Poster Session Proceedings

7th IEEE European Symposium on Security and Privacy

Following the EuroS&P 2022 conference, the abstracts of the posters are made available to the community in these proceedings. In order to provide a citable source document, the proceedings are published at Zenodo (https://zenodo.org), an open research repository developed under the European OpenAIRE program and operated by CERN, which enables sharing and preserving of research outputs.

We would like to thank all the authors for the creativity and effort that went into their submissions, and especially the poster presenters (including the invited posters' presenters), for their tireless engagement with the attendees during the lively poster session. We are also grateful to Gabriele Costa and Alessio Merlo, the general chairs; Daniele Cono D'Elia and Leonardo Querzoni, the publication chairs; Carmela Troncoso and David Evans, the program chairs, and to everyone else who helped facilitate the poster session. We extend a special thanks to Gabriele Costa for supporting the post-conference process of publishing the posters.

Vera Rimmer, KU Leuven, Belgium
Guillermo Suarez-Tangil, IMDEA Networks Institute, UK
IEEE European Symposium on Security and Privacy 2022 Poster Co-Chairs

Posters

1. A boostershot for transferable physically realizable adversarial examples
   Willem Verheyen, Sander Joos, Tim Van hamme, Davy Preuveneers, Wouter Joosen
2. A practical methodology for ML-Based EM Side Channel Disassemblers
   Cesar Arguello, Hunter Searle, Sara Rampazzi, Kevin Butler
3. Detecting Network Anomalies from Small Traffic Samples using Graph Neural Network
   Aviv Yehezkel, Eyal Elyashiv
4. Exploiting Timing Side-Channel Leaks in Web Applications that Tell on Themselves
   Vik Vanderlinden, Tom Van Goethem, Wouter Joosen, Mathy Vanhoef
5. How Attackers Determine the Ransom in Ransomware Attacks
   Tom Meurs, Marianne Junger, Abhishta Abhishta
6. One of a Kind: Correlating Robustness to Adversarial Examples and Face Uniqueness
   Giuseppe Garofalo, Tim Van hamme, Davy Preuveneers, Wouter Joosen
7. Pillars of Sand: The current state of Datasets in the field of Network Intrusion Detection
   Gints Engelen, Robert Flood, Lisa Liu-Thorrold, Vera Rimmer, Henry Clausen, David Aspinall, Wouter Joosen
8. Systematic Elicitation of Common Security Design Flaws
   Stef Verreydt, Laurens Sion, Koen Yskout, Wouter Joosen
9. TESTABLE: Testability-driven security and privacy testing for Web Applications
   Luca Compagna, Giancarlo Pellegrino, Davide Balzarotti, Martin Johns, Ángel Cuevas, Battista Biggio, Leyla Bilge, Fabian Yamaguchi, Matteo Meucci
10. The Beauty and the Beast (40 years of process algebra and cybersecurity)
    Silvia De Francisci, Gabriele Costa, Rocco De Nicola
11. The impact of data sampling in the anonymization pipeline
    Jenno Verdonck, Kevin De Boeck, Michiel Willocx, Jorn Lapon, Vincent Naessens
12. The impact of public data during de-anonymization: a case study
    Kevin De Boeck, Jenno Verdonck, Michiel Willocx, Jorn Lapon, Vincent Naessens
13. Towards Cyber Resilience of Cyber-Physical Systems using Tiny Twins
    Fereidoun Moradi, Sara Abbaspour Asadollah, Marjan Sirjani

Poster: A boostershot for transferable physically realizable adversarial examples

Verheyen Willem, Sander Joos, Tim Van hamme, Davy Preuveneers, Wouter Joosen
imec-DistriNet, KU Leuven

***Abstract*—Adversarial perturbations are claimed to enlarge the attack surface of machine learning models. However, as the most prominent attack methodologies require unrealistically strong adversaries, they are hardly used in attacks against real-world systems. In this paper, we alleviate the constraints on the threat model and attack a face recognition system with physically realizable perturbations in a black-box scenario, provided a single attack attempt. As such, we are forced to rely on more pragmatic, but less effective, attack methods that leverage transferability – adversarial perturbations successful on known models tend to also work on unknown ones. We overcome the poor attack success rate of transferability by using adversarially trained surrogate models.**

I. Introduction

A well-known problem of neural networks is their susceptibility to adversarial examples, e.g., images perturbed in such a way that changes are imperceptible to humans but impair the standard operation of neural networks. Despite a large body of work on methodologies to generate adversarial examples, the number of attacks on real-world models that take advantage of them is limited. This low prevalence in real-world attacks can be explained by the restrictiveness of the threat model that is present in practical ML-based systems. Attacking such systems requires a strong adversary with capabilities that are often unrealistic in practice. Moreover, the adversary's aim is to evade a face recognition system without knowledge about the network architecture but does possess information about a limited number of identities that can be recognized. Furthermore, the adversary's goal is to evade detection on the first try and is therefore limited to a single query without digital access to the target model.

Therefore, the adversary needs to rely on robust surrogate models to find a physical adversarial perturbation in the shape of glasses [1]. It is shown by previous studies [2]–[4] that robust models learn more universal representations of the training data as opposed to non-robust models. As such, the robust model learns better generalizing features, which serves as evidence that adversarial perturbations generated on robust surrogate models target features which are also present in other networks that fulfill a similar purpose.

II. Related work

In this section, we introduce related work that motivates the use of adversarially trained surrogate models to increase transfer-based adversarial examples.

Transferability allows an adversary to generate adversarial examples on a known model and use these to attack unknown models. Although transfer-based attacks have a relatively low success rate compared to other attack methods [5]–[8], their major benefit is the limited query amount to the target model.

One proposed method to increase the low success rate of transfer-based adversarial examples is to generate them on surrogate models that have been adversarially trained on attacks of similar nature [9]. The reason behind this is that models robust against adversarial examples learn more generalizing features that are shared with other DNNs. Therefore, adversarial examples that exploit these features transfer better than those generated on non-robust DNNs.

Increasing the success rate of transfer-based adversarial examples has been explored further in the scope of digital adversarial examples [3], [4], [10]–[12]. Yet, similar to the transferability of adversarial examples, current understanding of this topic remains incomplete, especially for physical adversarial examples.

III. Attack methodology and experimental setup

In this section, we first describe the attack methodology followed by the experimental setup; among others, we specify how the adversarial perturbations are generated, and describe the data and models used.

A. Attack methodology

1) Obtain a collection of samples to train the surrogate model. This contains samples of the attacker that ultimately need to be misclassified, but also samples from other identities assumed to be recognized by the model under attack.
2) Perform adversarial training on the surrogate model with the collected set of samples by finding optimal perturbations.
3) Use the newly robust surrogate model to craft adversarial examples and only utilize those that are successful on the surrogate while their benign counterpart is also classified correctly.
4) Use the attacks that successfully fooled the surrogate model to attack the target model.

B. Experimental setup

To perform our evaluation in light of face recognition we require: 1) a sufficiently large face dataset which can be split into two training and one attack portion, 2) different model architectures, 3) a methodology to generate physical

realizable attacks, and 4) models with a varying degree of robustness to obtain the adversarial test sets.

In the following we describe how we obtain each of the aforementioned requirements.

Req. 1: In contrast to prior work that investigates transferability between DNNs that classify identical classes [9], [13], we make a distinction between the identities recognized by the face recognition surrogate- and target model with some degree of overlap. We therefore use a portion of the VGGFACE2-dataset [14] and split this into three distinct subsets, each consisting of 400 identities that represent one core dataset shared by both surrogate- and target model and one additional set of identities for both the surrogate and target model respectively.

Req. 2: We consider the following architectures for both the surrogate- and target models, for which we use pre-trained weights: VGGFace pretrained on the VGGFace dataset [14], Facenet pretrained on the VGGFACE2 dataset [15] and VGG19 pretrained on the VGGFACE2 dataset. We fine-tune these models with our data, such that they classify the adequate set of identities.

Req. 3: Our physically realizable adversarial examples are based on the work by Sharif et al. [1]. we consider physically realizable adversarial examples with localized perturbations in the shape of glass frames. The adversarial glasses are always cropped to match the size of a person's face and rotated accordingly.

Req. 4: We first construct a number of increasingly robust classifiers and then use these to construct test sets containing adversarial examples. Robust classifiers $\hat{f}_i$ are obtained with adversarial training on the initial classifier f, for a number of epochs i with adversarial examples generated from our dataset D. For each surrogate model, we then generate a test set $\hat{D}_i$ containing adversarial examples that are misclassified, and thus successful as an attack. Adversarial attacks are only added to the adversarial test set if their benign counterpart is still classified correctly by the surrogate model. This allows us to partially mitigate the effect of the well-known problem where adversarial training causes a drop in standard accuracy [16], [17]. Subsequently, we use these test sets to evaluate the transferability of our robustly generated adversarial examples to other classifiers. We only consider adversarial examples on the target model when their benign counterpart is also classified correctly. This is an important assumption, as it is very likely whenever a benign sample is misclassified, its resulting adversarial examples are also likely to be misclassified.

IV. Evaluation

We use the adversarial test sets generated on the surrogate models to attack the different target models and measure the transfer rate across them.

First, we demonstrate that adversarial examples generated on robust surrogate models have a higher probability of success when used to attack target models. In order to do so, we compare the transfer rates of attacks in adversarial test sets $\hat{D}_0$ and $\hat{D}_{max}$, where $\hat{D}_0$ contains adversarial examples generated on the non-adversarially trained surrogate and $\hat{D}_{max}$ contains adversarial examples generated on the most robust surrogate. Fig. 1 shows the distribution of transfer rates for both $\hat{D}_0$ (non-robust) and $\hat{D}_{max}$ (robust) when they are used to attack different target models. The target models considered are both not adversarially trained, and adversarially trained using the adversarial examples of similar nature. The transfer rate of adversarial examples generated on robust surrogate models increases from an average of 27% to 40% and from 4% to 15% for non-robust and robust target models respectively.

This shows that even when the target model is robust against the considered adversarial examples, the attack success rate can be increased when using robust surrogate models.

Next, Fig. 2 shows that as model robustness increases as a result of adversarial training, so does the transferability of physical realizable adversarial examples. This is in line with findings in previous work that claims that the classifier becomes more robust to adversarial examples, they rely more on robust features instead of non-robust features. As a result, features learned by robust classifiers benefit a higher degree of universality, whereas non-robust features are less universal and thus transfer worse [12].

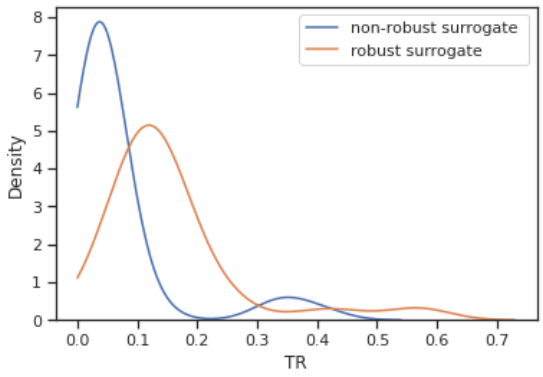


Fig. 1: KDE-plot of the transfer ratio of physical realizable adversarial examples generated on robust and non-robust surrogate models and transferred to non-robust and robust target models.

V. Discussion

This section discusses the main implications of our work, shortcomings, and possible directions for future work.

a) ***Security implications****:* In this work, we aimed to increase the transferability of physical adversarial perturbations to better accommodate threat models that require physically realizable perturbations against black-box models with a near zero-query budget. Our results extend the findings of recent works [3], [9]–[11] that already provided evidence that the transfer rate of adversarial examples with unlocalized perturbations increases when generated on robust classifiers. In this work, we leverage these findings and propose model robustness as a prior for the generation of physical realizable adversarial examples. Specifically, we demonstrate an impressive increase in attack success rate between 1.5x and 7x against

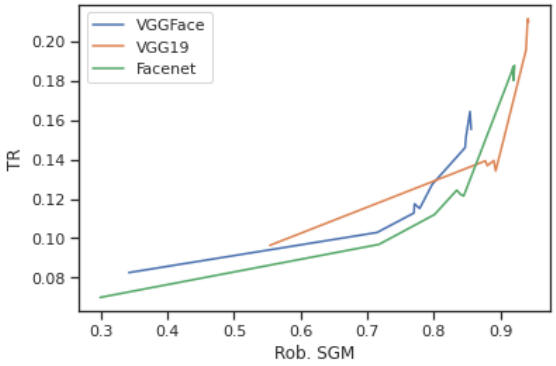


Fig. 2: Average transfer rates for increasingly robust surrogate models. The X-axis represents the accuracy on adversarial examples of the surrogate model, the y-axis represents the transfer rate to different black box target models.

face recognition systems when using robust surrogate models over their non-robust counterparts. On the one hand, with an absolute success rate which varies between 6 and 56% we are not consistently evading detection by face recognition systems. On the other hand, in an impersonation scenario against an authentication system, such an increase is significant, as often multiple authentication attempts are allowed [18], e.g., FaceID allows five authentication attempts before switching to PIN input.

*b) **Validity threats**:* The experimental setup inherits several less realistic assumptions on the proposed threat model which we attempt to solve in our current ongoing work.

We assume both target- and surrogate models use partially overlapping datasets in their training procedure. This is an exaggerated simplification of a real-world scenario where an attacker has limited to no knowledge of the data used to train the target model. However, we assume that further minimizing this overlap will have a limited impact on our results because the intuition behind deep neural networks is that they can distribute feature space evenly across different classes, separating each class with similar distances to each other. Therefore, we propose the use of feature extractors to overcome relying on overlapping datasets which, in turn, also relates better to face recognition/authentication as an open-world problem. Furthermore, we plan on considering impersonation attacks as well to bridge the transition to authentication.

VI. Conclusion

In this paper, we propose a method to increase the attack success rate of adversarial examples to face recognition systems in a highly restrictive, yet realistic black-box setting. We do so by leveraging and enhancing the transferability property of adversarial examples that are realizable in the physical world by generating attacks on adversarially trained surrogate models. Specifically, we found that using a robust surrogate model over its non-robust counterpart drastically increases transferability with a factor of 1.5 up to 7 for single attempt attacks compared to the state-of-the-art. Moreover, even in the case of low absolute attack success rates such increases are significant for attacks against applications that allow for more than one attempt but implement rate limiting, e.g., face authentication systems. In conclusion, we believe that this work provides a compelling contribution to the creation of adversarial examples that impose a significant threat to practical machine learning applications.

References

[1] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[2] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," 2019.

[3] M. Terzi, A. Achille, M. Maggipinto, and G. A. Susto, "Adversarial Training Reduces Information and Improves Transferability," *arXiv:2007.11259 [cs, stat]*, Dec. 2020. arXiv: 2007.11259.

[4] J. M. Springer, M. Mitchell, and G. T. Kenyon, "A little robustness goes a long way: Leveraging robust features for targeted transfer attacks," 2021.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv:1412.6572 [cs, stat]*, Mar. 2015. arXiv: 1412.6572.

