Covert Timing Channel Considering Execution Time Distribution in Real-Time Systems

Jaeheon Kwak
College of Software
Sungkyunkwan University(SKKU), Republic of Korea
Email: 0jaehunny0@gmail.com

Jinkyu Lee
College of Software
Sungkyunkwan University(SKKU), Republic of Korea
Email: jinkyu.lee@skku.edu

***Abstract*—Indispensable rule of real-time systems is that systems must guarantee any task's jobs to meet their own deadlines. To ensure this rule even in worst case, real-time systems get task's worst case execution time (WCET) and schdule task as all tasks always take exactly WCET. In practice, however, there's a significant difference between real exection time and WCET. In this paper we present a novel covert timing channel, which considers task's real execution time distribution to decrease its error rate. We prove that it is possible to leak information in multi-level security (MLS) real-time systems. In addition, we propose four methods to defend or detect the covert timing channel.**

I. Introduction

In real-time systems, failure of any task to meet deadline cause disastrous damage. Real-time systems schedule tasks for every task to meet its deadline, even if whole tasks takes their worst case execution time (WCET). Scheduler gets WCET information about each task, and determine task's priority in schedule with task's period, deadline, and WCET. RM (Rate Monotonic), EDF (Earliest Deadline First), LLF (Least Laxity First) [1], [2] are well known real-time scheduling algorithms. All those kinds of scheduling algorithms regard execution time of tasks as WCET. In practice, however, execution time of tasks considerably have difference with WCET. Many researchers tried to analyze WCET [3], [4], [5], and they revealed that only precious few tasks run as much as WCET and most of tasks run about half of WCET. Figure 1 shows execution time distribution for common tasks [3].

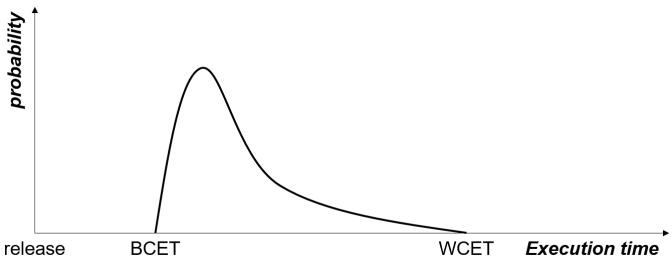


Fig. 1: Execution time distribution for synthetic task [3]

Because of predictability of real-time system, the systems are exposed to timing inference attack, such as side channel, covert timing channel. And several studies suggested attack methods [6], [7], [8] or defence methods [9], [10], [11], [12]. However, until now, there was not much focus on new attack mechanisms. In this paper, we present a novel covert timing channel attack. A paper have already analyzed covert timing channel in multi-level security (MLS) [13] real-time systems [7] and it has limits of the channel accuracy and it only analyzed the channel under rate monotonic scheduling.

Our new covert timing channel considers real execution time distribution to increase channel accuracy and can be used for any scheduling algorithms. It gathers information of other tasks, and guesses estimated execution time distribution. Sender and receiver flow information with controlling their tasks execution time, and reduce error of bit transmission, which is occurred by interruption of other tasks between reciever and sender's job, with calculation of estimated execution time distribution stochastically.

This paper makes following contributions:

1) We introduce a novel covert timing channel and analyze it, which considers real execution time to extend the channel accuracy
2) We find problems of our methods and suggest solutions for it
3) We implement partial simulation for our covert timing channel model and evaluate its performance.
4) We propose 4 defence methods which can block or detect our covert timing channel.

II. Related Work

Son and Alves-Foss have already designed covert timing channel in MLS real-time systems. Their approach was checking response time of tasks. This approach requires two tasks. First task, T_H has higher security level and higher priority than second task and works as sender. Second task, T_L has lower security level and lower priority than first one and works as receiver. They have appointment that if T_H extends its execution time, than it means it sends a bit '1' to receiver, if T_H shrink its execution time, it means it sends a bit '0' to receiver.

During this transmission, other tasks will take place between T_H and T_L, and it cause noise of this channel. In figure 2, noisy tasks T_N which have higher priority than T_H or T_L locate between T_H and T_L. So the response time of T_L is

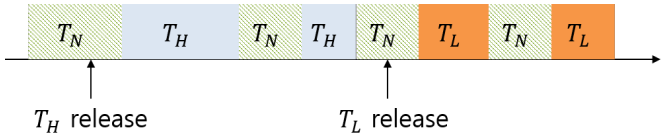


Fig. 2: Interruption between T_H and T_L by T_N

also affected by T_N, not only T_H. Therefore, T_L cannot infer what bit T_H sent in some cases and it could get error bits. To solve this problem, some new method which can reduce not deducible response time.

