Formal Analysis of Sneak-Peek: A Data Centre Attack and Its Mitigations

Wei Chen, Yuhui Lin, Vashti Galpin, Vivek Nigam, Myungjin Lee, David Aspinall

Formal Analysis of Sneak-Peek: A Data Centre Attack and its Mitigations

Wei Chen ✉[1], Yuhui Lin[1], Vashti Galpin[1][0000−0001−8914−1122], Vivek Nigam[2], Myungjin Lee[1], and David Aspinall[1,3]

[1] University of Edinburgh, UK
{wchen2,ylin2,Vashti.Galpin,myungjin.lee,David.Aspinall}@ed.ac.uk
[2] Fortiss GmbH, Germany
vivek.nigam@gmail.com
[3] Alan Turing Institute, UK

Abstract. Attackers can exploit covert channels, such as timing side-channels, to transmit information without data owners or network administrators being aware. Sneak-Peek is a recently considered data centre attack, where, in a multi-tenant setting, an insider attacker can communicate with colluding outsiders by intentionally adding delays to traffic on logically isolated but physically shared links. Timing attack mitigations typically introduce delays or randomness which can make it difficult to understand the trade-off between level of security (bandwidth of the covert channel) and performance loss. We demonstrate that formal methods can help. We analyse the impacts of two Sneak-Peek mitigations, namely, noise addition and path hopping. We provide a precise mathematical model of the attack and of the effectiveness these defences. This mathematical analysis is extended by two tool-based stochastic formal models, one formalized in UPPAAL and the other in CARMA. The formal models can capture more general and larger networks than a paper-based analysis, can be used to check properties and make measurements, and are more easily modifiable than conventional network simulations. With UPPAAL, we can analyse the effectiveness of mitigations and with CARMA, we can analyse how these mitigations affect latencies in typical data centre topologies. As results, we show that using a selective strategy for path hopping is better than a random strategy, that using the two defences in conjunction may actually be worse than using a single defence, and we show the connection between hop frequency and network latency.

1 Introduction

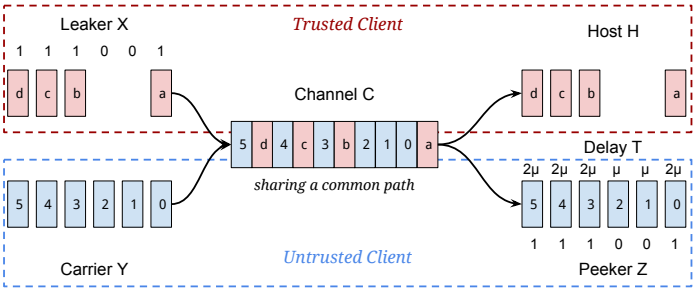
By exploiting covert channels, attackers can learn useful information without the data owners or network administrators realising that information has been leaked. Covert channels are wide-ranging and in general, may be difficult to detect and defend against. They include timing channels, such as the Spectre and Meltdown CPU vulnerabilities, where information is transmitted by delays in a computation or message timings [6, 12, 24]; storage channels, where information is transmitted by using shared locations; information is transmitted from electrical power consumption [7].

There are ways to identify potential covert channels [14]; then mitigation mechanisms can be deployed, such as blinding mechanisms to conceal transmitted data [3].

Mitigations typically penalise performance, however, imposing additional computation or adding artificial delays [9]. Therefore, before deploying a countermeasure, one must analyze the impacts it may have in the network, understanding the trade-offs between security and (network) performance. A security engineer needs to answer questions such as: *Which defences should I use?*, *Where should they be used?*, *Under which (traffic) assumptions will the defence work?*, *What will be the performance penalty?*

This paper shows that mathematical analysis and formal methods can help answer these questions. We study the Sneak-Peek attack introduced by Tahir et al [24]. Sneak-Peek is a high-rate covert channel in multi-tenant cloud computing environments. An insider attacker in a host of one client can transmit information to colluders in another client by intentionally adding delays to internal communications that happen to share a physical link with the outside colluders. The external colluders receive the signalled messages despite their machines being logically isolated from the first client.

The Sneak-Peek attack is illustrated and explained below:



- The *carrier* sends a constant stream of packets to the *peeker* using the channel. This traffic is allowed as both are outside the trusted network.
- The *leaker* sends a stream of packets to a host H inside the trusted network using the same channel.
- The leaker encodes data by adding delays to the stream: a delay encodes a binary '1' and no delay encodes '0'. The encoding is agreed by all colluding participants.
- Delays inserted by the leaker cause (blue) packets sent by the carrier to reach the peeker later. By measuring arrival times, the peeker can decode the leaked data.

Tahir et al suggested a defence mechanism for this attack based on *path hopping*: redirecting traffic dynamically to avoid the potentially compromised links. This defence is possible with Software-Defined Networks (SDNs) used in data centres: the SDN controller has a global view on traffic and can change routings dynamically under algorithmic control. Here we precisely analyse two defence mechanisms:

- **Background traffic:** additional (random) traffic interferes with timing channels by disrupting latency measurements used for signalling.
- **Path hopping:** the SDN controller can migrate network flows to different paths. Different paths have different delays and might not be shared externally, so path hopping mitigates Sneak-Peak timing channels.