[6] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into Transferable Adversarial Examples and Black-box Attacks," *arXiv:1611.02770 [cs]*, Feb. 2017. arXiv: 1611.02770.

[7] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," 2017.

[8] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples," *arXiv:1605.07277 [cs]*, May 2016. arXiv: 1605.07277.

[9] J. M. Springer, M. Mitchell, and G. T. Kenyon, "Adversarial perturbations are not so weird: Entanglement of robust and non-robust features in neural network classifiers," 2021.

[10] H. Salman, A. Ilyas, L. Engstrom, A. Kapoor, and A. Madry, "Do Adversarially Robust ImageNet Models Transfer Better?," *arXiv:2007.08489 [cs, stat]*, Dec. 2020. arXiv: 2007.08489.

[11] S. Shan, E. Wenger, J. Zhang, H. Li, H. Zheng, and B. Y. Zhao, "Fawkes: Protecting privacy against unauthorized deep learning models," 2020.

[12] J. M. Springer, M. Mitchell, and G. T. Kenyon, "Adversarial Perturbations Are Not So Weird: Entanglement of Robust and Non-Robust Features in Neural Network Classifiers," *arXiv:2102.05110 [cs]*, Feb. 2021. arXiv: 2102.05110.

[13] F. Utrera, E. Kravitz, N. B. Erichson, R. Khanna, and M. W. Mahoney, "Adversarially-Trained Deep Nets Transfer Better: Illustration on Image Classification," *arXiv:2007.05869 [cs, stat]*, Apr. 2021. arXiv: 2007.05869.

[14] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference*, 2015.

[15] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," 2018.

[16] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," 2019.

[17] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," 2019.

[18] P. Markert, D. V. Bailey, M. Golla, M. Dürmuth, and A. J. Aviv, "This pin can be easily guessed: Analyzing the security of smartphone unlock pins," 2021.

Poster: A practical methodology for ML-Based EM Side Channel Disassemblers

Cesar Arguello
CISE
University of Florida
carguello1@ufl.edu

Hunter Searle
CISE
University of Florida
huntersearle@ufl.edu

Sara Rampazzi
CISE
University of Florida
srampazzi@ufl.edu

Kevin Butler
CISE
University of Florida
butler@ufl.edu

***Abstract*—Providing security guarantees for embedded devices with limited interface capabilities is an increasingly crucial task. Although these devices don't have traditional interfaces, they still generate unintentional electromagnetic signals that correlate with the instructions being executed. By collecting these traces using our methodology and leveraging a random forest algorithm to develop a machine learning model, we built an EM side channel based instruction level disassembler. The disassembler was tested on an Arduino UNO board, yielding an accuracy of 88.69% instruction recognition for traces from twelve instructions captured at a single location in the device; this is an improvement compared to the 75.6% (for twenty instructions) reported in previous similar work.**

***Index Terms*—electromagnetism, side-channel, disassembly, security**

1. Introduction

Embedded devices form an integral part of the modern computing ecosystem. They can be found in a myriad of applications, ranging from household appliances to security-critical industrial controllers. Securing this wide range of devices is a massive and crucial design challenge, especially with the rise in connectivity of the Internet of Things and emerging threats [1]. Many of these devices, such as fuel tank monitors or farming field sensors, offer little insight into their internal workings due to proprietary technology or limited interfaces. Additionally, devices such as medical devices might be deployed long-term, with little or no ability to update their software against cyberthreats. This combination of longevity and minimal access create a situation where devices are susceptible to many forms of attacks, including disrupting functionality, falsifying sensor output, and increasing power consumption to drain batteries [2].

One proposed approach to the challenge of protecting embedded devices against these attacks is to use side-channel information. Side-channels refer to information that is leaked by unintentional signals generated in the normal operation of a processor. There are many forms of side-channel, including power, noise, electromagnetism, timing, etc [3] [4] [5]. In each case, the signal is correlated to the operations that generated them, so that information about those operations can be retrieved from the signal. Of these, power side-channels offer the clearest signal, and have consequently received more attention from researchers. However, they require a direct connection to the processor's power rail. Electromagnetic signals, however, can be captured with no physical connection, making them more preferable as a side-channel security mechanism.

In this work we propose an efficient approach for building an electromagnetic side-channel based disassembler, which can identify specific instructions being executed on an embedded processor. We then use our approach to build a proof-of-concept EM-based disassembler to be used for implementing anomaly detection and control flow mechanisms on low-cost embedded systems without the need for firmware or hardware redesign. In comparison with previous work [6], [7] that uses wiring or multiple measurement points, our approach leverages the presence in the device's board of electronic components that generate EM emanations (e.g., amplifiers) correlated to the internal processor operations. Using a simple random forest algorithm for classification, we achieve single instruction granularity with 88.69% accuracy over twelve different instructions on an ATMega328P 16MHz processor. This is an improvement over previous work [6] which required multiple measurement points and was only tested on a 4MHz processor to achieve 75.6% accuracy on 20 instructions. These encouraging results open the possibility of utilizing our approach to build more efficient and high-accuracy disassemblers to be used for anomaly detection.

2. Background

Side-channel signals have been a known source of information leakage for decades [8] [9] [10]. They have been exploited for recovering different types of data, including screen images [10], secret keys [3], and audio [11]. Only recently have side-channel analysis techniques been turned to program disassembly, starting when Eisenbarth et al. built a disassembler using power analysis [12]. Since then, much progress has been made in this research area. SCANDALee was the first successful em-based disassembler [6]. Their method required multiple measurement locations, and could only recover a portion of the instruction set on a slower (4MHz) processor. New techniques were introduced for disassembly by Park et al., who were able to recover nearly the entire instruction set, along with registers, of a faster processor (ATMega 238P) [7], although they only consider power side-channels, which require a wired connection. Neural networks have also been leveraged for disassembly, achieving similar levels of accuracy only on power side channel [13]. Other works have sought to ensure security without instruction-level

disassembly, such as Khan et al.'s EM-based intrusion detection system [14]. Our methodology builds on these previous works to overcome the limitations of multiple-point measurement and signal extraction to achieve instruction-level disassembly using EM emissions.

3. Methodology Overview

Our process is composed of three steps: 1) Leakage detection, 2) Signal extraction, and 3) Classification. Leakage detection is only done once, whereas signal extraction and classification are performed every time.

Leakage Identification. Following the template-based method used by previous work [7], we sample individual instructions to build a template model before classifying them in real code. Our approach begins by identifying the component from which EM emanations are most strongly correlated to executing instructions. This is done empirically with a grid search across the device PCB using a EMI probe located a few centimeters from the board. At each point in the grid, we perform a simple classification using a Random Forest algorithm that has been adapted for use with time series implemented by sktime [15]. To use this method we generate a template built from samples of individual instructions with a no-operation (NOP) immediately preceding and following it to better identify the signal correlation with the executed instructions. Before and after the target instruction, we also implement a trigger operation for inducing a voltage change (e.g., an instruction to flip a GPIO bit). This trigger is used to separate the individual instructions for classification. This procedure gives a lower single-instructions accuracy than our final classification, however, it shows the relative information leakage that is used to identify one or more optimal measurement locations corresponding to the "leaky" electronic components (see Figure 2). After this procedure the optimal measuring point is selected and used for the rest of the process. Finally, because our process uses the Fast Fourier Transform of the signal, rather than the time series data, we identify the target frequency bands using a spectrum analyzer.

Signal Extraction and Classification. The process to build our template and to classify unknown signals follow the same steps of signal extraction and classification. This step consists in measuring the magnetic field around the optimal point identified in the first step. Then we automatically separate the instructions using the triggers, and compute the FFT for each instruction. The magnitudes of the transform at the frequencies identified in step one are then used as features to train our model. While our methodology can be adapted to any machine learning algorithm, in our case study we found that the Random Forest algorithm (not to be confused with the Time Series Random Forest algorithm that we used for the grid search) gave the highest single-instruction accuracy.

4. Proof-of-concept Disassembler

To test our methodology, we built a proof-of-concept disassembler. We selected a subset of the AVR instruction set by examining real-world code for a stack overflow attack, then tested our disassembler on an ATMega328P.

Experimental setup. Our acquisition setup is shown in Figure 1. We used a Tektronix H10 H-Field Probe connected to a Tektronix MDO4024C oscilloscope to collect traces from an Arduino UNO.

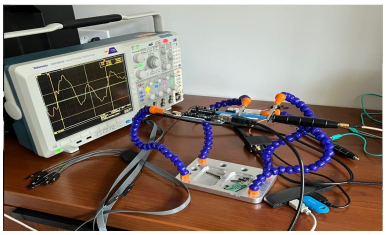


Figure 1. Acquisition setup made of a Tektronix H10 H-Field Probe connected to a Tektronix MDO4024C oscilloscope.

Leakage Identification Phase. We began with a 80-point grid search over the Arduino board and ran the Time Series Random Forest classifier. We found that the highest accuracy was near an operational amplifier connected to the crystal oscillator, shown in Figure 2. This was the point at which all other measurements were taken.

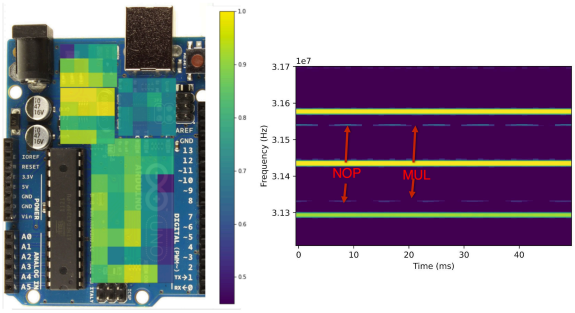


Figure 2. (Left) Recognition accuracy for the 80 subsection in which the device under test was divided. The area near to the operational amplifier connected to the crystal oscillator shows the highest accuracy (yellow). (Right) Example of spectral profiling on a group of MULs and NOPs.

We used the Signal Hound SA44B spectrum analyzer to study the frequency spectrum. Because the leakage component was closer to the Arduino 16 MHz clock, we examined frequencies within 1 MHz of the clock sub-harmonics. Our analysis found that frequencies between 31.3-31.6 MHz showed the largest voltage difference for our test instructions. The bands are shown in Figure 2. Finally, we performed the hyperparameter optimization for the tested classification algorithms as shown in Table 1. The random forest algorithm presented the highest accuracy and was hence chosen as the optimal one for the development of the disassembler.

TABLE 1. RANDOM SEARCH HYPERPARAMETER OPTIMIZATION

Algorithm	***Accuracy***	***Optimum Hyperparameters***
Random Forest	85%	Num. Estimators = 1000 Min. Interval = 2
Random Interval Spectra	56%	Num. Estimators = 829 ACF lang = 400
K Nearest Neighbors	76%	Num. Neighbors = 100
Support Vector Machines	82%	Kernel = linear Gamma = 0.1

4.1. Evaluation

4.1.1. Single Instructions Recognition. The results of accuracy recognition for twelve selected instructions after performing four fold cross validation in the developed disassembler are summarized in Figure 3. The proposed implementation yielded an 88.69% recognition accuracy.

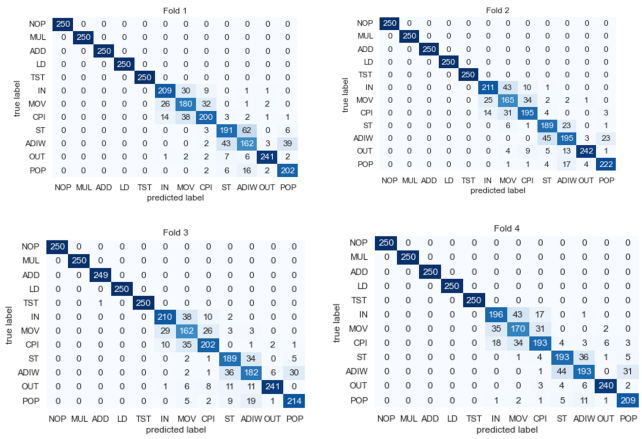


Figure 3. Confusion matrix of our EM side-channel based disassembler four cross validated

4.1.2. Code Recognition. To test the capability of our approach to recognize instructions on a potential real-world case study, we consider the scenario of a real-world medical device, SyringePump [16]. The pump is designed to deliver medications to a patient at periodic intervals. The system typically consists of a syringe, an actuator (a stepper motor), and a control unit (Arduino UNO) that takes commands from the serial port. We focus on

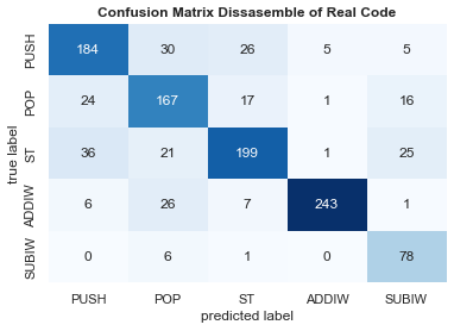


Figure 4. Confusion matrix for identification of instructions in three cycle instructions of the partially implemented Arduino SerialRead function

a specific section of the Arduino SerialRead function as test case. This function is crucial for anomaly detectors to infer the presence of ongoing buffer overflow attacks on the internal buffer of the serial port.

A small driving program was made such that a computer sends three arbitrary characters to the Arduino board serial port and the SerialRead function stores those characters in a buffer. Our acquisition setup was used to collect traces for the relevant five instructions used by the function. The driving program was run 500 times and 75% of these traces were used to improve the single-instruction model while the rest was used for testing. The recognition accuracy obtained was 77.4% as shown in Figure 4.

5. Observations and Future Work

The proposed methodology allows single-instruction classification with high accuracy. Moreover, by observing the number of certain operations execution as well as the timing deviation from the regular behavior, an anomaly detector can leverage this findings to automatically determined if an attack was causing such deviation. Although, the results of this test case scenario are preliminary and more testing is required to evaluate the robustness of our approach on different scenarios, they also show the potential of our methodology to be used not only identify anomalies but also provide forensic evidence for their categorization. Planned future work will address a more in-depth evaluation on multiple case scenarios and different processors to fully validate our methodology.

Acknowledgment

This work was funded in part by The Center for Enabling Cyber Defense in Analog and Mixed Signal Domain (CYAN - AFRL FA8650-19-1-1741) and a gift from Facebook.

References

[1] I. Ahmad, R. Ziar, and M. Niazy, "Survey on iot: Security threats and applications," vol. 2, pp. 42–46, 2020.

[2] M. Baezner and P. Robin, "Stuxnet," 2018.

[3] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," 1999.

[4] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic Analysis: Concrete Results," 2001.

[5] C. Rechberger and E. Oswald, "Practical Template Attacks," 2004.

[6] D. Strobel, F. Bache, D. Oswald, F. Schellenberg, and C. Paar, "Scandalee: a side-channel-based disassembler using local electromagnetic emanations," in *2015 Design, Automation & Test in Europe Conference & Exhibition*, 2015.