III. Covert Timing Channel

A. Covert Channel and Covert Timing Channel

Covert timing channel is a kind of covert channel. Before we explain covet timing channel, we should explain covert channel. Covert channel means a channel which enables unauthorized information flow [14]. It locate between two users like figure 3 and leak data covertly. Administrator should prevent it, but the covert channel is hard to detect. It usually send and receive data abnormally. For example, by checking whether some file exist or not, they send series of bits, 1 and 0 [15]. And the covert timing channel is a channel which sends data by controlling allocated system resource. In this paper we send bit data by controlling execution time.

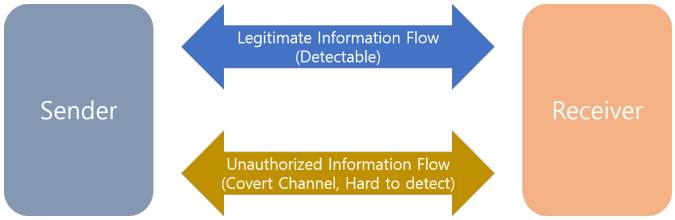


Fig. 3: Concept map of covert channel

B. Model and Mechanism

Our covert timing channel's basic mechanism is similar with Son and Alves-Foss's work [7]. We consider uni-processor system which use preemptive scheduling. And we modeled sender and receiver, and their ability and goal as following:

- Sender, T_H: Sender knows receiver task's period and time when its job starts running and when its job finishes at that moment. It can control its execution time within its WCET range. It has higher security level than receiver and wants to send data, unauthorized to lower security level users, to receiver.
- Receiver, T_L: Receiver knows sender task's period and time when its job start running at that moment. It has lower security level than sender and wants to get unauthorized data from sender.

They form covert timing channel by controlling execution time of T_H. They appointed that if sender extend its execution time, it means sending a bit 1, and other case, a bit zero, in advance. Because there is any other way sender and receiver to communicate, sender cannot know receiver got value well and receiver cannot not whether the received bits are correct or not. Also due to noisy task's work, transmission error is inevitable. We introduce a novel method can solve those problems pretty well in section 6.

IV. Execution Time Distribution

All tasks must meet their deadlines in real-time systems, albeit all tasks take WCET and have worst case. Thus, task scheduler think task's execution time as its WCET from the very first. But its execution time in actual environment is hugely diffrent from WCET. Several studies analyzed it [3], [4], [5] and its general distribution shape like figure 1. In this paper we assume that task's execution time distribution is exact with gumbel distribution [16], which is suggested by Edgar and Burns [5]. Gumbel distribution's equation is

$$f(x) = e^{-(x+e^{-x})}$$

and its probability density function and cumulative density function graph looks like figure 4 when its location is 1 and scale is 0.

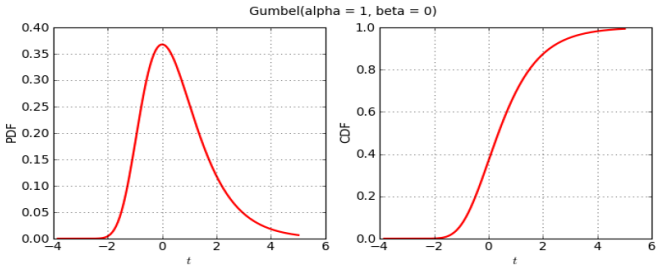


Fig. 4: PDF and CDF grpah of gumbel distribution

The key point is that scheduler schedules tasks as tasks take WCET all the time but their execution time close to half of WCET in practice, and if task's job finish earlier than its WCET, than next job will start even if past job's appointed execution time is not end yet. Son and Alves-Foss regarded noisy channel's activity as just error, so attacker cannot use covert timing channel when error rate is too high to leak data. To reduce error rate, we will utilize task's real execution time distribution, predict and calculate interruption of T_N.