Tahir et al considered the impact of background traffic on the covert channel, but without directly considering it as a mitigation in itself. They proposed several strategies for path hopping, but we take their analysis much further, as well as introducing stochastic formal methods to model the situation.

Contributions. In summary, our main contributions are:

1. A mathematical analysis using probability and information theory of the Sneak-Peek attack and the effectiveness of the two mitigations above.
2. A formal model using timed automata in the UPPAAL tool [4], which captures the Sneak-Peek attack, and can verify and measure transmission of data on the covert channel, including with background traffic and a simple form of path hopping.
3. A formal model built from parametric specifications in CARMA [17], which can capture more complex topologies and investigate the network latency imposed by different mitigation strategies. CARMA has a continuous-time Markov chain semantics and is designed to model collective adaptive systems.
4. Using the above, numerical results about channel bandwidth, attack effectiveness and network overheads. Results are compared between the mathematical analysis and tools as an internal validation.

Our work represents novel applications of the formal methods chosen, a novel combination of methods, and also obtains new results about the attack scenario.

Overview of paper. An outline of the paper and results is as follows:

- Section 2 starts with the simple topology (four hosts and two nodes) used in [24], and introduces our mathematical approach and first UPPAAL model. From the (different) mathematical analysis and formal model, we obtain measurements of channel capacity which agree with one another and with previous results in [24].
- Section 3 studies background traffic and its effect on the covert channel. The math and the UPPAAL model reveal that the success of the Sneak-Peek attack is sensitive to background traffic, suggesting its use directly as a mitigation (e.g., by injecting noise, or mixing traffic from elsewhere).
- Section 4 examines path hopping. Maths demonstrates the difference between a random path hopping strategy versus one which chooses a path deterministically: path hopping using a deterministic strategy among paths of different delay lengths can be better than hopping randomly. The UPPAAL model becomes more complex than the paper analysis: multiple paths and background traffic are modelled together, demonstrating how to answer some of the questions above. We show that using both defences together is not always better than either alone.
- Section 5 describes our parametric model in CARMA. A problem with timed automata is that for a new network topology, one needs specify the whole model from scratch. Instead, the CARMA implementation is based on a specification of the network, that is, its nodes and connections, and outputs a formal model which can be simulated or model-checked. This allows us to easily model large, more realistic networks. Our main goal with CARMA is to analyze the increase of network latency incurred by deploying defences on a typical data centre topology fragment (so-called 4-ary fat tree). Justified by our mathematical analysis, we propose strategies for path hopping: the key idea is to target flows over shared infrastructure that are likely to carry covert channels. We analyse this defence in CARMA on the data centre, assessing its impact on network latency. Simulation results here show that, in general, path hopping is a rather expensive mitigation.

Finally, Section 6 concludes the paper, mentioning some of the related work.

2 Simple timing channels

In the following, we provide a precise analytical model (Section 2.1) of the Sneak-Peek attack and we validate it using UPPAAL (Section 2.2).

2.1 An analytic model

Let us assume that the leaker X sends packets to the host H with probablity $p = P(X = 1)$. The carrier Y sends packets to the peeker Z constantly, i.e., $P(Y = 1) = 1$. Here, we formalise each entity as a random variable. The *delay* T denotes: the difference between departure and arrival times of a packet from Y to Z if packets from Y can only be delayed by packets from X; otherwise, the inter-arrival time for packets from Y. The peeker recovers the secret sent by X from T using the rule: $Z = 1$ if $T > \theta$ and $Z = 0$ if $T \leq \theta$, with θ a threshold on T.

The channel capacity [8, Chapter 7] is the maximum mutual information between X and Z over the probability distribution of X. We calculate it as follows. First, we assume that given X the delay T follows the normal distribution: $T|(X = i_{\in\{0,1\}}) \sim \mathcal{N}((i+1)\mu, \sigma^2)$, where μ is the average delay for a packet from Y going through the channel, and σ is the standard deviation. Then, the probabilities of decoding errors are:

$$\begin{aligned} \text{err}_0 &= P(Z = 1|X = 0) = P(T > \theta|X = 0) = 1 - \Phi((\theta - \mu)/\sigma) \\ \text{err}_1 &= P(Z = 0|X = 1) = P(T \leq \theta|X = 1) = \Phi\left((\theta - 2\mu)/\sigma\right) \end{aligned}$$

where Φ is the cumulative distribution function of the standard normal distribution. We have the mutual information $I(X; Z)$ between X and Z:

$$\begin{aligned} I(X; Z) &= \sum_{X,Z} P(Z|X)P(X) \log \frac{P(Z|X)}{P(Z)} \\ &= H(\text{err}_0 - (\text{err}_0 + \text{err}_1 - 1) \times p) - (1 - p)H(\text{err}_0) - pH_2(\text{err}_1) \end{aligned}$$

where $H(x) = -x \log x - (1 - x) \log(1 - x)$. The capacity $C_{\in[0,1]}$ of this channel is achieved when $\frac{dI(X;Z)}{dp} = 0$. Since we have no idea of what kind of secret will be sent, it is unreasonable to claim that the probability p depends on decoding errors $\text{err}_{i\in\{0,1\}}$. A feasible assumption on the probability distribution of X is $p = 0.5$, i.e., the same probability for 0s and 1s (the message may well be compressed or encrypted). Note that this is not a conventional binary symmetric channel [8], because there are different decoding errors.