[7] J. Park, X. Xu, Y. Jin, D. Forte, and M. Tehranipoor, "Power-based side-channel instruction-level disassembler," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018.

[8] R. J. Anderson, "Emission Security," in *Security Engineering*, 2001.

[9] "AFSSM 7011 - Emission Security Countermeasures Review." [Online]. Available: https://cryptome.org/afssm-7011.htm

[10] W. van Eck, "Electromagnetic radiation from video display units: An eavesdropping risk?" *Computers & Security*, 1985.

[11] J. Choi, H.-Y. Yang, and D.-H. Cho, "TEMPEST Comeback: A Realistic Audio Eavesdropping Threat on Mixed-signal SoCs," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[12] T. Eisenbarth, C. Paar, and B. Weghenkel, "Building a Side Channel Based Disassembler," *Transactions on Computational Science X*, 2010.

[13] P. Narimani, M. A. Akhaee, and S. A. Habibi, "Side-channel based disassembler for avr micro-controllers using convolutional neural networks," in *International ISC Conference on Information Security and Cryptology (ISCISC)*, 2021.

[14] H. A. Khan, N. Sehatbakhsh, L. N. Nguyen, R. L. Callan, A. Yeredor, M. Prvulovic, and A. Zajić, "IDEA: Intrusion Detection through Electromagnetic-Signal Analysis for Critical Embedded and Cyber-Physical Systems," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[15] "Timeseriesforestclassifier." [Online]. Available: https://www.sktime.org/en/latest/api_reference/auto_generated/ sktime.classification.interval_based.TimeSeriesForestClassifier.html

[16] "Open syringe-pump source code and project." [Online]. Available: https://github.com/naroom/OpenSyringePump

Poster: Detecting Network Anomalies from Small Traffic Samples using Graph Neural Network

Aviv Yehezkel
Cynamics
Israel
aviv@cynamics.ai

Eyal Elyashiv
Cynamics
Boston, US
eyal@cynamics.ai

***Abstract*—Detecting anomalies in computer networks is a classic, long-term research problem. While almost every kind of model architecture has been proposed, previous works usually analyzed the entire network traffic. However, such analysis implies high memory and processing overhead, and is becoming less applicable for large networks. In this poster we present a work in progress which studies a previously under-researched setting where only a small fraction of the network traffic is given. Our approach pre-processes the samples and transforms the computer network into a graph neural network (GNN). It distills node features, edge features and graph representations to learn a vector embedding for each endpoint which characterizes its normal behaviors. Then, a link predictor model is used to estimate the likelihood of network communications and detect anomalies.**

***Index Terms*—Deep-Learning; Graph Neural Network; Network Anomaly Detection; Network Security**

1. Introduction

Detecting anomalies in computer networks is a classic, long-term research problem. Almost every kind of model architecture has been studied: statistical, clustering, classification, information-theory, deep-learning, and others (see [1] for a recent comprehensive survey on network anomaly detection). Specifically, various learning-based approaches were used [2]–[6]. However, previous works were based on analyzing the entire network traffic, which is less applicable for large networks.

A much less-studied approach is the sampling approach, which is especially suitable for high-speed (gigabit or more) or high throughput networks, where only a small fraction of the packets (for example 1%) is being sampled and summarized [7].

In a recent work, we showed how a small percent of uniform sampling can be used to efficiently and accurately detect network anomalies and attacks using the concept of "auto-encoder losses transfer learning" [8]. Our approach collected 1% network samples for each client, trained an auto-encoder neural network for each client's network and then normalized all auto-encoder losses of the different clients, providing the ability to transform loss vectors of different client networks with potentially significant varying characteristics, properties, and behaviors into a similar statistical distribution. The normalized losses can then be forwarded to a global detection model that detects and classifies threats in a generalized way that is agnostic to the specific client. We used extensive simulation study to compare the "sampled" approach to the existing "unsampled" state-of-the-art approach and showed its superior detection accuracy.

In this poster, we present a work in progress which intends to take our previous research one step deeper in the network - from the gateways to the endpoints. The proposed approach will transform the computer network into a graph neural network (GNN). It will distill node features, edge features and graph representations to learn a vector embedding for each endpoint which characterizes its normal behaviors. Then, a link predictor model will be used to estimate the likelihood of network communications and detect anomalies in the endpoint level. As this work is still in progress, the poster abstract will mainly present our research idea and approach, without providing evaluations and preliminary results.

In addition to the lower processing cost, a sampling-based network anomaly detection approach has many advantages, such as: (a) Data privacy: the packet's payload is not analyzed at any moment; (b) built-in robustness against sophisticated attacker utilizing ML techniques, due to the randomized nature.

The key contributions of the work being presented in the poster are:

- A novel approach for transforming a computer network to graph neural network.
- Learning a vector, characterizing the normal behaviours of each endpoint, using multiple aspects of the sampled traffic data (of the node, edge and graph) and detecting anomalies in the endpoint level in a manner that is agnostic to the traffic volume and network size.
- A new method for network anomaly detection using a small fraction of network samples based on a combination of graph neural network and link predictor model.

2. The Approach

The approach consists of three main stages (see in Figure 1):

1) Computer networks are transformed to graph neural networks based on key aspects of the network traffic, endpoints (nodes) and communications (edges).

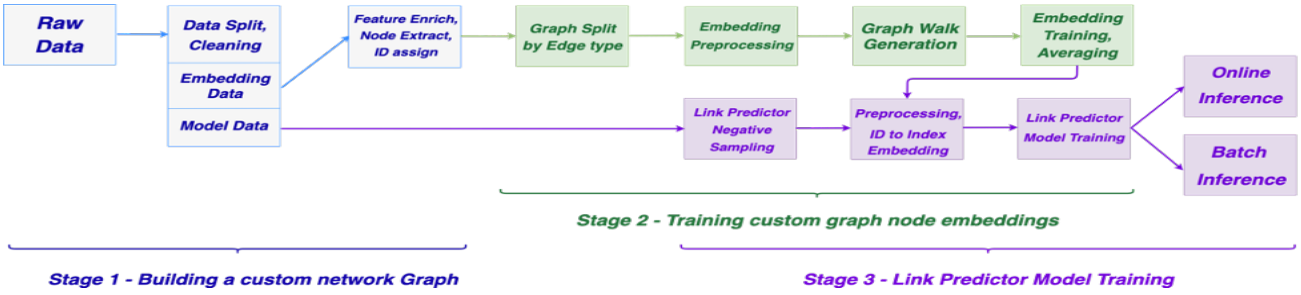


Figure 1. An high-level overview of the proposed approach.

2) A vector embedding is learned for each graph node, characterizing its normal behaviors by fusing together different node perspectives (node features, edge features and graph representations).
3) A link-predictor model is trained to estimate a likelihood of network communications and detect anomalous links.

First, sampled IP flows[1] data is collected from main network gateways (e.g., firewalls and switches). Each record represents a meta-data summarization of communication between two endpoints (IP addresses) in the network with their flow details: source IP address, destination IP address, source port, destination port, IP protocol, creation time, number of packets in flow, flow length in bytes.

Then, the network traffic is transformed to a graph neural network with two types of nodes:

- IP entities – the flow's source and destination IPs.
- "PPP" triplets entities – an "artificial" node representing the communication between IP entities, composed of a combination between the flow's source port, destination port and IP protocol, i.e., (source port, IP protocol, destination port) triplet.

2.1. Building the Network Graph

In this stage, the raw sample data is used to build a network graph with different node extraction methods for the IP entities set and PPP triplets set.

The IP entities set consists of different network IP entities. Every IP is first enriched with its network category, either internal (local endpoint in the client network), external (a public IP owned by the client network) or public (public internet IP outside the client network). Both internal and external IP entities are assigned with their own node IDs.

Public IPs are processed differently: each public IP is further enriched with its hosting country and organization details, by querying an IP enrichment repository. Then, each unique tuple of (country, organization) is assigned a node ID. Thus, different public IPs that share the same hosting country and organization will be assigned the same ID and treated by the network graph as the same entity. This is done to reduce the graph dimension and improve the model generalization in inference-time for new "unseen" IPs, by ensuring that the same logical communication will not be treated differently because of different raw IP values.

The PPP set consists of triplets in the form of (source port, IP protocol, destination port). It is also pre-processed in order to reduce the network size and improve generalization. First, too-rare IP protocols with a total flow packet count lower than a certain threshold (e.g., 0.1% of the total count) are assigned a general IP protocol identifier, so as not to negatively affect the model learning by their significant imbalanced proportion. Then, an additional pre-processing is done on the port values. Each port value (source/destination) is assigned a category, according to the Internet Assigned Numbers Authority (IANA [9]) port range convention: common service ports have high significance and thus each will be assigned a unique identifier, but client ports have low significance for their specific value and thus will all be assigned a shared identifier.

After both protocols and ports are categorized, every unique triplet of (categorized source port, categorized IP protocol, categorized destination port) is assigned its own node ID. The IP and PPP nodes together represent our vocabulary. An additional dedicated graph node is being created for out-of-vocabulary IP entities and/or PPP triplets, mapping to a general unknown identifier.

Finally, the graph edges are created to transform each flow of (source IP, PPP, destination IP) to two unidirectional edges: the first is between source IP node to PPP node, and the second is between the PPP node to destination IP node. The main building blocks of this stage are summarized in Figure 2.

2.2. Training Vector Embeddings

In this stage we employ a graph embedding technique to distill node features, edge (link) features (between graph nodes) and graph structure information to learn a vector embedding representation to each node. These embeddings provide a "network context" for each of the network entities that will be used in the last stage for learning a link predictor and detecting anomalies.

We will use several edges types:

- By average packet volume to account for different traffic volumes sent over the flow. We will create 10 different edge types by splitting the flow packet count to 10 equal sized bins.
- By time of day. We will create 24 different edge types by splitting the flow by its hour.

We will use several node features, creating a vector for each node consisting of: IP address subnet B ID

1. A flow is defined as a set of IP packets sharing a set of common properties, such as source/destination IP addresses and TCP/UDP ports.

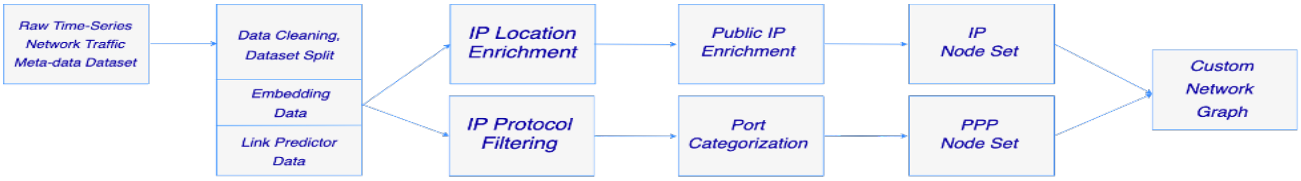


Figure 2. Overview of the main building blocks used for creating the network graph.

(255.255.X.X) , IP address subnet C ID (255.255.255.X), IP address country ID , IP address organization ID and network location category (internal, external or public).

Then, the embeddings will be learned based on the metapath2vec algorithm of graph walks [10]. Each node is sampled for random graph "walks" or paths, starting from the selected node. This process results in a 200-length vector embedding learned for each node for each one of the 36 extracted graphs (10 packet volume average graphs and 24 hourly graphs). Thus, each graph node is represented by a (36, 200) matrix. To reach a one-dimensional node representation, the 36 node vectors are averaged to a single 200-length vector. The main building blocks of this stage are summarized in Figure 3.

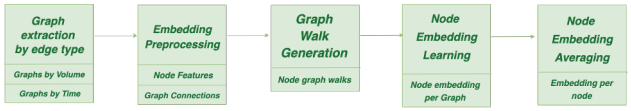


Figure 3. Overview of the main building blocks used for learning graph node embeddings.

2.3. Link Predictor Model

In the final stage, a link predictor model is being trained to detect anomalies. Traditionally, a link predictor model is given a pair of graph nodes and is trained to estimate the likelihood for an edge between them [11].

In our case, the link predictor model will learn to estimate a likelihood for network communications: given a triplet combination (IP node, PPP node, IP node) as input, instead of the traditional 2-node single graph connection. Positive labeled inputs are sampled from the given data, and negative labeled inputs are generated by pairing random triplets that were not part of the given data. Finally, combining the positive and negative datasets forms the input data for the link predictor model.

The link predictor architecture is a feed-forward neural network that consists of the following layers (presented in Figure 4):

- An input embedding layer, with an input size of (3, 200) accounting for input triplet of (IP,PPP,IP), each embedding of size 200.
- A flatten operation combining the 3 input node vectors.
- Two dense layers with ReLU activation.
- A final output dense layer with sigmoid activation to output a likelihood probability.

Training loss will be calculated with binary cross-entropy loss, and network optimization will be done using a stochastic gradient descent algorithm, such as the Adam optimizer.

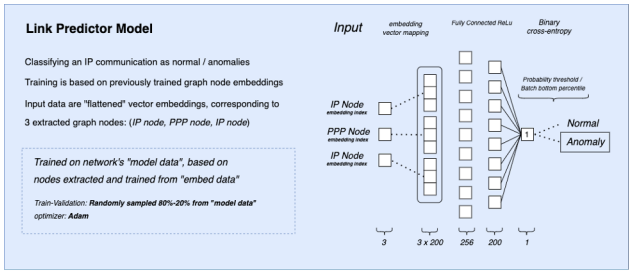


Figure 4. The link predictor model architecture.

At the final stage, model inference will be done in real-time using a predetermined probability threshold, detecting network connections beneath the threshold as anomalies. The threshold will be learned statistically after the model training (e.g., the 99.99th probabilities' percentile).

References

[1] Gilberto Junior, Joel Rodrigues, Luiz Carvalho, Jalal Al-Muhtadi, and Mario Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 2019.

[2] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. *NDSS*, 2018.

[3] Sawsan Abdul Rahman, Hanine Tout, Chamseddine Talhi, and Azzam Mourad. Internet of things intrusion detection: Centralized, on-device, or federated learning? *IEEE Network*, 34(6), 2020.

[4] Ying Zhao, Junjun Chen, Di Wu, Jian Teng, and Shui Yu. Multi-task network anomaly detection using federated learning. In *Proceedings of the 10th international symposium on information and communication technology*, 2019.

[5] Poonam Mehetrey, Behrooz Shahriari, and Melody Moh. Collaborative ensemble-learning based intrusion detection systems for clouds. In *2016 International Conference on Collaboration Technologies and Systems (CTS)*.

[6] Lianbing Deng, Daming Li, Xiang Yao, David Cox, and Haoxiang Wang. Mobile network intrusion detection for iot system based on transfer learning algorithm. *Cluster Computing*, 22(4), 2019.

[7] Baek-Young Choi and Supratik Bhattacharyya. On the accuracy and overhead of cisco sampled netflow. 2005.