V. Execution Time Distribution Inference

A. Least Common Multiplying Detecting

Sender and receiver can determine their execution time, but cannot predict other task's execution time. So we will used prediction via schedule's Least Common Multiple (LCM) and execution time distribution. There are some researches studied method to predict or analyze schedule's LCM [11], [17], [18]. And because of real-time systems' predictability, it's not hard to find schedule's period(LCM). Algorithm 1 is a naive method to find schedule's LCM via checking release time and start time. If system didn't prepared any security

defence method, even this simple method can work easily. We will construct our covert timing channel with assumption that this naive approach can make sense. It can be blocked easily, but to find LCM is not main work of this paper, we have left that problem to future work.

Algorithm 1 Naive method to find LCM of schedule

R_i: the ith job's release time
S_i: the ith job's start time
L_{dif}: the list which store difference of release time and start time

```
i=0
j=0
while true do
    i = i + 1
    temp = R_i - S_i
    if j == 0 and temp == L_dif[0] then
        j = 1
    else if temp == L_dif[j] then
        if 2 * j == i then
            return i
        end if
        j = j + 1
    else
        j = 0
    end if
    L_dif.append(temp)
end while
```

B. Execution Time Observing

After we deduce LCM of schedule, we have to infer execution time distribution of noisy tasks. Because we assumed that all tasks follow gumbel distribution, we find gumbel distribution's parameter, location and scale with data. And to get data, we should observe noisy task's execution time. However, as we have mentioned before, sender and receiver cannot observe noisy task's execution time equally. Because they cannot know the opposite task's starting time and finishing time. Thus, we should find best way for two tasks to observe similar result. Figure 5 shows observation scope of T_H and T_L. Then it subtract execution time of T_H with observed time, to calculate sum of T_N's execution time, which delay response time of T_L. And at T_H's observation scope, T'_N is not detected. So, we should solve it, and we will suggest sender-receiver least common multiplying method at next subsection. Let's say release time, start time, execution time, finish time, deadline, and delayed time by T_N of task i as R_i, S_i, E_i, F_i, D_i, X_i. X_H and X_L can be obtained easily,

$$X_H = F_H - R_H - E_H$$
$$X_L = S_L - R_H - E_H$$

In addition, observed result will be dissimilar, related to execution time of T_H. Therefore, observation of T_H and T_N should be separated with T_H takes nearly zero time(when sends zero bits) or WCET(when sends one bits).

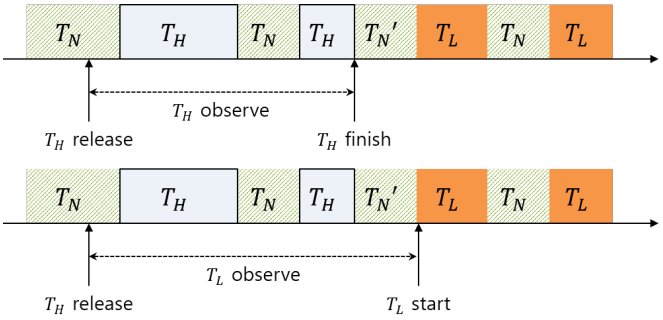


Fig. 5: Observation scope of T_H and T_L

C. Sender-Receiver Least Common Multiplying

Because at T_H's observation scope, T_H cannot detect T'_N, we will ignore T_N', by sending and receiving at only sender and receiver's periods least common multiplying ($SR-LCM$). At $SR-LCM$, T_H and T_L are guaranteed to release their jobs at the same time. So, we can reduce the effect of T_N', which hinders covert timing channel transmission. We identified this method can decrease the effect of T_N' by simulation, and we will discover it at simulation section. Because the effect of T_N' decreased and T_H never detect it, our approach just ignore it. It can be dangerous try when if the gap between T_H and T_L is huge, but we think it it reasonable approach and we also checked it whether it is reasonable or not by simulation. By the way, there's a caveat here. Like figure 6, among LCM, there are several $SR-LCM$s, but because the location of them are different each other, so they should observe T_N separately. We have assumed that tasks can deduce schedule's LCM before, T_H and T_L also can find LCM and check order of $SR-LCM$.