Example 1. We assume that the average delay is 40ms if X doesn't send a packet and 80ms if X sends a packet. The standard deviations both are 10ms. We set the threshold θ to 60ms at which the total error $\text{err}_0 + \text{err}_1$ achieves the minimum value 0.0456. Then, the capacity is 0.843 which is achieved when $p = 0.5$. Shannon's Theorem [23] tells us that the channel capacity is the maximum achievable information rate. So, an upper bound of bandwidth (KB/s) for this channel is: $(B/(1+pS))C$, with S the average size (KB) of packets sent by X and Y, and B the average bandwidth (KB/s) of physical links available for this channel. If the packet size is 64KB on average and the bandwidth is 1GB/s, the upper bound of bandwidth is 1.12 KB/s.

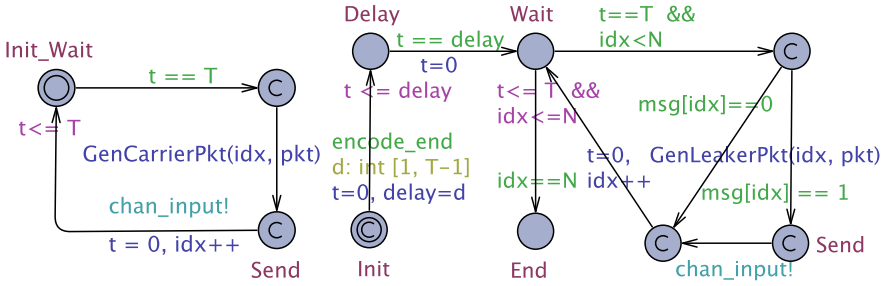


Fig. 1. Timed automata for the carrier (to the left) and the leaker (to the right).

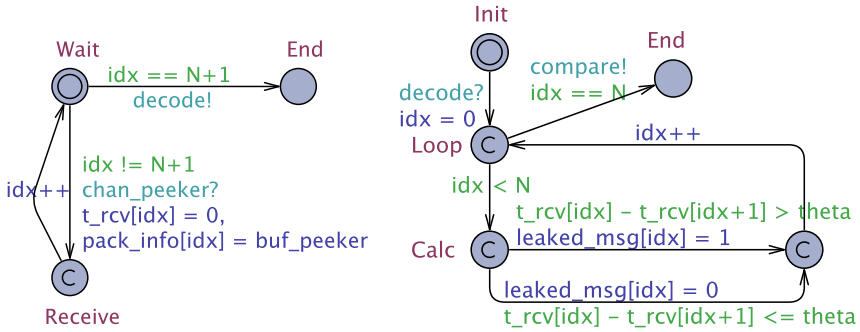


Fig. 2. Timed automata for the peeker (to the left) and the decoding process (to the right).

2.2 A formal model in UPPAAL

While the mathematical analysis provides precise analysis, the combinatorial problem involved explodes with the increase of network paths. We now start to address this problem, by manually formalising simple timing channels in UPPAAL (Figs 1,2). We simulate the Sneak-Peek attack and validate the mathematical analysis given in previous section.

The simple channel is modelled as a queue with a delay following a normal distribution with mean mu and standard deviation 1. Fig. 1 (left) models the carrier. Initial state Init_Wait is denoted by a double circle. Invariant t <= T, where t is a local clock, ensures that a transition has to occur within T time units. Sending a packet is modelled by storing a packet in a shared variable buf_input, followed by signalling a synchronisation by chan_input!. ! denotes triggers of a synchronization, and ? denotes the receiver. A state with a C is a *committed state*, which constrains that no time elapses when it is involved in a transition. With committed states, it can be ensured that time intervals of sending packets is exactly T time units. Similarly, in Fig. 1 (right), for every T time units, the leaker decides whether to send a packet to the channel or not depending on the current bit of the message to be leaked. The leaker and the carrier are not synchronised in reality. This is specified in our model by the additional delays added for the leaker. As shown in Fig. 2 (left), the peeker records the arrival time for each packet with clocks t_rcv[i]. The message is recovered by calculating inter-arrival in Fig. 2 (right).

Running simulation with {N = 10, T = 4, mu = 4, theta = 6} in UPPAAL, we get an average accuracy 94.5% out of 7598 runs. That is, the decoding error rate is 0.055 which is close to the value (0.0456) produced by mathematical analysis in Section 2.1.

3 Timing channels with background traffic

In this section, we extend the mathematical model and the UPPAAL model for simple timing channels with background traffic.

3.1 Modelling background traffic

We assume that Y sends a packet every t seconds. Let N be the random variable characterising the background traffic, namely, the number of packets from the background traffic within a time interval t. We assume that N follows the Poisson distribution: $P(N = k) \sim e^{-\lambda}\frac{\lambda^k}{k!}$, with λ the average number of packets from the background traffic within a time interval t. Let us suppose that given X and N the delay T follows the normal distribution: $T|(X = i_{\in\{0,1\}}, N = k_{\in\mathbb{N}}) \sim \mathcal{N}((k+i+1)\mu, \sigma^2)$. When X and N are independent, the probabilities of decoding errors are:

$$\begin{aligned}
\text{err}_0 &= P(Z=1|X=0) = \sum_k P(T>\theta|X=0 \wedge N=k)P(N=k) \\
&= e^{-\lambda}\sum_k \left(1-\Phi\left(\frac{\theta-(k+1)\mu}{\sigma}\right)\right)\frac{\lambda^k}{k!} \\
\text{err}_1 &= P(Z=0|X=1) = e^{-\lambda}\sum_k \Phi\left(\frac{\theta-(k+2)\mu}{\sigma}\right)\frac{\lambda^k}{k!}
\end{aligned}$$

It is not necessary that N follows the Poisson distribution. The decoding errors can be customised by using a distribution from the empirical study of network traffic in data centres, e.g., Log-normal and Weibull distributions [5].