[8] Aviv Yehezkel, Eyal Elyashiv, and Or Soffer. Network anomaly detection using transfer learning based on auto-encoders loss normalization. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2021.

[9] Michelle Cotton, Lars Eggert, Dr. Joseph D. Touch, Magnus Westerlund, and Stuart Cheshire. Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry. RFC 6335, 2011.

[10] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. *KDD*, 2017.

[11] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *NIPS*, 2018.

Poster: Exploiting Timing Side-Channel Leaks in Web Applications that Tell on Themselves

Vik Vanderlinden, Tom Van Goethem, Wouter Joosen and Mathy Vanhoef
imec-DistriNet, KU Leuven
{*firstname.lastname*}*@kuleuven.be*

***Abstract*—The performance of remote timing attacks is highly dependent on the network connection that the attack is executed over, where jitter in both the up- and downstream direction can significantly deteriorate an attack's performance. Traditional timing attacks overcome this problem by obtaining a large number of measurements.**

In this poster, we present a technique to remove the inaccuracies caused by downstream jitter in a remote timing attack, which we expect to reduce the number of measurements required to perform a successful timing attack. Our core idea is to exploit timestamps in HTTP responses, whose values are independent of the downstream jitter. To abuse these timestamps, the adversary synchronizes with the target web server's clock edge, after which the observed timestamps allow the adversary to infer secret information.

We present a method to synchronize with the server's clock and discuss how to compensate for the clock drift between the attacker and target machines. To evaluate the feasibility of our technique, we also investigate the occurrence of timestamps in HTTP responses for the top 10,000 sites according to the Tranco list.

***Index Terms*—Side-channel attacks, timing attacks, web-based attacks, network security**

1. Introduction

The first remote timing attack was performed in 2005 by Brumley and Boneh, who used more than 1.4 million samples to leak a 1024-bit RSA key from a server [1]. Historically, to exploit a remote timing attack an adversary has to obtain many samples to be able to differentiate requests in a statistically significant way. Collecting a high number of samples makes the attack more robust against the jitter imposed by the network. By performing a test like the box test proposed by Crosby, Wallach and Riedi, an attacker can confidently differentiate between operations on a server using a quantifiable metric and leak private data [3].

In order to reduce the need for obtaining such a high amount of samples, the network jitter that is present has to be reduced or removed. By eliminating the dependence on one or both network paths, the jitter can be eliminated from the obtained samples. Previous work showed that both up- and downstream jitter can be removed by coalescing multiple requests into a single TCP segment and looking at the order in which the responses are being returned [5]. However, their technique only works over HTTP/2 and requires that the server uses concurrent processing. We propose a new sequential timing attack that eliminates downstream jitter and can leak sensitive information under less strict prerequisites.

First, the core concept of the attack and the prerequisites will be presented. Second, some necessary optimizations to make the attack feasible will be discussed. Specifically, the clocks of the attacker and target machines have to be synchronized in order to leak information from the target. Adding to the complexity, these clocks will experience a relative drift between them over time, due to the minor inaccuracies in the physical hardware-clocks they use. Because the attack takes a non-negligible amount of time, the relative drift between machines should be compensated for in order to keep the synchronization valid throughout the attack. Finally, the occurrence of timing information on the web is discussed due to its vital importance to a successful attack.

2. Proposed Attack

Consider two requests, one of which includes a secret operation that takes additional processing time (the 'target' request), the other does not (the 'baseline' request). When these two requests are sent to a server at exactly the same moment, the expected outcome would be that the response to the target request is returned after the response to the baseline request because additional processing time has passed. Sending both requests can be timed such that the baseline response is returned before some state change on the server and the target response after the state change. The state change is reflected in the respective responses to both the baseline and target requests. The fact that some state is different in both responses can be used by a malicious actor to leak information. One example of state that constantly changes (increments) and is thus a logical choice for an attack is the current time. If a server reflects information about the current time in its responses, this values continuously changes, by definition, over time and can thus be used to leak private information.

The proposed attack eliminates the downstream jitter by exploiting timing information included in HTTP responses from the target server. Because the response is constructed on the server, the timing information that originated on the server travels over the downstream path unchanged while the jitter imposed on this path has no effect on the contents (among which the timing information) of the HTTP response.

3. Clock Synchronization

The amount of responses that are returned close to a target server's clock edge should be maximized to get the largest potential for differentiating between requests. Bringing the moment the response is sent from the target server close to the target clock edge is exactly the goal of the clock synchronization.

An overview of the clock synchronization process is depicted in fig. 1. The middle bar represents the target server, with clock ticks (where the time on the server rolls over to the subsequent value) indicated by vertical bars. Before the clock synchronization is performed, the client has no knowledge about the timing of the server clock ticks (or edges). As shown at the top of fig. 1, the client may send requests but will not necessarily be close to the clock edges of the target server (it is essentially similar to a random initialization). After synchronization, the goal is to have 50% of the responses be returned before the clock edges and 50% after the clock edges. Practically, the client has to find an offset to delay after its local clock tick that moves the sending of the responses on the server-side just the correct amount of time such that they are being returned around the moment of the clock edge, which is shown at the bottom of the fig. 1.

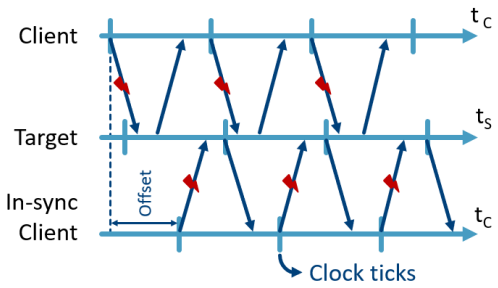


Figure 1. An overview of the effect of clock synchronization. The middle bar represents the target server with clock ticks indicated by vertical bars. The top bar is an unsynchronized client that sends requests to the target server. On the bottom is the same client after the synchronization, now sending requests at an offset in time such that the responses are returned at the exact moments of the target clock ticks. The synchronization process consists of finding the correct offset for the client in such a way that 50% of the responses are returned before and 50% are returned after the target's clock edges.

The synchronization process only uses one request (the 'baseline') that is repeated to perform the synchronization. The attacker machine starts by sending requests over equidistant offsets within an interval (usually one second to start). By observing the responses of the target server, the clock edge may be discovered.

When a response is generated before the clock edge on the target server, consider the time to be t_S. After the clock edge, the time is then incremented to $t_S + 1$. The attacker machine can detect this change and can infer between which offsets the target clock edge occurred. In practice, the detection is less straightforward because the time on the target server is incremented every second and not all requests necessary for the synchronization can be sent within one second. This means that an attacker cannot simply compare the returned timing values directly. Rather, the attacker can use the relative difference between the times on the attacker and target machines, because the time is also incremented on the attacker machine at a more or less similar rate.

When the interval at which the target clock edges occur is found, the attacker can choose to iteratively repeat the process of synchronization within this interval with smaller offsets. This whole process is of course still hindered by the jitter acting upon each request sent over the network. Our preliminary tests show that the synchronization can be performed down to an accuracy of a millisecond with a very low number of required samples. To increase the accuracy of the synchronization further, more optimizations are required, as discussed in the next section.

4. Clock Drift Compensation

Because clocks are inherently inaccurate, be it to a rather small degree that is not disturbing to a human, the clocks of the attacker and target machines may drift away from each other. Relative clock drift has a deteriorating effect on the performed clock synchronization. First because any synchronization that has been successfully performed will only be valid for a small amount of time, until the relative drift will have moved the actual synchronization point away from the detected result. Second, a synchronization of higher resolution may take too such a long time to execute that the actual synchronization point may have already drifted out of focus of the synchronization algorithm (which is iteratively narrowing down on one point) before the synchronization has been completed. Clearly these effects make the attack impossible to perform as is, because an accurate clock synchronization is vital to the success of the attack.

In order to make the attack feasible, the relative drift between the attacker and target machines should be compensated. The absolute drift of a machine's clock can depend on many factors, including but not limited to temperature fluctuations, CPU usage spikes etc. [2]. By using machines in a shared public cloud environment, most factors that may affect the drift are not in control of an adversary. On the other hand, these environments allow an adversary to execute an attack from devices that are physically near the target machines. This means shorter network paths and thus more accurate attacks, thereby incentivizing the use of a public cloud environment.

The relative drift is not stable over time, because it is the sum of two absolute drifts (of the attacker and target machines) which themselves are not stable over time due to the external factors. The relative drift between a machine in our university network in Belgium and a Amazon AWS server in Frankfurt, Germany was monitored for multiple months in 2021. The result depicted in fig. 2 shows that the drift may be approximated as a linear drift most of the time. Note in particular that the time-scale of this figure is large and thus the drift seems extremely large but is actually relatively stable over most periods of a few days. This linear nature gives an adversary the time to approximate the drift close enough to accurately compensate it during the remainder of the attack by taking multiple samples of the target machine clock edge over a few hours and performing a linear interpolation. Most instances at which the drift seems to suddenly change in fig. 2 are reboots of the machine in our university network.

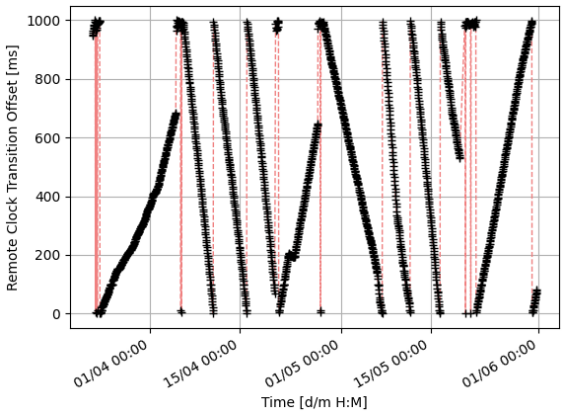


Figure 2. The relative drift between a machine in our university network in Belgium and an instance in the Amazon AWS public cloud environment in Frankfurt, Germany. Sudden changes in drift are mostly reboots of a machine involved in the experiment.

To execute an attack with drift compensation, the drift has to be estimated first, after which all timing values are incremented with the estimated relative drift since the start of the attack (including clock synchronization). Using the drift compensation, we managed to increase the accuracy of the synchronization down to around 10 microseconds. However, it should be noted that this effort significantly increases both the attack time and required number of requests (due to drift compensation and more involved synchronization). An extensive comparison between this method and a standard timing attack regarding the number of requests has yet to be made.

5. Timing Values on the Web

The proposed attack requires the use of timing information embedded in the server's response. This information can either be present in one or more headers or in the body (e.g. embedded within a HTML page) of the HTTP response.

Timing information can occur in different formats. First, there are absolute timing values that represent a specific point in the history or future, usually represented with a year and date. Second, there are relative timing values that represent an offset relative to some absolute time, such as Unix timestamps (the number of seconds since 1 January 1970 UTC). Finally, there are intervals, that define an amount of time and yet are not inherently bound to any absolute time (i.e. an interval such as "5 seconds," is not bound to a specific moment in time).

To detect these different types of timing values, a number of regular expressions have been constructed that match a large number of common formats of timing values. The detection in HTTP headers is a simple string-based match against the each header's value because a HTTP header is essentially a single string. Detection in structured data such as HTML is more difficult. To find timing values in HTML, a bottom-up traversal of the HTML DOM-tree is performed for each visited page.

To evaluate how commonly timing values occur on the web, the Tranco list[1] [4] generated on 22 February 2021 top 10,000 sites were visited by a custom crawler. In total, the crawler successfully visited 41,836 web pages, 5 per site minus some crawler errors. Each page was visited twice with a delay of 10 s between visits. By visiting twice, an easier distinction between variable timing values and static content on a page can be made. It is entirely possible for timing values to occur statically on a site, e.g. the date a blog post was published.

All data gathered was post-processed before any analyses were performed. The post-processing is an important step to filter out static timing values, false positives from the regular expressions and timing values that do not occur consistently on a web page. Because all pages have been visited twice, the post-processor attempts to create a mapping between all found values in both visits and removes those values that are identical on both visits (probably static or false positive results) and that only occur in one out of two visits (no consistent occurrence).

An interesting statistic is that a timing value occurs in the *date* (or *Date*) header in 92.36% of the response documents (of all responses, the document responses with status code 200 are the only ones used for the analyses). Some web pages return a timing value in their *date* header that is unchanged for more than 10 s (the resolution of the timing information is very low). These values are discarded by the post-processing step, being flagged as static values. Even if these values are not entirely static, the granularity of the timing values is too low to use in the attack, thus them being discarded is not an issue.

In total, 5.65% of the documents did not include a single timing value and 37.56% of the responses only have one timing value embedded on them. For most of the responses having one value, this value will be in the *date* header. The median number of timing values per response is 2. The average, however, is 6.39 indicating that there are a number of large outliers. With 990 response including more than 20 timing values and 62 of those even including more than 500 it is clear that these outliers increase the average. Finally, the analyses show that only 22.77% of timing values were found in headers, showing that it is worth including the HTTP response bodies in further analyses.

1. Available at https://tranco-list.eu/list/85NV.

References

[1] D. Brumley and D. Boneh, "Remote timing attacks are practical," in Computer Networks, 48(5), 2005, https://doi.org/10.1016/j.comnet.2005.01.010, pp.701—716.

[2] S. J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," in CCS, 2006, https://doi.org/10.1145/1180405.1180410, pp.27—36.

[3] S. A. Crosby, D. S. Wallach and R. H. Riedi, "Opportunities and Limits of Remote Timing Attacks," in ACM Transactions on Information and System Security, 12(3), 2009, https://doi.org/10.1145/1455526.1455530, pp.1—29.

[4] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński and W. Joosen, "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation," in NDSS, 2019, https://doi.org/10.14722/ndss.2019.23386

[5] T. Van Goethem, C. Pöpper, W. Joosen and M. Vanhoef, "Timeless timing attacks: Exploiting concurrency to leak secrets over remote connections," in Proceedings of the 29th USENIX Security Symposium, 2020, pp.1985-–2002.

POSTER: How Attackers Determine the Ransom in Ransomware Attacks

Tom Meurs
University of Twente
Enschede, Netherlands
t.w.a.meurs@utwente.nl

Marianne Junger
University of Twente
Enschede, Netherlands
m.junger@utwente.nl

Abhishta Abhishta
University of Twente
Enschede, Netherlands
s.abhishta@utwente.nl

Erik Tews
University of Twente
Enschede, Netherlands
e.tews@utwente.nl

***Abstract*—Ransomware may lead to massive economic damage to victims [13]. However, it is still unclear how attackers determine the amount of ransom. In this poster we empirically study the ransom requested by attackers in ransomware attacks. We analysed 371 ransomware attacks reported to the Dutch Police between 2019 and 2021. Our results indicate that attacker's effort and opportunity are important predictors for the ransom requested. The goal of the poster is to invite other researchers for collaboration.**

***Index Terms*—Ransomware, cyber attacks, criminal revenue, police reports**

1. Introduction

Ransomware attacks have become more prevalent over the past years [1]. Even though most ransomware attackers are financially motivated [9], the actual financial gains made by attackers are still unclear. This poster abstract aims at introducing a dataset that could be used to empirically study the ransom requested by attackers.