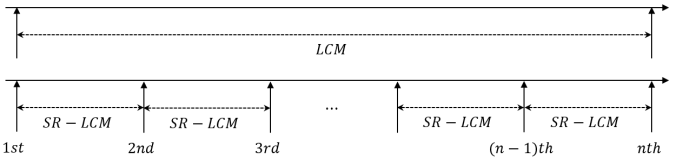


Fig. 6: Observation scope of T_H and T_L

D. Probability Inference of Bits

$$f(x)_H \Leftarrow SetX_H$$
$$f(x)_L \Leftarrow SetX_L$$

With observed execution time data, we can infer distribution of them. The general task's execution time follow gumbel distribution [5]. So we only have to parameter of gumbel distribution, location and scale. And it can be calculated by some process, python or R like most programming languages support it. Then we can get execution time distribution model.

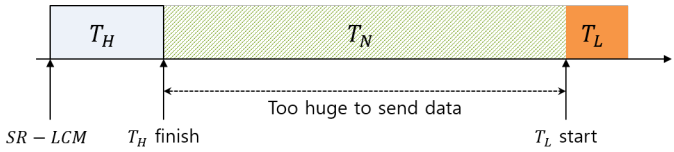
$$f(X_i)^{-1} = probability\ of\ T_N,\ if\ when\ bit\ 0\ sent$$
$$f(X_i - WCET_H)^{-1} = probability\ of\ T_N,\ if\ when\ bit\ 1\ sent$$

The last thing is just sending bits. T_H and T_N can send and receive with the covert timing channel, and when T_L meets uncertain case, then it can deduce what the bit T_H may have sent via execution time distribution model. T_H and T_N decide the transmission's success and transmitted data via probability like below.

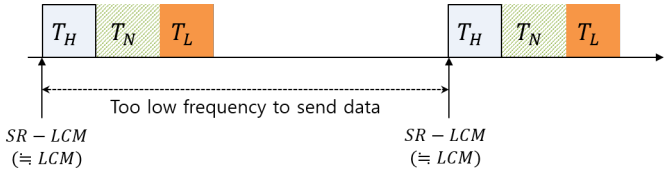
- $if\ f(X_i)^{-1} < f(X_i - WCET_H)^{-1}$
- $then\ take\ bit\ 1$
- $else\ take\ bit\ 0$

E. Problem and Solution

The covert timing channel can transmit bits in most of case, though, there are some special case when constructed covert timing channel cannot transmit bits. Figure 7 shows two case of the special case. When the priority between two task is two huge, they cannot send data successfully. We can assign proper periods, deadline, WCET to prevent those cases. And second case is $SR-LCM$ is too similar with LCM. It cause low chance two send bits, and decrease channel capacity. We can avoid it by selecting periods of T_H and T_L as prime numbers or non preferred period numbers.



(a) Huge Priority Difference



(b) $SR-LCM$ similar with LCM

Fig. 7: Problem Cases of Covert Timing Channel

VI. Simulation

Our method let T_H to ignore T_N' when it estimates T_N and use $SR-LCM$. So we have to assure it proper way. To know it, we used simulation. Our made metric M, which can determine ignored T_N is negligible. $(X_L - X_H)$ means T_N' and we divided its value by $WCET_H$ to calculate ratio of T_H's adjustable range. And we used rate monotonic scheduling algorithm. We used Python and SciPy (open source scientific tools for Python) [19] to utilize gumbel distribution and random number generation.

$$M = (X_L - X_H)/WCET_H$$

We did simulation tasks set which have 6 tasks, and picked sender randomly and receiver always least priority task. All period, deadline, WCET of task selected randomly and all execution time follows gumbel distribution except sender. To simplify simulation, we regared WCET and best execution time (BCET) as point which its probability is less than 1%, and we only used tasks which have WCET less than 15, and best execution time is larger than 1. It only observe 20 LCM times periods, 10 for when T_H takes nearly zero time, and 10 for when T_H takes WCET. We simulated them 100 times, and averaged the value of length of T_N' and sum of delayed time of T_N.

$SR-LCM$	$with\ SR-LCM$		$without\ SR-LCM$	
Sending Bit	Bit 0	Bit 1	Bit 0	Bit 1
$M = (X_L - X_H)/WCET_H$	0.483	0.463	1.160	1.222
Average Delayed time by T_N	3.816	5.603	11.905	13.327

Fig. 8: Summarized Result

The simulation is divided into two part. First one is for checking efficacy of using $SR-LCM$, and Second one is for calculating M. Figure 8 is the summarized result of simulation and Figure 9 and Figure 10 is the graph of showing $SR-LCM$'s performance and comparison between when sending bit 0 and 1. The result represent that $SR-LCM$ decreased M value and T_L's delayed time by T_N over 60%. Also, It is showed that sending bit 0 and 1 does not affect to T_N's behavior which locate between T_H and T_N. Finally the M value, when we apply $SR-LCM$ method, was lower than 1, and it means it can be concealed sufficiently by adjusting execution time of T_H from zero to WCET.