Example 2. Let us suppose that Y sends a packet every t seconds and the average number of packets from the background traffic is $\lambda = 0.5$ within the time interval t. We assume that within the time interval t the maximum number of packets from the background traffic is $K = 3$; and $T|(X, N)$ follows the normal distribution with $\mu = 40$ms and $\sigma = 10$ms. Let us set the threshold θ to $\frac{3}{2}\mu = 60$ms. The probabilities of decoding errors are $\text{err}_0 \approx 0.3986$ and $\text{err}_1 \approx 0.0138$. The mutual information between X and Z with $P(X = 1) = 0.5$ is 0.3528. That is, the background traffic at the level of $\frac{0.5}{t}$ packets/s, reduces the upper bound of information rate from 0.843 to 0.3528. If the bandwidth of physical links is $B = 1$GB/s and the average size of packets is $S = 64$KB, then an upper bound of the channel bandwidth is: $(B/((1 + 0.5 + \lambda)S))C \approx 2890.1$ bits/s ≈ 0.3528 KB/s.

The left graph in Fig. 3 demonstrates how the background traffic affects the capacity of simple timing channels. Here, the symbol λ denotes the average number of packets from the background traffic within a time interval 0.1s. When λ is 0, i.e., there is no background traffic, the mutual information $I_{0.5}(X; Z)$ achieves the highest value 0.843. This value drops quickly when λ increases. This is mainly due to the increase of the probability err_0 of decoding errors. The peeker can set the threshold θ according to the parameter λ, so as to reduce the effect caused by the background traffic, e.g., setting θ to $(1 + \lambda)\mu$. However, the recovery of channel capacity is limited, because the packets from the background traffic dominate the delay T when λ increases.

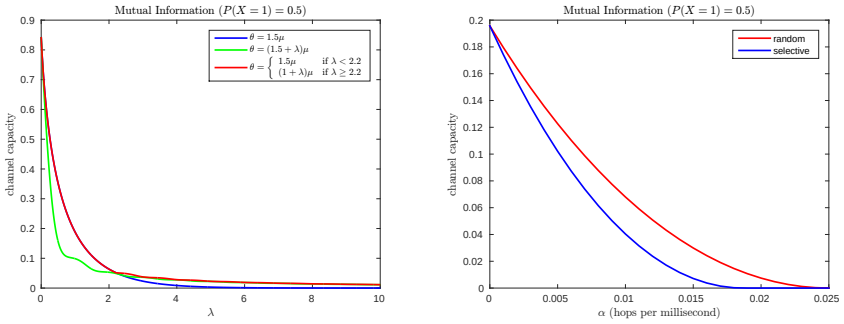


Fig. 3. The timing channels are sensitive to background traffic (left). The selective path hopping is more effective than the random path hopping when the threshold is fixed (right).

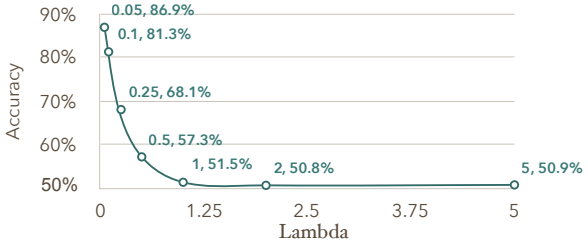


Fig. 4. Simulation of impact of background traffic to leakage accuracy.

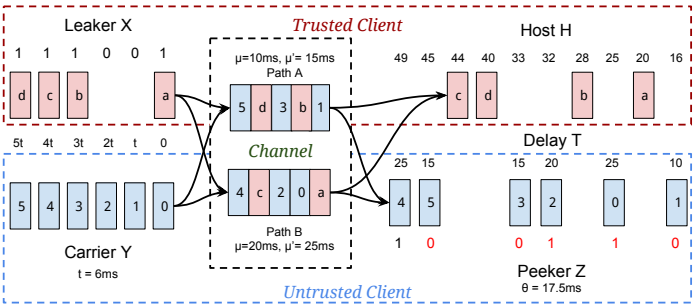
3.2 Extending the UPPAAL model

We extend the formal model given in Section 2.2 by adding a timed automaton for background traffic which sends a packet to the channel at an exponential rate of lambda. By running simulation with {N = 10, T = 4, mu = 4, theta = 6} and different lambda, we summarise the relation between lambda and accuracy in Fig. 4. The accuracy drops dramatically when background traffic rate increases from 0 to 1.25, but remains at the same level afterwards. Also, the accuracy does not drop below 50%, because background traffic can only introduce additional delays to turn 0 to 1, but cannot shorten the network delay to convert 1 to 0. This is consistent with our mathematical analysis results depicted in Fig. 3.