The Rational Choice Perspective (RCP) [2], [3] states that criminal decision-making is based on weighing the costs and benefits of an attack. Costs could be effort or risk of being caught by Law Enforcement. Benefits is mostly money, but could also be reputation. Based on RCP, we hypothesise that ransom requested depends on how much effort attackers put in a ransomware attack. Furthermore, there might be an increase of requested ransom over the years because improved anti-virus scanners might make it more difficult to perform ransomware attacks and therefore require more effort.

A complementary approach is the Routine Activity Theory (RAT) [10]. RAT focuses on the opportunities for attackers provided by context. From RAT it follows that victims with more money provide the opportunity for attackers to earn more money and therefore will demand higher ransom [6]. Furthermore, opportunity might vary between seasons [12], so requested ransom could also vary between seasons.

Summarizing, attacker's effort and opportunity could influence ransom requested (Figure 1) .We propose the following hypotheses:

- H1: If attackers put in more effort they will ask a larger ransom
- H2: There is an increasing trend of requested ransom over the years
- H3: High revenue of victims should lead to larger requested ransom
- H4: The requested ransom varies over the different seasons

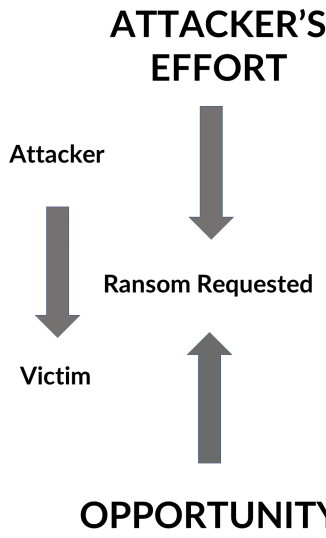


Figure 1: Theoretical framework

2. Methods

2.1. Sample

We investigated 371 ransomware attacks registered by the Dutch Police between 2019 and 2021. Ransomware attacks on individuals as companies were included in the sample. Ransomware attacks with victims outside the Netherlands, but reported to the Dutch Police, were excluded from this study.

2.2. Measures

To measure requested ransom information about the ransom at first contact with victims was collected. From the 371 observations, 172 attacks (46 %) reported ransom demanded by attackers. If ransom demanded was unknown, this was mostly (52 %) because attackers wanted victims to contact them to inform them about the ransom and the victim did not want to do so. In our analysis we perform analysis to see if there is selection bias of the unknown requested ransomware.

To measure effort information was collected on several variables.

a) Ransom note. We noted whether the criminals wanted to first have contact with the victim before informing what ransom they requested, from here defined as targeted ransom note (categories: yes/no). Yes means that first contact with the attackers was required to obtain information about the ransom. No means that the ransom was stated on the ransom note.

b) Exfiltrated of data measured whether data from the victim were exfiltrated (categories: yes/no).

c) Collaboration with other criminals, measures whether the attackers made use of RaaS [12] or whether they collaborated with other groups to perform the attack (categories: yes/no).

d) What type of access was used to infiltrate victim's network (categories: exploit/phishing/different),

e) Network Attached Storage measures whether attackers targeted the Network Attached Storage device (categories: NAS, yes/no).

f) Furthermore, the name of group that executed the attack was included if more than 5 attacks were observed, the rest was aggregated to the variable 'different'. We assumed that groups vary in the amount of effort used in attacks, and therefore might also vary on the required ransom.

To measure opportunity information was collected on several other variables: We noted

a) Company size, which was based on staff and

b) Yearly revenue.

Additional variables included:

c) Insurance (categories: yes/no),

d) Economic sector (categorized by the Dutch Chamber of Commerce),

e) Type of victim (categories: corporate/governmental/individual) and

f) Backups (categories: no/yes, but not possible to recover of data/yes, but could partially recover data / yes, and could fully recover data).

Temporal aspects were:

g) Season and

h) Year.

We assume that the context provides more opportunities when there are higher profits to be made: a victim with high revenue might be able and willing to pay a higher ransom. We did not have particular expectation on seasonality.

2.3. Analysis

First, we analyse selection bias on whether the requested ransom is known and correct with Heckman's two-step procedure [4], [7]. Second, we performed multiple imputation analysis using the R-package Mice [5]. Third, stepwise regression was performed on different models to find a parsimonious model. From explanatory analysis we found that ransom demanded, yearly turnover and staff were highly skewed, so we take the logarithm.

3. Results

Our final model (1) from the stepwise regression:

$$\begin{aligned} Log_{10}RR = 1.44 + 0.72DE + 1,03TR \\ + 0.51RaaS – 0.99NAS \\ + 0.32Log_{10}REV – 2.25IMR + \epsilon \end{aligned} \quad (1)$$

Where RR is the ransom requested, DE is data exfiltrated $\in \{0,1\}$, TR is targeted ransom note $\in \{0,1\}$, RaaS $\in \{0,1\}$, NAS $\in \{0,1\}$, REV is yearly turnover, and IMR is inverse Mills Ratio. IMR is the correction for the selection bias.

The coefficients should be interpreted as follows (Figure 2): 3034 euro is the geometrical mean. If data was ex-filtrated, the ransom requested would be +430% above the geometrical mean. If the ransom note was targeted, the ransom requested would be +980%. If the group was RaaS, the ransom requested would be +220%. If the attack targeted a NAS, the ransom requested would be -90%. An increase of the yearly turnover with 1% would increase the ransom requested with 0.32 %. Finally, the negative IMR means that the expected RR observed in this sample is lower than the expected RR in the population.

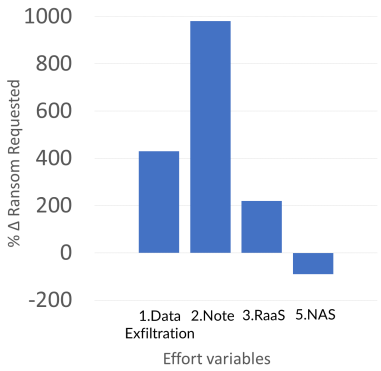


Figure 2: Significant results of effort variables on ransom requested.

No heteroskedasticity was found in (1) (Breusch-Pagan (6) = 10.171, p = 0.12). Residuals ϵ are normaly distributed (Shapiro-Wilk=0.993,p=0.08). To test the sensitivity of the imputation of missing values we performed the same regression on ransom requested without the observation with missing values and found the same coefficients to be significant ($\alpha = 0.05$).

The IMR was esimated using the probit regression:

$$RR_B = \gamma_1' IR + \gamma_2' TR + \nu \quad (2)$$

Where $RR_B \in \{0,1\}$ is whether the ransom requested was known in this sample, IR is whether there was a Incident Reponse company helping the company to recover after the attack and targeted ransomnote whether the criminal first wanted to make contact with the victim before telling how much ransom they requested for the decryption keys. With (2) we can estimate γ and therefore the IMR :

$$IMR = \frac{\phi(\hat{\gamma_1}' IR + \hat{\gamma_2}' TR)}{\Phi(\hat{\gamma_1}' IR + \hat{\gamma_2}' TR)} \quad (3)$$

For a detailed explanation why (2) and (3) lead to

unbiased, consistent and efficient estimators in (1) when selection bias is present, we refer to [5].

In conclusion, data exfiltration, targeted ransomnote, RaaS, NAS, revenue predict the ransom requested by attackers in ransomware attacks. Insurance, sector, backups, corporate/governmental/individuals and ransomware group, year and season do not influence the ransom requested.

Our results indicate that criminal effort and victim's company size influence the ransom demanded, but that temporal aspects did not effect the amount of ransom demanded. Furthermore, we found selection bias in this sample: the ransom requested in the population should be larger than in the present sample. Surprisingly, having backups did not effect the ransom requested. Also companies being insured did not lead to higher requested ransom, although this might be the result of low amount of observations. Furthermore we also did not find a relationship between year and season and ransomware requested.

4. Discussion and Conclusion

In this study we modeled the way ransomware actors determine how much ransom they demand during a ransomware attack. Regarding our hypotheses we found:

- H1: If attackers put in more effort they will ask a larger ransom, as expected.
- H2: There is no increasing trend of requested ransom over the years, in contrast with expectations
- H3: High revenue of victims should lead to larger requested ransom, as expected
- H4: In contrast with previous findings does the requested ransom not vary over the different seasons

Furthermore, we found there was a selection bias regarding requested ransomware in this sample and that the observed requested ransomware was an underestimation for the true requested ransomware in our sample.

There are at least two potential limitations concerning the results. First, the selection of attacks registered to the Dutch Police may be a (biased) subset of all attacks in the Netherlands. However, if the Dutch Police notices that there are Dutch victims who did not notify the Police, they will pro-actively contact the victim. A second potential limitation is that for some variables there was a lot of missing information.

Despite these limitations, this study is the first to systematically analyse ransomware with different ways of measuring effort and contextual variables. Whereas past researchers have used around 50 attacks and mostly companies who were the victim [8], the present study analyzed 371 attacks with individuals, companies and government as victim.

Although the generality of the current results must be established by future research, the present study has provided clear support that RCP and RAT are applicable to ransomware attacks. These findings suggest several courses of action for policy makers and Law Enforcement. First, improve prevention by reaching out to potential high-risk victims. Second, frustrate the ransomware process by increasing efforts of attackers for a successful ransomware attack.

References

[1] Oz, H., Aris, A., Levi, A., Uluagac, A. S. (2021). A survey on ransomware: Evolution, taxonomy, and defense solutions. arXiv preprint arXiv:2102.06249.

[2] Hechter, M., Kanazawa, S. (1997). Sociological rational choice theory. Annual review of sociology, 23(1), 191-214.

[3] Cornish, D. B., Clarke, R. V. (1989). Crime specialisation, crime displacement and rational choice theory. In Criminal behavior and the justice system (pp. 103-117). Springer, Berlin, Heidelberg.

[4] Heckman, J. J. (1979). Sample selection bias as a specification error. Econometrica: Journal of the econometric society, 153-161.

[5] Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in R. Journal of statistical software, 45, 1-67.

[6] Galinkin, E. (2021). Winning the Ransomware Lottery: A Game-Theoretic Model for Mitigating Ransomware Attacks. arXiv preprint arXiv:2107.14578.

[7] Bushway, S., Johnson, B. D., Slocum, L. A. (2007). Is the magic still there? The use of the Heckman two-step correction for selection bias in criminology. Journal of quantitative criminology, 23(2), 151-178.

[8] Yuryna Connolly, L., Wall, D. S., Lang, M., Oddson, B. (2020). An empirical study of ransomware attacks on organizations: an assessment of severity and salient factors affecting vulnerability. Journal of Cybersecurity, 6(1).

[9] Hassan, N. A. (2019). Ransomware revealed: a beginner's guide to protecting and recovering from ransomware attacks. Apress.

[10] Cohen, L. E., Felson, M. (1979). Social change and crime rate trends: A routine activity approach. American sociological review, 588-608.

[11] Junger, M., Wang, V., Schlömer, M. (2020). Fraud against businesses both online and offline: crime scripts, business characteristics, efforts, and benefits. Crime science, 9(1), 1-15.

[12] Huang, K., Siegel, M., Madnick, S. (2018). Systematically understanding the cyber attack business: A survey. ACM Computing Surveys (CSUR), 51(4), 1-36.

[13] Oosthoek, K., Cable, J., Smaragdakis, G. (2022). A Tale of Two Markets: Investigating the Ransomware Payments Economy. arXiv preprint arXiv:2205.05028.

Poster: One of a Kind: Correlating Robustness to Adversarial Examples and Face Uniqueness

1st Giuseppe Garofalo
imec-DistriNet, KU Leuven
giuseppe.garofalo@kuleuven.be

2nd Tim Van hamme
imec-DistriNet, KU Leuven
tim.vanhamme@kuleuven.be

3rd Davy Preuveneers
imec-DistriNet, KU Leuven
davy.preuveneers@kuleuven.be

4th Wouter Joosen
imec-DistriNet, KU Leuven
wouter.joosen@kuleuven.be

***Abstract*—Face authentication lacks key metrics to assess the robustness of users' representation within the system. We fill the gap by investigating face uniqueness, which is the distinctiveness of a face within a population, as a proxy for robustness against adversarial examples. By generating malicious input that escapes face verification, a dodging attack, we show a correlation between the amount of perturbation needed for successfully attacking a user and their uniqueness within a dataset. Our experiments span over multiple networks under a realistic threat model, indicating that unique users are significantly more resilient to gradient-based attacks than non-unique ones.**

***Index Terms*—component, formatting, style, styling, insert**

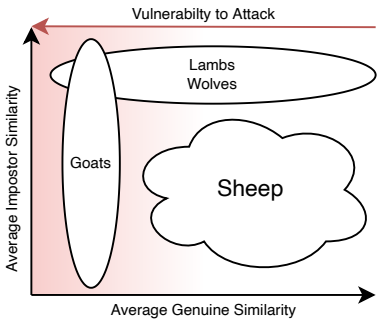


Figure 1. The four major classes of the Doddington's Zoo with, overlapped in the background, the vulnerability to evasion attacks.

1. Introduction

Unique faces are those that are decidedly different from the rest of the population while being easy to recognize [1]. In modern face recognition, biometric uniqueness is directly affected by the separation between two distributions: the scores originated from matching two samples of the same user, i.e. the *genuine distribution*, and the scores derived from matching samples of different users, i.e. the *impostor distribution*.

It is well known that different faces exhibit varying performance within a system [2], which is linked to their relative uniqueness within a dataset. These performance are mainly expressed in terms of False Acceptance Rate (FAR), which comes from the mislabeling of a user, and False Rejection Rate (FRR), which is failing to match two samples of the same user. Identifying groups of users who contribute disproportionately to a type of error can uncover their vulnerabilities, eventually improving their resilience. The Doddington's Zoo [3], shown in Fig. 1, is the first attempt to categorize users based on their verification performance, dividing between the score distribution of classes that cause the errors (goats, lambs, and wolves) and the distribution of the average user (sheep). The existence of these classes was later confirmed for a number of biometric modalities, including faces, and expanded to new classes in a concept known as *biometric menagerie* [2].

However, studies on the menagerie are usually disconnected from those on the security of modern face recognition. The advent of deep learning has boosted face matching accuracy while broadening the threat surface. The assumption that training and test data are independent and identically distributed (i.i.d.) has proven to be hard to satisfy in practice, hence shifts in the data distribution affect algorithm performance and lead to poor generalization. In a malicious setting, imperceptible modifications known as adversarial examples can fool a system into assigning the wrong label to its input [4]. This shift represents a violation of the i.i.d. assumption, contributing to the overall FAR and FRR of a system in a way previously not envisioned by animal categorizations like the Doddington's Zoo. Users who are contributing to the errors of a system (e.g. goats and lambs) need to be reconsidered in light of the novel threats posed by dataset shifts.

