on *Echo*-packets are not subject to transfer delays.

Table 4. AUC-scores for SSD methods with added transfer delays at $\delta_D = 1000$ ms.

	PContext	DeepCorr	WuNeur	RWalk	C-Over	Ano1	Ano2	WM
AUC	0.638	0.995	0.613	0.641	0.952	0.997	0.996	0.562

5.3 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 Sect. 2.3. Figure 6 depicts evolution of detection rates in dependence of the ratio of number of chaff packets to packets from the actual interaction.

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 5. Several approaches claimed to be resistent to chaff perturbations, however prior evaluations were limited chaff ratios below 100% without obvious reason.

It is surprising that the anomaly detection methods do not perform better at detecting chaff perturbations. Chaff in both approaches was however evaluated

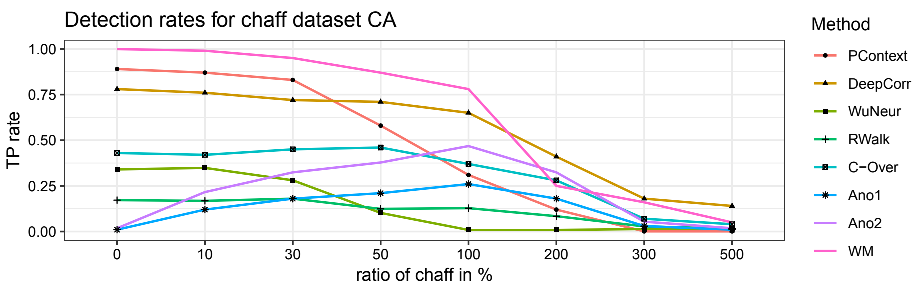

Fig. 6. Detection rates in dependence of δ_C for different methods on dataset CA with a fixed FP rate of 0.4%

with different traffic generation distribution and not compared against a background of traffic following a similar generation distribution, which could explain the disagreement between the results we are finding here.

Overall, these results are in disagreement with the "robustness" claims made for four of the selected approaches, namely PContext, RWalk, Ano1, and Ano2.

Table 5. AUC-scores for SSD methods with added chaff at 300% ratio.

	PContext	DeepCorr	WuNeur	RWalk	C-Over	Ano1	Ano2	WM
AUC	0.639	0.886	0.615	0.641	0.589	0.782	0.738	0.839

5.4 Summary

Overall, detection rates on dataset BA are mostly in line with the claimed capabilities except for RWalk, although detection rates are slightly lower than stated by most authors. Delay perturbation increases detection difficulty for most methods, except for Ano1, Ano2, and DeepCorr, which contradicts robustness claims for PContext and to some extend WM. Our inserted chaff perturbations however render detection impossible for all methods examined, which contradicts robustness claims for PContext, Ano1, Ano2, and RWalk, even though the claims were based on lower chaff levels.

As discussed in Sect. B and C, longer chains yield higher detection rates for RTT-based methods while Different network transmission settings seem to have overall little influence on detection rates.

6 Related Work

6.1 Testbeds and Data

In 2006, Xin et al. [24] developed a standard test bed for stepping-stone detection, called *SST* that generates interactive SSH and TelNet connection chains with variable host numbers. In contrast to our work, the authors give little detail on implemented evasive tactics, and is not available anymore.

An approach to use publicly available data comes from Houmansadr et al. [14],who simulate stepping stones by adding packet delays and drops retroactively to connections in the CAIDA data [2]. While this procedure seems sufficient for the evaluation of watermarking methods, it falls short on simulating the effects of an actual connection chain and leaves out chaff perturbations.

We find that when authors evaluate methods on self-generated data, tested evasive behaviours are often lacking analytical discussion and their implementations are too simplistic, leading to increased detection rates. An example of this can be seen in the evaluation of Ano1 [5], where a standard option in netcat is used to generate chaff perturbations for evaluation, or for PContext [27] where simulated chaff is added randomly after the traffic collection. Furthermore, often a relatively low limit on the amount of inserted chaff perturbations is assumed without obvious reason, thus avoiding evaluation at higher ratios.