4 Mitigating timing channel attacks

We extend our mathematical model and UPPAAL model with path hopping. Path hopping is a timing channel mitigation mechanism in SDN. The assumption behind this method is: if trusted and untrusted networks share several paths, then hopping between these paths will reduce the capacity of timing channels. We illustrate its use next.

Example 3. Consider the extension of the simple timing channel depicted in the above figure with two paths A and B shared between trusted and untrusted networks. Assume that the average delay of a packet from Y going through A and B is respectively 10ms and 20ms and from X is, respectively, 15ms and 25ms. We assume that X and Y send



packets at the same rate, e.g., X sends zero or one packet and Y sends one packet every 6ms. Once a packet is received by the switch, it chooses a path from A and B and sends the packet using the chosen path. Assume packets which are sent at the time n with n odd, will use path A; and other packets will use path B. Setting the threshold θ to the average of delays, i.e., 17.5ms, the peeker Z can recover the secret from delays by applying the rules given in Section 2.1. However, using two paths A and B will affect the packet arrival time, e.g., the packets 1 (16ms) and 5 (45ms) respectively arrive earlier than the packets 0 (25ms) and 4 (49ms), the peeker Z actually gets the bit sequence 011001 which contains five decoding errors, comparing to the original bit sequence 100111 sent by X. So apart from noise caused by delays, path hopping introduces another kind of noise: as bits may be received out-of-order.

4.1 A mathematical analysis of path hopping

Assume that the delay of a path A follows the normal distribution, i.e., $T_A|(X=0) \sim \mathcal{N}(\mu_A, \sigma_A^2)$ and $T_A|(X=1) \sim \mathcal{N}(\mu'_A, \sigma_A^2)$. The probabilities of decoding errors of the path A are err_0^A and err_1^A. Let q denote the probability of disorder for a packet sent by Y. By disorder we mean that the nth packet arrives later than the mth packet when $m > n$, which only happens when a flow is switched from a slower to a quicker path. The decoding errors are:

$$\text{err}_i = (1-q) \times \sum_A (P(A) \times \text{err}_i^A) + q \times P(Z = 1-i)\,.$$

By solving the above equations we have:

$$\text{err}_i = (1-q) \times E_A(\text{err}_i^A) + \frac{q}{2} \times \left(1 + E_A(\text{err}_i^A) - E_A(\text{err}_{1-i}^A)\right)$$

where E_A denotes the expectation ranging over paths. The average radius $\bar{r}$ of disorders caused by a hop from a slower path A to a quicker path B is $E_{(A,B)}(\lfloor(\mu_A + \mu'_A - \mu_B - \mu'_B)/(2t)\rfloor)$. Let α be the number of hops in one millisecond and β be the probability of hopping from a slower to a quicker path. We approximate q by: $\min(1.0, \frac{\alpha \times \beta \times 2\bar{r}}{1/t}) \approx \min(1.0, \alpha\beta \times E_{(A,B)}(\mu_A + \mu'_A - \mu_B - \mu'_B))$.

Example 4. Consider the collection $\{(\mu_A = n \times 20\text{ms}, \mu'_A = 2n \times 20\text{ms}) \mid 1 \le n \le 3\}$ of paths with standard derivation $\sigma = 5$ms. Suppose that every path has the same chance to be chosen, i.e., $P(A) = \frac{1}{3}$. Setting the threshold θ to the average delays of these

paths, i.e., 60ms, we get the average probabilities of decoding errors: $E_A(\text{err}_0^A) \approx 0.174$ and $E_A(\text{err}_1^A) \approx 0.333$. Assuming that every hop from a slower path to a quicker path has the same probability $\frac{1}{3}$, we get the probability q of disorders: $\min(1.0, \alpha\beta \times 80)$. We can randomly hop from one path to another in which the probability β of hopping from a slower to a quicker path is $\frac{1}{2}$. A better hopping strategy is the path selective, e.g., following the hop sequence $CBAC$ with C is slower than B and B is slower than A. The selective strategy gives us a higher probability $\beta = \frac{2}{3}$. Let α range over $[0, 0.025]$, i.e., from zero hop to 25 hops per second. We calculate the channel capacity and show the results as the right graph in Fig. 3. By increasing the number of hops the channel capacity can be reduced to 0. In this example, zero capacity is achieved respectively at 19 (selective) and 25 (random) hops per second.

4.2 Modelling and analysing path hopping in UPPAAL

We further extend our UPPAAL model by adding path hopping (random replacement and random selection [24]). Our analysis demonstrates that the effectiveness of path hopping is reduced with the increase of background traffic and the decrease on the number of flows. So the (expensive) path hopping defence should not be used when background traffic is high and especially when there are few flows.

A K-path channel is abstracted as K independent queues with an individual delay mu. Each queue also has a path id ranging from 0 to K - 1. Assuming that there are N_FLOWS flows in the network, each packet is assigned with a flow id ranging from 0 to N_LOWS - 1, where 0 and 1 are reserved for the carrier and the leaker, and the remaining ones are for background traffic. Array flow_path[N_FLOWS] is introduced to associate a flow to a path so that packets can be distributed to the right queue.Fig. 5 (right) models path hopping mitigation. For every time period of alpha, each path flips an equal coin to deicide whether to update with a random path id in flow_path, or not.