Fig. 9: Effect of $SR-LCM$ Method

VII. Defence Methods

The process of building covert timing channel was pretty complex, but to detect or prevent it is easy. Of course administrator should consider costs of defence, and it is not negligible thing. We prepared three methods to defence or detect it, and because their principles are hands-down thing, we don't analyze them at here.

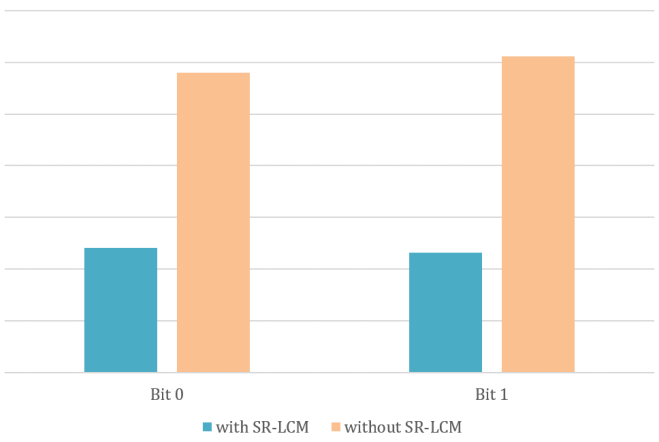


Fig. 10: Comparison Between Bit 0 and 1

A. Hiding Least Common Multiply

Our method is based on LCM inferring. If tasks cannot know schedule LCM, they cannot periodicity and channel accuracy goes down. There are many studies [9], [10], [11], [12] that tried to hide predictability or periodicity of tasks. And its not only real-time system's weak point, we can apply various scheduling security mechanisms.

B. Randomly Ending Schedule

We used execution time distribution and response time checking to build cover timing channel. When task's job end randomly, we cannot use it more. Figure 11 indicate controllable area of a task. System or task can extend its execution time as much as its WCET. And if system randomize its finishing point, it will break gumbel distribution and inferential ability of sender and receiver, thus it can defence covert timing channel attack.

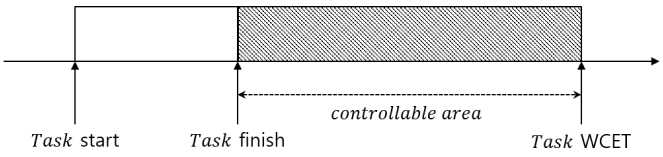


Fig. 11: Controllable area of a task

C. TaskShuffler [10]

TaskShuffler [10] is a timing inference attack defense method, which randomizes task schedule by shufflling tasks. It discovered that the methods highly increase entropy of schedule via Yoon, Mohan, Chen, and Sha's experiment. If schedule has high uncertainty, attacker cannot infer meaningful things. So it can prevent covert timing channel. Figure 12 shows two schedule, first one is original schedule and second one is schedule which is applied TaskShuffler. It can increase amount of context switch, but administrator can balance it by trade uncertainty off with randomization degree.

LCM_1	A	A	A	B	C	A	A	A	C	
LCM_2	A	A	A	B	C	A	A	A	C	
LCM_3	A	A	A	B	C	A	A	A	C	
LCM_4	A	A	A	B	C	A	A	A	C	

(a) Original Schedule

LCM_1	B	A	A	A	C	A	C	A		A
LCM_2	A	C	A	A	B	A	A		A	C
LCM_3	A	B	C	A	A	C	A	A	A	
LCM_4	A	A	B	A	C		A	A	A	C

(b) Randomized Schedule by TaskShuffler [10]

Fig. 12: Changed Scheudle by TaskShuffler [10]

D. Detect by Execution Time Distribution

Above two methods require run-time costs to scheduler, and it can be burden to some schedule. Detecting execution time can be a solution for it. If we observe task's execution time, sender's behavior will stick out because its execution time become nearly zero or WCET when they sends bit data to receiver. And by finding it, we can find some suspicious tasks. It also can applied to real-time detector, but to less burden, we can use it periodically.