7 Conclusion

In this work, we set out to evaluate the state-of-the-art of SSD methods using a comprehensive data generation framework. Our framework simulates realistic stepping-stone behaviour with SSH-tunnels in different settings and varying amounts of evasive perturbation tactics. We will release a large dataset that highlights multiple aspects in SSD, and is suitable to train ML-based methods.

Overall, our results show that attackers can reliably evade detection by using the right type and amount of chaff perturbation, which disproves several claims made about the robustness against this evasive tactic. Although to a lesser degree, our implemented delay perturbations still affect detection rates for most methods.

Currently, it seems that watermarking methods are most suited to reliably detect simple stepping-stones in real-life deployment. The performance of DeepCorr indicates that deep neural networks show the most potential at detecting attacks that use chaff or delay perturbations if they are trained on suitable data. We find that detection and false-positive rates for RTT-based methods are significantly lower than for other methods.

Acknowledgments. We are grateful to BT Group PLC who are supporting the PhD research of the first author in the UK EPSRC CASE scheme, giving invaluable guidance on the needs and possibilities of intelligent security tools and their evaluation. The third author was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1 and the Office of Naval Research ONR NICOP award N62909-17-1-2065.

A False Positives

Table 6 depicts the relative contribution[2] 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.

[2] After adjusting for their weight.

Table 6. Relative contribution in % of different benign data to the FP rate.

	PContext	DeepCorr	WuNeur	RWalk	C-Over	Ano1	Ano2	WM
CAIDA	0.36	0.46	0.47	0.67	0.53	0.48	0.35	0.81
SSH	0.53	0.46	0.21	0.28	0.27	0.05	0.02	0.08
Multimedia	0.11	0.08	0.32	0.04	0.20	0.47	0.63	0.11

B Influence of Chain Length

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 should and do not see a significant effect from varying chain lengths[3]. 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.

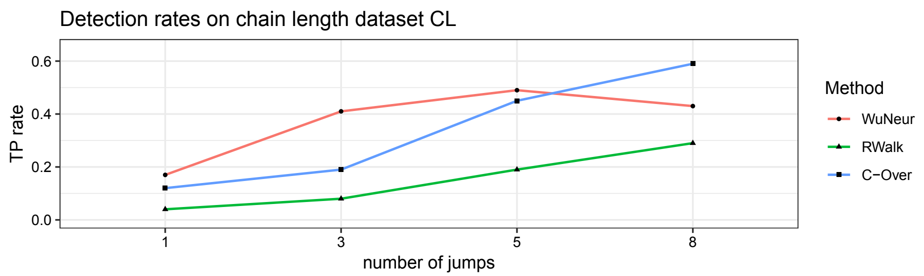

Fig. 7. Detection rates in dependence of chain length for different methods on dataset CL with a fixed FP rate of 0.4%

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

Table 7. Influence of network congestion on detection rates at a fixed FP rate of 0.4%. The given percentages are describing the change of the detection rate under the given congestion setting when compared to the overall average.

	Value	TP deviation from average				
		DeepCorr	WuNeur	RWalk	C-Over	WM
RTT	5 ms	−0.2%	+41.3%	−42.3%	−36%	+0.03%
	70 ms	−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%

[3] For non-RTT-methods, the detection rate error (2.6%–6.5%) for each length was larger than the detection rate differences (0.2%–3.7%) across different lengths.

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

C Influence of Network Settings

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[4].

As visible in Table 7, 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.

References

1. Mcafee technical report on night dragon operation. Technical report (2015)
2. The CAIDA UCSD Anonymized Internet Traces 2018 (2018). Accessed 10 Feb 2020
3. Ayala, L.: Active medical device cyber-attacks. Cybersecurity for Hospitals and Healthcare Facilities, pp. 19–37. Apress, Berkeley, CA (2016). https://doi.org/10.1007/978-1-4842-2155-6_3
4. Clausen, H., Flood, R., Aspinall, D.: Traffic generation using containerization for machine learning. In: Proceedings of the Dynamic and Novel Advances in Machine Learning and Intelligent Cyber Security Workshop. ACM (2019)
5. Di Crescenzo, G., Ghosh, A., Kampasi, A., Talpade, R., Zhang, Y.: Detecting anomalies in active insider stepping stone attacks. JoWUA **2**(1), 103–120 (2011)
6. Ding, W., Le, K., Huang, S.-H.S.: Detecting stepping-stones under the influence of packet jittering. In: 2013 9th International Conference on Information Assurance and Security (IAS), pp. 31–36. IEEE (2013)
7. Donoho, D.L., Flesia, A.G., Shankar, U., Paxson, V., Coit, J., Staniford, S.: Multiscale stepping-stone detection: detecting pairs of jittered interactive streams by exploiting maximum tolerable delay. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516, pp. 17–35. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36084-0_2
8. EU ENISA. Baseline security recommendations for IoT in the context of critical information infrastructures (2017)
9. Fraser, G.: Tunneling, pivoting, and web application penetration testing. Technical report, SANS (2015)
10. Hemminger, S., et al.: Network emulation with netem. In: Linux Conference au, pp. 18–23 (2005)