We run two groups of simulation with common parameters of {N = 10, T = 4, mu = 4, theta = 6, K = 8, alpha = 10} and different numbers of background traffic flows (#bgt), i.e. 2 and 10. The results are depicted in Fig. 6 (left) together with the data from Fig. 4 (the accuracy with background traffic only) for comparison. The effect of path hopping without background traffic, i.e. 55%, is also shown as a reference line. Fig. 6 (right) zooms in the area of $1 \leq$ lambda ≤ 5. The results show that path hopping can reduce the accuracy significantly when $0 \leq$ lambda ≤ 1. However, when $1 \leq$ lambda ≤ 5 and background traffic has decreased the accuracy to a low range, the impact of path hopping becomes very small. Moreover, in the case of #bgt = 2, the accuracy goes up slightly comparing to the result with background traffic only. The following table shows the simulation results of the relation between the accuracy and #bgt when lambda = 5.

#bgt	1	2	3	4	6	8	10	12
Accuracy	53.0%	52.3%	51.5%	51.0%	50.8%	50.7%	50.8%	50.8%

When #bgt gets smaller, the accuracy becomes closer to the case when applying path hopping without background traffic. The cause is that, when #bgt is small, path hopping is more likely to reduce the amount of background traffic in the channel where the attacker flow stays, and weakens the impact of background traffic.

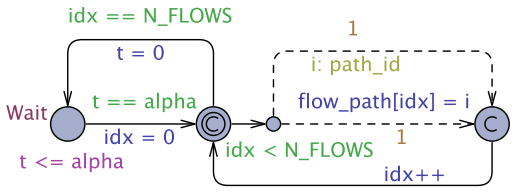


Fig. 5. Timed automata for path hopping.

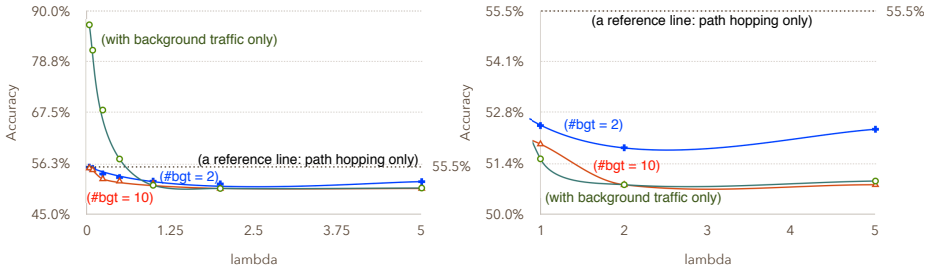


Fig. 6. Impact of path hopping with different number of background traffic flows (#bgt). The figure on the right zooms in the area of where left one when $1 \leq \text{lambda} \leq 5$.

Our results tell that path hopping keeps the accuracy in a low range. However, blindly applying might not always have positive impact, especially when background traffic rate is high and the #bgt is low.

5 Considering network topology

We now consider a different approach using the quantitative modelling language CARMA [17]. CARMA has semantics based on continuous-time Markov chain semantics and is supported by the CARMA Eclipse Plug-in [11]. CARMA has an expressive syntax for directed graphs making it appropriate and efficient for network modelling. This allows for a generic model which is parametrised by a specific network configuration using the syntax. An example is:

```
space network_name (){
nodes { [Sw1]; [Sw2]; [Sw3]; }
connections { [Sw1] -> [Sw2] {port = 1}; ... }}
```

This specifies three switches and a link from the first switch to the second switch via port 1. Nodes have type `location` and there are various operators defined over this type to obtain pre-sets, post-sets and edge weights such as `port`. A manually constructed space model can be used to obtain the topology of the network semi-automatically during simulation of the CARMA model and automatic generation can be implemented in a straightforward manner.

We present two scenarios: the first is similar to the UPPAAL model where we consider the probability of shared network infrastructure, and the second is the fat-tree topology used in data centres where we consider the cost of mitigation.

Fig. 7 illustrates the results of using MultiVeStA [22], a statistical model checker integrated into the CARMA command line tool, which can assess the probability of events or mean of values of interest expressed as temporal logic formulae. We use a

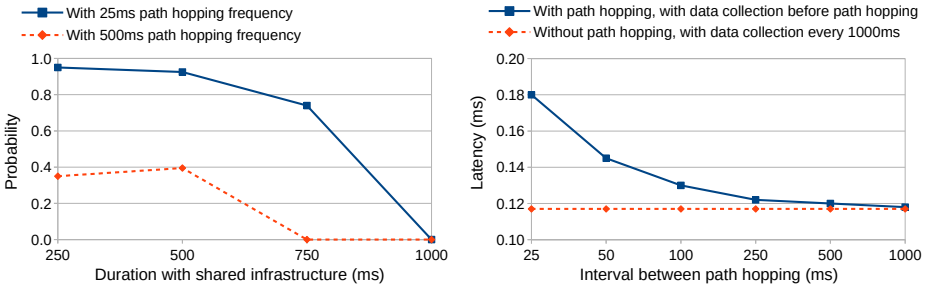


Fig. 7. Probability of network infrastructure shared by attackers (left) and comparison of latency with and without path hopping (right)

formula that describes the probability of the sneaker and peeker sharing the one of the two switches for a given duration under random path hopping (where both selection and placement are random). As can be seen with frequent path hopping (every 25ms), there is a high probability that there will be shared infrastructure for at least one duration of 250ms; however, the probability of shared infrastructure for a period of 1s is negligible. With less frequent path hopping (at every half second) it is less likely that the durations of shared infrastructure occur.