In our analysis we use a measure of uniqueness based on entropy to gather novel insights on the robustness of face recognition systems against adversarial examples. Motivated by modern tools [5], we construct a realistic yet simplified scenario of image publishing, where a user hides their identity before uploading a picture online. The attacker creates a human-imperceptible adversarial mask with the aim of fooling a face verification system that performs 1-vs-1 matching between a pair of images, which is called a dodging-attack. The amount of perturbation needed to escape matching will serve as an indicator of robustness towards an attack. We show that this notion of robustness is correlated with how well a user is embedded in the feature space, i.e. entropy-based uniqueness.

2. Methods

Our analysis can be divided in the following steps:

1) We compute the uniqueness of a set of users via Kullback-Leibler (KL) divergence estimation.
2) For a subset of the total users, we generate adversarial examples and derive attack robustness.
3) We correlate between uniqueness and robustness.
4) We quantify the unique and non-unique identities shared between networks.

2.1. Computing Uniqueness

Balazia et. al [1] approximate the KL divergence by using a distance estimator $D(x, S)$ that measures the average dissimilarity between a vector x and a set of vectors S in the embedding space. If Therefore, assuming $|G| = |I|$, the *uniqueness* U is equal to

$$KL(p_g|p_i) \approx U(G, I) = \frac{1}{|G|} \sum_{g \in G} \log \frac{D(g, I)}{D(g, G)} \quad (1)$$

where $D(g, G)$ measures the average distance of a template g from embeddings of the genuine distribution G, and $D(g, I)$ performs the same measurement towards the embeddings of other users, the impostor distribution I. It is possible to separate the contribution of the genuine scores, i.e. intra-class, from the one relative to the distance between genuine and impostor, i.e. inter-class:

$$U(G, I) = InterU(G, I) + IntraU(G) \quad (2)$$

$$InterU(G, I) = \frac{1}{|G|} \sum_{g \in G} \log D(g, I) \quad (3)$$

$$IntraU(G) = -\frac{1}{|G|} \sum_{g \in G} \log D(g, G) \quad (4)$$

2.2. Computing Attack Robustness

We define the robustness of an image, and by extension of a user, by computing the amount of perturbation in the input space needed for the adversarial example x_{adv} to cross the verification threshold θ, causing the mislabeling. We define this measure as Lowest Perturbation Budget (LPB). Given an embedding network f and a perturbation budget ϵ, we maximize the distance D:

$$\arg\max D(f(x_{adv}), f(x)), \ s.t. \ \|x_{adv} - x\|_p < \epsilon \quad (5)$$

Since ϵ is a parameter to be decided before carrying out the attack, the LPB of an image x, and by extension of a user, is found by performing a binary search:

$$LPB(x) = \min \epsilon, \ s.t. \ D(f(x_{adv}), f(x)) < \theta \quad (6)$$

TABLE 1. NETWORKS PERFORMANCES ON THE LFW DATASET.

Model	ACC	$TAR_{0.05\%}$	U	IntraU	InterU
FT	99.82	99.73	0.527	-2.939	3.467
FTo	99.75	99.60	0.535	-2.932	3.467
MF	99.43	98.20	0.437	-3.029	3.466
IR50-S	99.60	98.17	0.680	-2.786	3.466
IR50-C	99.68	99.17	0.512	-2.955	3.466
FN	99.23	85.80	0.730	-2.735	3.465

2.3. Correlation Analysis

Having computed the $U_{k,f}$ and $LPB_{k,f}$ for each user k, from embeddings computed using a model f, we are interested in the strength of the association between the two variables. We perform a correlation analysis by computing Kendall's τ score. Fixing a model f, given n pairs (U_x, LPB_x),

$$\tau = \frac{n_c - n_d}{\binom{n}{2}} \quad (7)$$

where n_c denotes the number of concordant pairs and n_d is the number of discordant pairs. A pair is concordant if given two users i and j, (U_i, U_j) and (LPB_i, LPB_j) have the same ordinal relationship (vice versa they are discordant). Therefore, τ measures the pairwise ordinal concordance between two variables, which is their monotonic relationship.

3. Results

In this section, the experimental setup is followed by an analysis of the results.

Experimental setup. Inspired by anti-facial recognition tools [5], we create adversarial examples by using the widely adopted gradient-based strategy FGSM and its iterative version BIM, both under the l_2 and l_{inf} norms. The attack is white-box and does not include a defensive strategy, which goes beyond the scope of our analysis. The attacked embedding networks are representative of the spectrum of SoTA face recognition solutions: FaceTransformer with and without overlapping patches (FT and FTo), MobileFace (MF), Inception-ResNet trained with a softmax and CosFace loss function (IR50-S and IR50-C), and FaceNet (FN). They share similarities that allow to selectively exclude the contribution of certain covariates when we focus on one single aspect of the models. *RobFR* [6] provides the backbone implementations on top of which we perform our white-box attack [1].

As test dataset, we pick the Labelled Faces in the Wild (LFW) [7]. LFW has the advantage of a big sample size, variability, and does not overlap with the training dataset of the attacked networks. From LFW we derive two subsets: *lfw-U* which is used to compute uniqueness, and *lfw-R* which is a further refined sample list to compute LPB scores.

Analysis. Table 1 displays the Uniqueness U and its intra-class and inter-class components. Surprisingly, the

1. https://github.com/ShawnXYang/Face-Robustness-Benchmark.

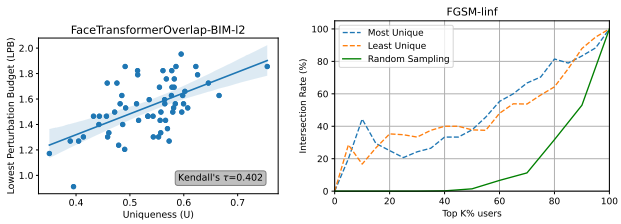


Figure 2. On the left, linear regression between LPB and U with CI=95%. On the right, percentage of shared identities across models as we increase the most or least unique users.

network with the highest system uniqueness, i.e. FaceNet, is the worst performing one on LFW. This is because U delves deep into the score distribution, providing information that go beyond the average case.

Table 2 shows a moderate-to-strong correlation between the uniqueness of a user U_k and the average lowest perturbation budget needed to escape verification LPB_k. IntraU closely resembles the correlation of the overall U. This is expected since IntraU explains most of variance of U (see Table 1). Differently, the correlation analyses between LPB and InterU show little negative correlation and are, in most cases, not statistically significant (p-value greater than 0.05). Nonetheless, FaceTransformerOverlap scores have a weak negative correlation in the case of the BIM attack that is significant and worth of further investigations. Fig. 2 (left) shows the U score as a function of LPB for FaceTransformerOverlap and the (BIM,l_2) attack.

The results have an interpretation from a Doddington's Zoo perspective (Fig. 1). Goats are users whose intraU is particularly low and are therefore especially susceptible to the class of attacks under study. On the other side of the spectrum, lambs, sheep and wolves are all eligible to containing a set of users with increased robustness compared to the average case. These users are the ones with higher U (and IntraU). Animal groups have been used to create adapting strategies that move verification threshold in order to cope with, e.g., large intra-class variance [8]. While adaptive thresholding can positively affect the FAR of the system in the benign case, moving the threshold triggers a cascade effect on the perturbation budgets needed to escape verification. Hence these strategies should account for the consequences on the robustness of the users.

Fig. 2 (right) plots the intersection rate between all the models as a function of the most unique and least unique K% users for the (FGSM,l_{inf}) configuration [2]. To highlight the significance of the intersection rates, a baseline is added to the graph showing the intersection rates for a random sampling of users repeated 6 times (as many as the number of considered models).

4. Conclusion

Our analysis underlines the strong existing correlation between the resilience of a user against *dodging attacks* and the their distinctiveness within the population. This gives a clear indication whether a face is harder to protect

2. Notably, the results hold for all the combinations *(method, norm)*.

TABLE 2. KENDALL'S τ BETWEEN THE U SCORES AND LPB OF THE USERS IN THE *lfw-U* LIST. A MODERATE-TO-STRONG POSITIVE CORRELATION IS FOUND FOR U AND INTRAU.

Model	Attack	Norm	τ_U	τ_{IntraU}	τ_{InterU}
FT	BIM	l2	0.38	0.39	-0.17
FT	BIM	linf	0.34	0.34	-0.15
FT	FGSM	l2	0.38	0.38	0.00
FT	FGSM	linf	0.44	0.44	0.02
FTo	BIM	l2	0.40	0.41	-0.26
FTo	BIM	linf	0.35	0.36	-0.26
FTo	FGSM	l2	0.39	0.39	-0.08
FTo	FGSM	linf	0.46	0.46	-0.11
MF	BIM	l2	0.45	0.46	-0.16
MF	BIM	linf	0.40	0.41	-0.19
MF	FGSM	l2	0.41	0.42	-0.08
MF	FGSM	linf	0.45	0.45	-0.09
IR50-S	BIM	l2	0.43	0.43	0.03
IR50-S	BIM	linf	0.39	0.40	0.04
IR50-S	FGSM	l2	0.43	0.42	0.05
IR50-S	FGSM	linf	0.47	0.46	0.07
IR50-C	BIM	l2	0.49	0.49	-0.17
IR50-C	BIM	linf	0.45	0.45	-0.17
IR50-C	FGSM	l2	0.41	0.39	-0.10
IR50-C	FGSM	linf	0.47	0.45	-0.09
FN	BIM	l2	0.44	0.45	-0.20
FN	BIM	linf	0.41	0.42	-0.16
FN	FGSM	l2	0.37	0.39	-0.16
FN	FGSM	linf	0.43	0.43	-0.14

using Anti-Facial recognition [5], emerging privacy tools that rely on adversarial perturbations. Our exploration of the embedding space can be expanded beyond the class of dataset shift we take into consideration for our experiments, to account for different attacks, like impersonation, and benign covariate shifts, like changes in lighting conditions. The implications of our findings range from a better characterization of biometric system performance to a new understanding of what makes a face, therefore a user, more resilient against gradient-based adversarial attacks.

References

[1] M. Balazia, S. Happy, F. Brémond, and A. Dantcheva, "How unique is a face: An investigative study," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 7066–7071.

[2] N. Yager and T. Dunstone, "The biometric menagerie," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 2, pp. 220–230, 2008.

[3] G. Doddington, W. Liggett, A. Martin, M. Przybocki, and D. Reynolds, "Sheep, goats, lambs and wolves: A statistical analysis of speaker performance in the nist 1998 speaker recognition evaluation," National Inst of Standards and Technology Gaithersburg Md, Tech. Rep., 1998.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[5] E. Wenger, S. Shan, H. Zheng, and B. Y. Zhao, "Sok: Anti-facial recognition technology," *arXiv preprint arXiv:2112.04558*, 2021.

[6] X. Yang, D. Yang, Y. Dong, W. Yu, H. Su, and J. Zhu, "Delving into the adversarial robustness on face recognition," *arXiv preprint arXiv:2007.04118*, 2020.

[7] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.

[8] A. Mhenni, E. Cherrier, C. Rosenberger, and N. E. B. Amara, "Analysis of doddington zoo classification for user dependent template update: Application to keystroke dynamics recognition," *Future Generation Computer Systems*, vol. 97, pp. 210–218, 2019.

Poster: Pillars of Sand: The current state of Datasets in the field of Network Intrusion Detection

Gints Engelen
imec-DistriNet
KU Leuven
Leuven, Belgium
gints.engelen@kuleuven.be

Robert Flood
University of Edinburgh
Edinburgh, UK
s1784464@ed.ac.uk

Lisa Liu
School of Engineering and IT
University of New South Wales
Canberra, Australia
l.liuthorrold@adfa.edu.au

Vera Rimmer
imec-DistriNet
KU Leuven
Leuven, Belgium
vera.rimmer@kuleuven.be

Henry Clausen
University of Edinburgh
Edinburgh, UK
henry.clausen@ed.ac.uk

David Aspinall
University of Edinburgh
London, UK

Wouter Joosen
imec-DistriNet
KU Leuven
Leuven, Belgium
wouter.joosen@kuleuven.be

***Abstract*—Network Intrusion Detection Systems play a critical role in protecting network architectures from harm. In the past decade, Machine Learning has moved to the forefront of research in this field, with many approaches resulting in great performance on benchmark NIDS datasets. The relevance of these performance results is however directly tied to the quality of the benchmark datasets used for training, which have so far not been subjected to thorough analysis. As part of our work, we have performed a large-scale manual investigation of the most commonly used publicly available NIDS datasets, where we have uncovered numerous errors due to problems in data pre-processing, attack simulation and labelling. We also highlight the lack of variability in both benign and malicious traffic, which often renders the classification task trivial. To quantify this variability, we have devised an automated methodology that can be applied without requiring expert domain knowledge. Nevertheless, we believe it is vital for any NIDS benchmark datasets to undergo a thorough manual analysis before being widely adopted. As a follow-up of our previous work where we provided an improved version of the CICIDS 2017 dataset, we are also actively working on improving the CSE-CIC-IDS 2018 dataset, which we intend to release to the research community.**

***Index Terms*—network intrusion detection, machine learning, benchmark dataset, data collection.**

1. Introduction

Network Intrusion Detection Systems (NIDS) are devices that are placed at strategic locations within a network infrastructure in order to protect it from internal and external threats. These threats range from attempts to gain unauthorised access to the network, to large-scale DDoS attacks that aim to disrupt the services of the network's hosts. With attackers becoming more sophisticated, new threats are emerging on a daily basis, and traditional rule-based Intrusion Detection Systems are at risk of being overwhelmed by the sheer number of zero-day attacks. This is why, over the past decade, NIDS research has gravitated towards Machine Learning (ML), which does not rely as much on manual updates in order to detect new attacks.

Research in this field has shown a lot of promise. High performance results on benchmark NIDS datasets [1]–[3] seem to indicate that using ML for NIDS is a solved problem. However, the obtained results are heavily dependent on the quality of the used datasets. While some research has highlighted the disparity between traffic found in these datasets and that of a real-world environment [4], the network traffic found in these datasets has generally not been subjected to a thorough manual investigation.

In earlier work, we have shown that the CICIDS 2017 dataset [5] suffered from a multitude of issues which made its use as a benchmark dataset questionable [6]; as part of that work we released an improved version of this dataset. Following up on that, we have performed a large-scale manual analysis of five modern and widely-used NIDS datasets, where we uncovered a wide range of issues pertaining to labelling, attack simulation, documentation, and network traffic realism.

As part of this work, we are working on improving the CSE-CIC-IDS 2018 dataset, specifically making sure that the ground truth of all labels is as accurate as possible. Using our fixed version of the CICFlowMeter tool [7] guarantees that this version of the dataset is also free from the kind of artefacts (*such as TCP Appendices*) that were present in the CICIDS 2017 dataset [6].