VIII. Conclusion

Hacking real-time systems can result critical calamity. So checking available attack method is essential but recent work haven't focused on it. We suggested a novel covert timing channel in MLS real-time systems which is based on Son and Alves-Foss's work. We developed it and added some new methods to increase channel accuracy. It uses execution time distribution and least common multiplying to increase accuracy of cover timing channel. Via calculating execution time distribution of disrupting tasks, sender and receiver can infer vague received result. In this process, we ignored specific period of tasks, T_N', and we checked it is negligible via simulation. Also, our approach that transmission at only sender and receiver periods' least common multiply time, can reduce interruption of other tasks and our simulation showed it reduced as much as 60%. Also, we suggested four methods that can prevent or detect our method.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by and the Ministry of Education and the Ministry of Science, ICT Future Planning (NRF-2017R1A2B2002458, NRF-2016R1D1A1B03930580). This research was also supported by the SW focused university supporting program by the Institute for Information and Communication Technology Promotion and the Ministry of Science, ICT Future Planning (2015-0-00914).

REFERENCES

[1] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[2] J. W. Liu, "Real-time systems. 2000."

[3] J. Hansen, S. A. Hissam, and G. A. Moreno, "Statistical-based wcet estimation and validation," in *Proceedings of the 9th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis*, 2009.

[4] G. Bernat, A. Colin, and S. M. Petters, "Wcet analysis of probabilistic hard real-time systems," in *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*. IEEE, 2002, pp. 279–288.

[5] S. Edgar and A. Burns, "Statistical analysis of wcet for scheduling," in *Real-Time Systems Symposium, 2001.(RTSS 2001). Proceedings. 22nd IEEE*. IEEE, 2001, pp. 215–224.

[6] P. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in CryptologyCRYPTO96*. Springer, 1996, pp. 104–113.

[7] J. Son *et al.*, "Covert timing channel analysis of rate monotonic real-time scheduling algorithm in mls systems," in *Information Assurance Workshop, 2006 IEEE*. IEEE, 2006, pp. 361–368.

[8] C.-Y. Chen, A. Ghassami, S. Nagy, M.-K. Yoon, S. Mohan, N. Kiyavash, R. B. Bobba, and R. Pellizzoni, "Schedule-based side-channel attack in fixed-priority real-time systems," Tech. Rep., 2015.

[9] M. Völp, C.-J. Hamann, and H. Härtig, "Avoiding timing channels in fixed-priority schedulers," in *Proceedings of the 2008 ACM symposium on Information, computer and communications security*. ACM, 2008, pp. 44–55.

[10] M.-K. Yoon, S. Mohan, C.-Y. Chen, and L. Sha, "Taskshuffler: A schedule randomization protocol for obfuscation against timing inference attacks in real-time systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2016 IEEE*. IEEE, 2016, pp. 1–12.

[11] R. Pellizzoni, N. Paryab, M.-K. Yoon, S. Bak, S. Mohan, and R. B. Bobba, "A generalized model for preventing information leakage in hard real-time systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015 IEEE*. IEEE, 2015, pp. 271–282.

[12] S. Mohan, M.-K. Yoon, R. Pellizzoni, and R. B. Bobba, "Integrating security constraints into fixed priority real-time schedulers," *Real-Time Systems*, vol. 52, no. 5, pp. 644–674, 2016.

[13] D. McCullough, "Specifications for multi-level security and a hook-up," in *Security and Privacy, 1987 IEEE Symposium on*. IEEE, 1987, pp. 161–161.

[14] J. C. Wray, "An analysis of covert timing channels," *Journal of Computer Security*, vol. 1, no. 3-4, pp. 219–232, 1992.

[15] M. Stamp, *Information security: principles and practice*. John Wiley & Sons, 2011.

[16] E. J. Gumbel, *Statistics of extremes*. Courier Corporation, 2012.

[17] C. Zimmer, B. Bhat, F. Mueller, and S. Mohan, "Time-based intrusion detection in cyber-physical systems," in *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*. ACM, 2010, pp. 109–118.

[18] M.-K. Yoon, S. Mohan, J. Choi, M. Christodorescu, and L. Sha, "Learning execution contexts from system call distributions for intrusion detection in embedded systems," *arXiv preprint arXiv:1501.05963*, 2015.

[19] E. Jones, T. Oliphant, and P. Peterson, "{SciPy}: open source scientific tools for {Python}," 2014.