[4] For all methods, the detection rate differences (0.7%–6.2%) were smaller across bandwidths than the overall detection rate errors (2.6%–6.5%).

11. Huang, S.-H.S., Kuo, Y.-W.: Detecting chaff perturbation on stepping-stone connection. In: 2011 IEEE 17th International Conference on Parallel and Distributed Systems, pp. 660–667. IEEE (2011)
12. Huang, S.-H.S., Zhang, H., Phay, M.: Detecting stepping-stone intruders by identifying crossover packets in SSH connections. In: 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), pp. 1043–1050. IEEE (2016)
13. Lee, R.M., Assante, M.J., Conway, T.: Analysis of the cyber attack onthe ukrainian power grid. Technical report, E-ISAC (2016)
14. Nasr, M., Bahramali, A., Houmansadr, A.: Deepcorr: strong flow correlation attacks on tor using deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1962–1976 (2018)
15. Padhye, J.D., Kothari, K., Venkateshaiah, M., Wright, M.: Evading stepping-stone detection under the cloak of streaming media with sneak. Comput. Networks **54**(13), 2310–2325 (2010)
16. Paxson, V., Floyd, S.: Wide area traffic: the failure of Poisson modeling. IEEE/ACM Trans. Networking **3**(3), 226–244 (1995)
17. Shullich, R., Chu, J., Ji, P., Chen, W.: A survey of research in stepping-stone detection. Int. J. Electron. Commer. Stud. **2**(2), 103–126 (2011)
18. Staniford-Chen, S., Heberlein, L.T.: Holding intruders accountable on the internet. In: Proceedings 1995 IEEE Symposium on Security and Privacy, pp. 39–49. IEEE (1995)
19. Tankard, C.: Advanced persistent threats and how to monitor and deter them. Network Secur. **2011**(8), 16–19 (2011)
20. Velan, P., Medková, J., Jirsík, T., Čeleda, P.: Network traffic characterisation using flow-based statistics. In: NOMS 2016 2016 IEEE/IFIP Network Operations and Management Symposium, pp. 907–912. IEEE (2016)
21. Wang, L., Yang, J.: A research survey in stepping-stone intrusion detection. EURASIP J. Wirel. Commun. Networking **2018**(1), 1–15 (2018). https://doi.org/10.1186/s13638-018-1303-2
22. Wang, X., Reeves, D.: Robust correlation of encrypted attack traffic through stepping stones by flow watermarking. IEEE Trans. Dependable Secure Comput. **8**(3), 434–449 (2010)
23. Wu, H.-C., Huang, S.-H.S.: Neural networks-based detection of stepping-stone intrusion. Expert Syst. Appl. **37**(2), 1431–1437 (2010)
24. Xin, J., Zhang, L., Aswegan, B., Dickerson, J., Daniels, T., Guan, Y.: 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. TRIDENTCOM 2006, p. 9. IEEE (2006)
25. Yang, J., Huang, S.-H.S.: Mining TCP/IP packets to detect stepping-stone intrusion. Comput. Secur. **26**(7–8), 479–484 (2007)
26. Yang, J., Woolbright, D.: Correlating TCP/IP packet contexts to detect stepping-stone intrusion. Comput. Secur. **30**(6–7), 538–546 (2011)
27. Yang, J., Zhang, Y.: RTT-based random walk approach to detect stepping-stone intrusion. In: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, pp. 558–563. IEEE (2015)