Finally, while we believe that any modern NIDS dataset will have to be subjected to intense scrutiny before seeing widespread adoption in the field, we also propose some automated techniques that could help give an overview of the overall quality of a NIDS dataset.

2. Background and Motivation

Machine Learning has already been applied with great success in other fields, notably due to its capability of learning a task while requiring minimal human supervision. Typical ML approaches do not directly operate on

raw data; instead, the data first goes through a process called *feature extraction* before being fed into the model.

When it comes to network traffic, a common way to pre-process the data is by grouping packets together into so-called *flows*, identified by a 5-tuple of {*Src IP, Dst IP, Src Port, Dst Port, Protocol*}. Typical flow-level features are, for example, *total forward packets*, *bytes sent per second*, and *total flow duration*.

The vast majority of ML approaches require significant amounts of training data. In contrast to fields like Computer Vision and Natural Language Processing, where training data is abundantly present, collecting training data in the field of Network Intrusion Detection is more challenging. This is mainly due to the inherent privacy concerns that come with collecting network traffic.

Research in this domain has tried to alleviate the problem of a lack of datasets by creating synthetic datasets in a controlled network environment, which were then made publicly available. Given the high performance results obtained by ML-based approaches on these datasets, it is striking that adoption of ML for NIDS in industry has been comparatively slow. One of the problems lies in the fact that these synthetically generated datasets were immediately adopted in the field, with comparatively little research done analysing the quality and validity of these datasets.

Due to the large variety and variability of network traffic as well as a rapidly evolving threat landscape, Network Intrusion Detection in a real world setting is hard. Benchmark datasets serve to certify a model's capability of successfully operating in such a complex real world environment, and so the classification of traffic in a benchmark dataset should be equally difficult. Lastly, correct ground truth of all labels in a benchmark dataset is crucial in order to verify that the model successfully learned the classification task at hand.

3. Datasets and Methodology

In order to help improve the quality and utility of NIDS datasets, we perform a large-scale manual analysis of 5 widely-used datasets. As part of our selection criteria we strive to include datasets that have been cited numerous times (>100), and where authors published both the raw traffic files (PCAPs) as well as feature-extracted data (flows). We also included 2 IoT-based datasets - despite not meeting the citation criteria - in order to diversify the types of network architectures that we analyse.

The selected datasets are CSE-CIC-IDS 2018 [8], UNSW NB15 [9], TON-IoT [10] and IoT-23 [11]. Due to some updates in our methodology we also revisited CICIDS 2017 [5], despite already having analysed it once in our previous work [6].

3.1. Verifying label correctness

In this step we focus our efforts on the malicious labels. When analysing the validity of these labels, we verify that the underlying network traffic actually exhibits characteristics that would be indicative of malicious traffic of the type specified by the label. Additionally, we verify as much as possible that the attack has been correctly implemented and that it has an appreciable effect on the target victim.

One key aspect that should facilitate this task is the documentation of the dataset generation process published by the dataset authors. For all analysed datasets, however, we found that a significant amount of information was lacking when it comes to the way that most attacks were implemented. Instead, we had to rely on manual analysis of the raw network traffic in order to understand the nature of the attack, and verify correctness of its given label.

For CSE-CIC-IDS 2018, as part of this manual analysis we also took the opportunity to fully reverse-engineer and subsequently improve on the existing labelling logic.

3.2. Network traffic variety

As mentioned previously, the varied nature of both benign and malicious network traffic should make the classification task quite hard. Moreover, we can expect that a smart adversary will try to make his malicious traffic appear to be as similar to benign traffic as possible. For simpler attacks such as DoS and DDoS, we expect that the attack traffic shares some characteristics with large spikes of normal traffic. In summary, malicious traffic found in benchmark datasets should not be *too* dissimilar to benign traffic, as this would otherwise simplify the classification task considerably.

In order to measure this similarity, we employ 4 different "metrics":

- Overlap Region Volume: Calculates the amount of overlap between data points of 2 different classes
- Maximum Fischer's Discriminant Ratio: Gives an indication of how easily data from 2 classes can be separated
- 1-Nearest Neighbours Ratio: Assigns to each datapoint the class of its nearest neighbour. A very high number for a given class indicates that there is very little overlap with other classes, again pointing to a trivial classification task for the model.
- Few-Shot Learning Accuracy: Using a Siamese model, we feed it 2 samples at a time and have it learn whether they belong to the same class or not. We then observe how many samples it needs to train on before reaching an accuracy score of over 95%.

4. Results

4.1. Manual Analysis

As part of our manual analysis, we found numerous problems which can be grouped into five categories:

- Repetitive Attack traffic: Attack traffic that is repetitive to the extent that the classification problem is considerably simplified. While some attack classes exhibit this characteristic as part of their nature (e.g. Portscan traffic), we expect most attack classes to each contain a decent variety of attack traffic.
- Unclear Attack: Attacks where we are unable to identify the attack taking place or why the attack

Mistake Type	Majority Affected	Partially Affected	Does Not Apply
Repetitive	22	5	40
Unclear Attack	10	1	56
Ineffective Attack	5	1	61
Simulation Artifacts	16	1	50
Mislabelled Data	6	14	47
Total	41	16	10

TABLE 1. NUMBER OF ATTACK CLASSES SUFFERING FROM EACH PROBLEM

should be considered as malicious. For instance, downloading a file from a server.

- Ineffective Attack: The attack fails to have an effect on the target victim.
- Simulation Artifacts: Unintended side effects of the dataset generation process which lead the trained classifier to exhibit shortcut learning. This category includes artifacts from the feature extraction process.
- Mislabelled Data: The attack class contains benign traffic or attack traffic from another class.

A breakdown for the presence of these problems across all classes of the five datasets is given in table 1. We say that a class is *majorly* affected by a problem if it is present in more than 50% of the flows. If the number of flows is below that threshold, we say that the class is *partially* affected.

4.2. Automated Traffic Analysis

So far, results of our automated analysis help explain why classifiers trained on these dataset easily reach very good performance numbers: for almost every malicious class in our tested datasets, the Overlap Region Volume was below 0.1. This means that the distributions of this class and the benign traffic overlapped minimally. For many cases the ORV value was 0, meaning the overlap limits itself to at most one single datapoint. The Maximum Fischer's Discriminant and 1-Nearest Neighbours ratios further confirmed this.

Across virtually all classes, our Siamese network was capable of achieving over 95% accuracy after only having seen 200 samples. In some cases, such as the UNSW NB15's *Worms* attack, it achieved over 99% accuracy after seeing less than 50 samples.

5. Discussion and Future Work

Based on our findings, we can ascertain two things.

Firstly, for each dataset our manual analysis revealed numerous problems with the data labelling process, attack setup and/or feature extraction process. Secondly, our automated analysis found that the malicious traffic shared little similarities with underlying benign traffic, meaning that the classification task is far from challening.

We believe the above-mentioned issues are serious to the extent that these datasets can not be claimed to be representative of real-world traffic. As such, good performance results obtained by classifiers on these benchmark datasets can not be expected to generalise to a live production environment.

As part of our work, we are also working on an algorithm that would help detect mislabeled samples in a NIDS dataset, as manual analysis of each individual data point is infeasible. Finally, we will publish the fixed CSE-CIC-IDS 2018 dataset, as well as the extended documentation containing the details of all of our findings on our website [7].

6. Conclusion

In our work, we are performing a large-scale analysis of five popular NIDS datasets. We uncovered numerous problems in the dataset generation process, and additionally found that the malicious traffic found in these datasets is often easily distinguishable from benign traffic. As a result, existing benchmark datasets are not suitable to evaluate a classifier's capability to be succesfully deployed in a real-world environment. As part of our work we hope to pave the way for better NIDS datasets, and we stress again that we believe a dataset should be subjected to a thorough manual analysis by domain experts before being adopted by the wider research community.

References

[1] L. Leichtnam, E. Totel, N. Prigent, and L. Mé, "Sec2graph: Network attack detection based on novelty detection on graph structured data," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2020, pp. 238–258.

[2] A. R. Gad, A. A. Nashat, and T. M. Barkat, "Intrusion detection system using machine learning for vehicular ad hoc networks based on ton-iot dataset," *IEEE Access*, vol. 9, pp. 142 206–142 217, 2021.

[3] J. Yoo, B. Min, S. Kim, D. Shin, and D. Shin, "Study on network intrusion detection method using discrete pre-processing method and convolution neural network," *IEEE Access*, vol. 9, pp. 142 348–142 361, 2021.

[4] H. Hindy, D. Brosset, E. Bayne, A. K. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104 650–104 675, 2020.

[5] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[6] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the cicids2017 case study," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 7–12.

[7] "Extended documentation of the corrected CICFlowMeter tool, and the regenerated CICIDS 2017 dataset." https://downloads.distrinet-research.be/WTMC2021.

[8] "CSE-CIC-IDS 2018 Dataset," https://www.unb.ca/cic/datasets/ids-2018.html, accessed: 2022-04-30.

[9] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[10] N. Moustafa, "A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets," *Sustainable Cities and Society*, vol. 72, p. 102994, 2021.

[11] "IoT-23 dataset," https://www.stratosphereips.org/datasets-iot23, accessed: 2022-04-30.

Poster: Systematic Elicitation of Common Security Design Flaws

Stef Verreydt
imec-DistriNet
KU Leuven
Heverlee, Belgium
stef.verreydt@kuleuven.be

Koen Yskout
imec-DistriNet
KU Leuven
Heverlee, Belgium
koen.yskout@kuleuven.be

Laurens Sion
imec-DistriNet
KU Leuven
Heverlee, Belgium
laurens.sion@kuleuven.be

Wouter Joosen
imec-DistriNet
KU Leuven
Heverlee, Belgium
wouter.joosen@kuleuven.be

***Abstract*—Threat modeling allows potential security threats to be identified and mitigated at design time. Countermeasures in current threat modeling approaches are mostly modeled as a boolean: either they are implemented, or they are not. This does not allow to take into account potential design flaws for the countermeasure itself. A considerable number of security issues is, however, related to the wrong or incomplete application of common security tactics. For example, the effectiveness of audit logs drops if the data written to the logs is not sanitized. In this paper, we describe our novel approach which aims to systematically and automatically identify common security design flaws.**

***Index Terms*—Threat modeling, CWE, Security-by-design**

1. Introduction

Security and privacy by design principles are becoming increasingly important to develop secure software systems. Indeed, insecure design is one of the most critical software risks according to the OWASP Top 10 2021 [1], and adhering to security and privacy by design principles is even obligated by regulations such as the General Data Protection Regulation (GDPR) [2].

Threat modeling provides a systematic approach to analyze the security and privacy of a software design, thereby allowing potential threats to be identified early on in the development lifecycle. The first step of a threat modeling exercise involves creating a model of the system being analyzed, usually as a Data Flow Diagram (DFD). The DFD notation comprises just five elements, namely *processes*, *data stores*, *external entities*, *data flows* and *trust boundaries*. That model can then be analyzed to identify potential security threats. Tool support for automatic threat elicitation based on machine-readable system models is widespread, and new techniques are being developed rapidly [3]. Common threat elicitation methods used by these tools are based on STRIDE (an acronym for *spoofing*, *tampering*, *information disclosure*, *denial of service* and *elevation of privilege*), but more specific types of threats or attacks such as CAPEC, CWE or CVE entries are also identified by some [4].

The next step of a threat modeling exercise is to mitigate the identified threats by introducing countermeasures. This often involves standard tactics, for example using logging to mitigate repudiation threats [5]. Applying such tactics in a design requires careful consideration of their precise requirements and assumptions. For example, data written to audit logs should be sanitized (CWE-117[1]), and should not contain sensitive information such as credentials (CWE-532[2]). Ideally, design flaws which violate such requirements should be flagged automatically.

One of the underlying issues which prevents this type of analyses is that most threat modeling tools [6] only allow to capture the effect of a countermeasure, and not how countermeasures are included in a design or which elements are involved [7]. For example, a repudiation threat could be marked as mitigated in the Microsoft Threat Modeling Tool [6], but there is no support to explicitly capture the countermeasure which mitigates the threat. To allow tracing back why and how threats are mitigated, Sion et al. [7] extended the DFD notation with a first-class representation for countermeasures. For example, a logging countermeasure can be explicitly added to the system model, allowing tool support to automatically mark certain repudiation threats as mitigated. In our novel approach, we leverage this explicit countermeasure information to automatically identify flaws rooted in the design of the countermeasure itself.

Tuma et al. [8] leverages this notation to automatically identify common security design flaws. Their approach, however, only identified five flaws automatically, and their identification method is mostly based on missing countermeasures rather than design flaws in the countermeasures themselves. For example, one of the flaws identified by their approach is *"insufficient auditing"*, which is identified simply based on the lack of a logging countermeasure. Still their empirical evaluation shows that automatically identifying security design flaws is possible with acceptable precision and recall. In this paper, we therefore describe an approach similar to the one by Tuma et al. [8], but with the ability to automatically identify flaws rooted in the design of countermeasures themselves.

In summary, the goal of our proposal is the following:

Goal. *Automatically eliciting potential security design flaws related to applying standard security tactics during threat modeling.*

In what follows, we provide an overview of our proposed approach, and discuss the advantages compared to traditional threat modeling approaches.

1. https://cwe.mitre.org/data/definitions/117.html
2. https://cwe.mitre.org/data/definitions/532.html

2. Proposal Overview

A high-level overview of our proposal is shown in Fig. 1. The proposed approach was implemented as a extension of an existing threat modeling tool [9] to demonstrate its feasibility. We shortly discuss the main concepts of our proposal and describe the main advantages compared to existing threat modeling approaches.

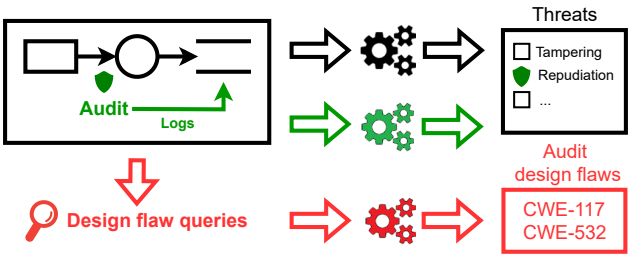


Figure 1. High-level overview of our proposal.
*The default threat modeling flow is shown in black. The extension by Sion et al. [7], which adds first-class representations for countermeasures, is highlighted in **green**. Our proposal further extends this notation by adding queries for common security design flaws, as shown in **red**.*

2.1. Extended system model

The following information should be included in the system model to enable the systematic elicitation of common security design issues:

- a DFD of the system, annotated with data type information;
- a structured and generic description of common security tactics; and
- information on how these tactics are applied in the system being analyzed.

Each of these is shortly discussed in what follows.