We also want to assess the impact of the path hopping mitigation on the network performance for a larger networking scenario using SDN, and thereby evaluate the trade-offs between security and performance. Working with an SDN model, we can quantify the cost of path hopping by considering different frequencies of rule updates (which are necessary to determine new routes at the switch level). We measure overall packet latency (the average time taken by a packet from host to destination) to determine the reduction in performance caused by this mitigation. Fat-trees [1] improve on a single-rooted tree by providing multiple paths between hosts, and the size of the network depends on the number of ports in a switch. for k ports, there are $5k^2/4$ switches supporting $k^3/4$ hosts and providing $k^2/4$ paths between each pair of hosts.

Our model describes a 4-ary fat-tree where hosts generate and accept packets, the switches route packets according to their rule tables. The controller installs the flow rules in the switches initially, requests traffic information from the switches and determines when to switch flows. Packet generation is determined by exponential distributions, and can be specified for each host individually. Previous research shows that the time taken for 10 rule updates [21] varies from 1 ms (milliseconds) to 20 ms depending on the type of switch. In our model, we assume that the time taken is exponentially distributed with average 1 ms. We have two clients, and in each client there is low traffic between all hosts and one large flow between two hosts which traverses infrastructure that may be shared.

Fig. 7 shows the costs of path hopping (with random selection and placement) in terms of how it affects latency. We assume that data collection from switches is done before path hopping to allow for non-random selection and placement of flows, and we include this cost even in the random case. The blue line in the graph shows how as the frequency of path hopping decreases so does the latency. At a path hopping interval of 1 second, the impact is negligible compared with latency for no path swapping and data collection every 1 second. However, with more frequent path hopping every 25ms, there is a 50% increase in latency.

Earlier results in Section 4.1 show that with a rate of around 20 hops/s (depending on the type of path hopping used) without background traffic, a covert channel's capacity can be reduced to zero. But this is very expensive in terms of the cost of switch updating. Even at a path hopping frequency of 100ms, there is a penalty of 10% to latency. This suggests that any potential flow selection and/or placement procedure that reduces the necessity of frequent path hopping would improve path hopping as a mitigation for covert channel attacks, such as selecting a flow route which includes network elements that may be shared. Additionally, we could investigate approaches to mitigation that are not possible in current switches such as choosing probabilistically between two routes at the switch-level. Approaches suggested here might also be combined with load balancing methods such as CONGA [2].

6 Conclusions and related work

We set out to investigate data centre attacks and their mitigations by applying rigorous and formal methods. We extended the initial work on the Sneak-Peek attack [24] by applying a mathematical analysis feeding in to two formal models, the UPPAAL model which captures low-level details of the covert channel between a small number of nodes, and the CARMA model which extends to larger data centre topology fragments. These models let us explore new questions which, in conventional networking research, the burden of full simulation (or real experiments on testbeds) would have made overly time consuming to explore. Some high-level conclusions were:

- a selective strategy for path hopping mitigation can beat a random strategy;
- using path hopping together with background traffic mixing may be worse than using either mitigation alone;
- on data centre topologies, even with SDN, over-enthusiastic use of path hopping may incur an unreasonably large cost on network latency.

Taken together, our investigation suggests that the most functional and cost-effective mitigations should take into account topology and existing traffic flows. Thus, SDN-based mitigations may be appropriate but, at least for the Sneak-Peek example, may need to be more intricate than some of the algorithms deployed so far.

In general, we believe that our "mixed formal methods" approach may be useful for similar security problems in applied networking and network security and we are eager to apply them further. Ultimately, for a solution being considered for real deployment, we would of course want to validate the findings with testbeds or field experiments.

Limitations. The attacker may try to determine the best threshold dynamically when mitigations are deployed. In that case the channel capacity will vary but it will be bounded by the limits of the model without path hopping, moving between similar curves like shown in Fig. 3. To completely model and analyse the situation with strategies for dynamic threshold setting, one would need significantly more complex mathematical and formal models. Our UPPAAL model would allow this latter exploration rather effectively. In theory, the channel capacity can be reduced to zero, as shown in Example 4. However, completely eliminating covert channels is impossible in practice.

Related work. Applying formal methods to networking has gained considerable interest in the last decade, including use of programming language techniques for describing networks and SDN, as well as formal verification and model-checking applied to ensuring properties. An example is Kuai, a model checker for safety and network-consistency properties of SDN [18]. Network components are specified generically as in our CARMA approach, and the network topology is specified separately. This research considers whether classical Boolean safety properties are true of all traces, but cannot assess these properties probabilistically, unlike our approach.

Statistical model checking of discrete-time probabilistic models has been used to evaluate different selective strategies for mitigating telephony Denial of Service (DoS) attacks [16], and an distributed DoS attack on the TCAM memory of SDN switches [19]. Our research considers covert channel attacks rather than (D)DoS and we focus on continuous-time reasoning allowing for assessment of performance as well as reasoning about the attack. Translating UML models to UPPAAL has been used for reasoning about SDN [20]. Our approach takes a more direct approach to specify topologies which allows straightforward scaling up of network size.

To disrupt timing channel attacks in cryptographic schemes, bucketing which involves accumulation of messages followed by batch sending [15, 9] and evaluation of randomised countermeasures using SMT solvers [10] are two techniques which have been applied. This research has similarities to the path hopping mitigation but is applied in a different context. Researchers have proposed a defence for web-fingerprinting attacks in anonymised networks where bursts of packets are padded to make them less traceable [13]. Our methods might also help to study trade-offs in anonymity networks like TOR and we are currently considering how our approach can be applied to the Meltdown-like attacks.

Acknowledgements Our work is supported by EPSRC project EP/L02277X/1 and the Alan Turing Institute. The UPPAAL and CARMA models and experimental data are available at the web site `http://groups.inf.ed.ac.uk/security/RasE/`.

References

1. M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proceedings of ACM SIGCOMM 2008*, pages 63–74, 2008.
2. M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. The Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese. CONGA: distributed congestion-aware load balancing for datacenters. In *Proceedings of ACM SIGCOMM'14*.
3. M. Backes and B. Köpf. Formally bounding the side-channel leakage in unknown-message attacks. In *Proceedings of ESORICS 2008*, pages 517–532, 2008.
4. G. Behrmann, A. David, and K.G. Larsen. A tutorial on UPPAAL. In *Revised lectures from SFM 2004*, pages 200–236. 2004.
5. T. Benson, A. Akella, and D.A. Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 267–280. ACM, 2010.
6. Arnab Kumar Biswas, Dipak Ghosal, and Shishir Nagaraja. A survey of timing channels and countermeasures. *ACM Comput. Surv.*, 50(1):6:1–6:39, 2017.

7. S.S. Clark, H.A. Mustafa, B. Ransford, J. Sorber, K. Fu, and W. Xu. Current events: Identifying webpages by tapping the electrical outlet. In *Proceedings of ESORICS 2013*, pages 700–717, 2013.
8. T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 2006.
9. Y.G. Dantas, T. Hamann, H. Mantel, and J. Schickel. An experimental study of a bucketing approach. In *Proceedings of QAPL 2017*, pages 517–532, 2017.
10. H. Eldib, C. Wang, and P. Schaumont. Formal verification of software countermeasures against side-channel attacks. *ACM Trans. Softw. Eng. Methodol.*, 24:11:1–11:24, 2014.
11. J. Hillston and M. Loreti. CARMA eclipse plug-in: A tool supporting design and analysis of collective adaptive systems. In *Proceedings of QEST 2016*, pages 167–171, 2016.
12. G. Ho, D. Boneh, L. Ballard, and N. Provos. Tick tock: Building browser red pills from timing side channels. In *Proceedings of WOOT '14*, 2014.
13. M. Juárez, M. Imani, M. Perry, C. Díaz, and M. Wright. Toward an efficient website fingerprinting defense. In I.G. Askoxylakis, S. Ioannidis, S.K. Katsikas, and C.A. Meadows, editors, *Proceedings of ESORICS 2016*, pages 27–46, 2016.
14. R.A. Kemmerer. A practical approach to identifying storage and timing channels: Twenty years later. In *Proceedings of ACSAC 2002*, pages 109–118, 2002.
15. B. Köpf and M. Dürmuth. A provably secure and efficient countermeasure against timing attacks. In *Proceedings of IEEE CSF 2009*, pages 324–335, 2009.
16. M.O.O. Lemos, Y.G. Dantas, I.E. Fonseca, and V. Nigam. On the accuracy of formal verification of selective defenses for TDoS attacks. *JLAMP*, 94:45–67, 2018.
17. M. Loreti and J. Hillston. Modelling and analysis of collective adaptive systems with CARMA and its tools. In *Proceedings of SFM 2016*, pages 83–119, 2016.
18. R. Majumdar, S.D. Tetali, and Z. Wang. Kuai: A model checker for software-defined networks. In *Proceedings of FMCAD '14*, pages 27:163–27:170, 2014.
19. T.A. Pascoal, Y.G. Dantas, I.E. Fonseca, and V. Nigam. Slow TCAM Exhaustion DDoS Attack. In *Proceedings of IFIP SEC 2017*, pages 17–31, 2017.
20. V.V. Podymov and U.V. Popesko. UPPAAL-based software-defined network verification. In *Proceedings of TMPA '13*, pages 9–14, 2013.
21. C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A.W. Moore. OFLOPS: an open framework for openflow switch evaluation. In *Proceedings of PAM 2012*, pages 85–95, 2012.
22. S. Sebastio and A. Vandin. MultiVeStA: statistical model checking for discrete event simulators. In *Proceedings of ValueTools '13*, pages 310–315, 2013.
23. C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5:3–55, 2001.
24. R. Tahir, M. T. Khan, X. Gong, A. Ahmed, A. Ghassami, H. Kazmi, M. Caesar, F. Zaffar, and N. Kiyavash. Sneak-peek: High speed covert channels in data center networks. In *Proceedings of IEEE INFOCOM 2016*, pages 1–9, 2016.