2.1.1. System description. Figure 2 shows a DFD for a simple client-server application which will be used as a running example. To enable systematically identifying that, for example, data written to audit logs is not sanitized, information on data types is required, which is not included in the default DFD notation. The data flows are therefore annotated with data type information, similar to the proposal by Tuma et al. [10].

2.1.2. Security tactic description. Applying a traditional threat modeling approach such as STRIDE to the DFD shown in Fig. 2 would return several potential threats, for example a tampering threat on the *Audit Logs* or a repudiation threat on the *Server*. The default DFD notation, however, does not allow to systematically capture that the audit tactic is applied to mitigate the repudiation threat on the *Server*. For that, we first need to define what the audit tactic encompasses. For our proposed approach, common security tactics are described generically, similarly to the proposal by van den Berghe et al. [11]. The advantage of generic tactic descriptions is that they can be applied in different countermeasures and across different system models. A tactic is defined by (i) a name, (ii) the threat(s) which it mitigates, and (iii) the roles which make up the tactic. A role is defined by a name and a type (*proces*, *data store*, *external entity*, *data flow* or *data type*). A generic description of the audit tactic is shown in Fig. 3.

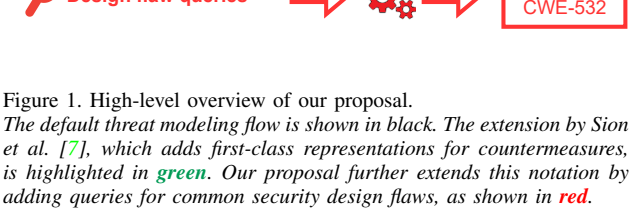


Figure 2. A DFD showing that a *Client* (external entity) can request data from the *Server* (process), which logs the request in one data store and fetches sensitive from another.

```
Name: Audit
Roles:
   - Logged event     : Data Flow
   - Logged data      : Data Type
   - Log database     : Data Store
   - Protected entity: Process
   - Logging process  : Process
Mitigates:
   - Repudiation threats on protected entity
```

Figure 3. Structured description of the audit tactic.

2.1.3. Applying tactics. The system model can then be enriched with information on how tactics are applied using the solution-aware DFD notation proposed by Sion et al. [7]. For our proposal, we define a countermeasure as the application of one or more tactics in a specific model. Concretely, a countermeasure is defined by (i) a name, (ii) the applied tactic(s), and (iii) a set of role bindings which, for each of the roles mentioned in the applied tactics' descriptions, specify the DFD element fulfilling that role. Figure 4 describes a countermeasure which applies the audit tactic to the DFD shown in Fig. 2.

```
Name:  Client request logging
Applied tactics: Audit
Role bindings:
   - Logged event     : df1
   - Logged data      : request data
   - Log database     : Audit Logs
   - Protected entity: Server
   - Logging process  : Server
```

Figure 4. A countermeasure describing how the the audit tactic (Fig. 3) is applied to Fig. 2.

Based on this information, tool support [9] can automatically mark *Repudiation* threats on the *Server* as mitigated, as the *Server* fulfills the role of *Protected entity*, which, as described in the audit tactic (Fig. 3), is protected against *Repudiation* threats. Our approach build on this by allowing tool support to also identify design flaws in the countermeasure itself, as will be explained next.

2.2. Security design flaw queries

Based on the generic and structured tactic descriptions (Section 2.1.2), queries can be composed for common design flaws related to the tactics. Similar to the tactic descriptions, queries are also described generically so that they can be applied to any system model. A common flaw

for the audit tactic is, for example, that the data written to the logs is not sanitized, as described by CWE-117.[3] Thus, if a countermeasure applies the audit tactic, and the data written to the logs is (or contains) the exact data contained in the logged flow, then CWE-117 should be elicited for that countermeasure. The query for this flaw is shown in Fig. 5.

```
query cwe-117{Solution S} {
  S applies tactic named 'Audit';
  S binds the 'Logged event' role to some data flow DF;
  DF is annotated with a data type X;
  S binds the 'Logged data' role to data type Y;
  Y == X, or Y includes X;
}
```

Figure 5. Pseudo code for the CWE-117 elicitation pattern.

2.3. Advantages

As described earlier, applying STRIDE to the DFD shown in Fig. 2 would result in several threats, for example a tampering threat on the *Audit Logs* or a repudiation threat on the *Server*. As the system model includes audit logs, the repudiation threat can be marked as mitigated. Traditional threat modeling approaches, however, provide no further guidance on how to systematically evaluate whether the logging solution is designed correctly. Sufficient security knowledge and manual effort is required to identify that (a) a tampering threat on the audit logs reduces the effectiveness of the logging solution, and (b) encryption is not sufficient to mitigate the tampering threat, even though it is a standard tactic for integrity. Indeed, the tampering threat may have been marked as mitigated if an encryption tactic was applied, but this does not prevent more complex issues such as CWE-117 (which can be categorized as a tampering threat).

In comparison, our proposal allows such issues to be identified systematically and automatically. Furthermore, tactic descriptions and flaw queries can be reused across models, thus limiting the security expertise required to apply our approach. Finally, the issues identified by our approach are tailored to the context, which is not the case for traditional approaches. For example, whereas a STRIDE analysis would simply flag an unmitigated tampering threat on the audit logs, our proposal enables automatically generating more detailed issue descriptions such as *"Request data provided by the client is directly written to the audit logs, which may allow attackers to forge log entries. As a result, the client request logging solution may not suffice to prevent repudiation threats. See* cwe-117 *for more information."* This allows to clearly trace the specific cause of the flaw, as well as its impact.

3. Discussion and Future Work

In summary, by extending a system model with information on how tactics are applied in a design, our approach allows common security design flaws to be identified systematically and automatically. Compared to traditional threat modeling approaches, where countermeasures are mostly modeled as booleans (present/absent), this reduces the time and effort needed to find common security flaws. Furthermore, the required security expertise is also reduced, as tactic descriptions and flaw queries are generic and reusable across system models.

To demonstrate the feasibility of our approach, a tool prototype was developed, as well as a number of tactic descriptions and flaw queries for common security design issues. Applying these queries to example models such as the running example did not reveal any issues. In future work, we aim to compose a more extensive catalog of tactics and common flaw queries based on existing knowledge. The Architectural Concepts view of the CWE[4] can serve as a starting point, as it contains a detailed collection of potential design flaws, organized by architectural security tactics to which they apply [12]. Furthermore, an evaluation is needed based on a concrete and realistic case.

References

[1] The OWASP Foundation, "OWASP Top 10 - 2021," https://owasp.org/Top10/, 2021.

[2] European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016," *Official Journal of the European Union*, vol. 59, no. L 119, pp. 1–88, May 2016.

[3] Z. Shi, K. Graffi, D. Starobinski, and N. Matyunin, "Threat modeling tools: A taxonomy," *IEEE Security & Privacy*, no. 01, pp. 2–13, dec 2021.

[4] B. J. Berger, K. Sohr, and R. Koschke, "Automatically Extracting Threats from Extended Data Flow Diagrams," in *Engineering Secure Software and Systems*, ser. Lecture Notes in Computer Science, J. Caballero, E. Bodden, and E. Athanasopoulos, Eds. Springer International Publishing, 2016, pp. 56–71.

[5] A. Shostack, *Threat Modeling: Designing for Security*, 1st ed., 2014.

[6] Microsoft Corporation. (2020) Microsoft threat modeling tool 7. [Online]. Available: https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool

[7] L. Sion, K. Yskout, D. Van Landuyt, and W. Joosen, "Solution-aware data flow diagrams for security threat modeling," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1425–1432. [Online]. Available: https://doi-org.kuleuven.e-bronnen.be/10.1145/3167132.3167285

[8] K. Tuma, L. Sion, R. Scandariato, and K. Yskout, "Automating the early detection of security design flaws," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 332–342. [Online]. Available: https://doi.org/10.1145/3365438.3410954

[9] L. Sion, D. Van Landuyt, K. Yskout, and W. Joosen, "Sparta: Security & privacy architecture through risk-driven threat assessment," in *International Conference on Software Architecture*. IEEE, 8 2018, pp. 89–92. [Online]. Available: https://lirias.kuleuven.be/1656829

[10] K. Tuma, R. Scandariato, M. Widman, and C. Sandberg, "Towards security threats that matter," in *Computer Security*. Springer International Publishing, 2018, pp. 47–62.

[11] A. van den Berghe, K. Yskout, and W. Joosen, "A reimagined catalogue of software security patterns," in *The 3rd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS'22)*. ACM, 2022.

[12] J. C. S. Santos, K. Tarrit, and M. Mirakhorli, "A catalog of security architecture weaknesses," in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, 2017, pp. 220–223.

3. https://cwe.mitre.org/data/definitions/117.html

4. https://cwe.mitre.org/data/definitions/1008.html

Poster: Testability-driven security and privacy testing for Web Applications

Luca Compagna
SAP Security Research

Giancarlo Pellegrino
CISPA Helmholtz Center for Information Security

Davide Balzarotti
Eurecom

Martin Johns
Technical University Braunschweig

Angel Cuevas
Universidad Carlos III de Madrid

Battista Biggio
Pluribus One

Leyla Bilge
Norton Lifelock

Fabian Yamaguchi
ShiftLeft Inc

Matteo Meucci
IMQ Minded Security

***Abstract*—Modern web applications play a pivotal role in our digital society. Motivated by the many security vulnerabilities and data breaches routinely reported on those applications, we initiated the EU TESTABLE research project to address the main challenges of building and maintaining web applications secure and privacy-friendly. The ultimate goal is to lay the foundations for a new integration of security and privacy into the software development lifecycle (SDLC), by proposing the novel idea of combining two metrics to quantify the security and privacy risks of a program, i.e., the testability of the codebase (via the novel concept of "testability patterns") and the indicators for vulnerable behaviors.**

We have already achieved promising results by applying our research proposal in the area of static analysis security testing (SAST) [2]. Hundreds of tarpits—code instructions challenging for SAST tools—have been identified and captured in testability patterns that we used to measure SAST tools effectiveness as well as to improve the "SAST testability" of open source applications via code refactoring routines removing the tarpits. More than 180 new vulnerabilities have been uncovered after refactoring and confirmed by open source applications' owners. We believe similar promising results can also be achieved in other areas such as dynamic analysis, privacy, and machine learning.

***Index Terms*—web, testability, risk, measurement, metrics, security, privacy, ML**

1. Introduction

Web applications are ubiquitous, and they are used in a multitude of different domains. According to the 2020 Edgescan Security Report, "Web application security is where the majority of risk still resides" [1]. This is confirmed by the fact that most of the recent data breaches took advantage of the poor security of web applications.

Software security testing plays a fundamental role to mitigate this problem. In particular, developers regularly use static code analysis and automated testing tools to verify their application and identify vulnerabilities. But existing solutions are often limited in their ability to automatically discover security problems. When problems are not detected is always challenging to know for sure what the testing tools left uncovered.

While threat modeling and risk assessment methodologies are often adopted by industry at early phases of the software development lifecycle, they are too far from the implementation details and therefore they can only set high-level recommendations for later phases, such as testing. Computing risk measures that are closer to the application code itself can be used to complement early risk assessment phases, and could help to prioritize the testing effort. However, this area has so far mainly focused on determining the likelihood that an application contains a vulnerability (what we call **vulnerability indicators**). Therefore, even when these indicators exist, they provide little guidance to the developers on how the corresponding risk could be mitigated. For instance, knowing that a web application has a high risk of containing vulnerabilities (even when none have been detected during development and testing) is a piece of information that can be difficult to translate into actionable insights.

The poor state of web security is further exacerbated by two rapidly emerging aspects. First, more and more **web applications integrate machine learning (ML) components** at the code or at the service level to implement business functions or to protect web applications from abuse[1]. Unfortunately, existing techniques to detect vulnerabilities in web application lack the sophistication to interact with and interpret the behavior of ML components, thus impeding the analysis and testing of a larger and larger number of ML-based web applications. Second, web applications are facing more and more a social and political pressure to be designed, implemented, and deployed in a way that **preserves the users privacy**. This requirement is especially relevant in Europe after the adoption of the General Data Protection Regulation (GDPR) that aims, among other things, to establish a framework to guide organizations to manage citizens'—and by extension web applications users'—personal data. GDPR requires developers to extend existing risk-based and secure development methodologies with privacy-preserving solutions when designing and implementing the application software. It also introduces specific challenges to existing testing methodologies, as the privacy of an application

1. E.g., Google Cloud's Vision API https://cloud.google.com/vision or Microsoft Cognitive Services https://azure.microsoft.com/en-us/services/cognitive-services/

depends on the composition of a large number of third-party services and libraries, which are often not under developers' control. Moreover, while it is reasonable to believe that developers have a clear incentive in fixing security vulnerabilities, it might be counterproductive for them to collect *less* user data. These two aspects combined result in the fact that privacy testing cannot be performed solely on the developers' side, but it also requires constant monitoring from the end-user perspective. To summarize, we can classify existing shortcomings around three main challenges:

(C1). Security testing of web-based applications is a task of ever growing complexity, which cannot sufficiently be addressed with the current set of technologies. The rapid inclusion of ML classifiers as part of (or as backend of) web applications further reduces the effectiveness of traditional white and black-box testing solutions.

(C2). Existing approaches do not provide a clear feedback to the developers on how to interpret the (lack of) results. In particular for areas of the application that are hard to test or even out of reach for the utilized testing tool, no information is given on the encountered problems or about potential remedies to provide more robust testing.

(C3). Existing static and dynamic frameworks to test web applications focus on the identification of well-known *security* vulnerabilities. *Privacy issues* are regularly overlooked and often left to a third party analyst to discover through tedious manual processes.

While there is a lot of research investigating over (C1), we believe that (C3) and (C2) tend to be overlooked and more research should focus on that. In our EU funded project TESTABLE [3] we are researching over all those challenges. Our idea is to re-design the way we test web applications around a new **testability metric**.

By applying this idea in the context of SAST we obtained very promising results [2]. In there, hundreds of tarpits—code instructions challenging for SAST tools—have been identified and captured in **testability patterns** that enabled us to measure SAST tools effectiveness, to make developers aware of the tarpits in their code (via automated discovery rules for the tarpits), and to improve the "SAST testability" of open source web applications (via code refactoring routines removing the tarpits). Hundreds of new vulnerabilities have been uncovered after refactoring and confirmed by applications' owners.

An introduction to our approach, instantiated for the SAST domain, is discussed in Section 2. However, the approach is more general and future directions, shortly presented in Section 3, include its application to dynamic application security testing (DAST) and beyond security so to consider also the privacy and (adversarial) machine learning dimensions.

2. Approach and results for SAST

Fig. 1 depicts our approach. The core idea is to introduce the *(un-)testability dimension* in the risk score. Existing methodologies for measuring risk focus on the probability V that a certain web application would contain a vulnerability. These *mono-dimensional* methodologies are based on *vulnerability indicators* such as the size and complexity of the code, the type of information handled by

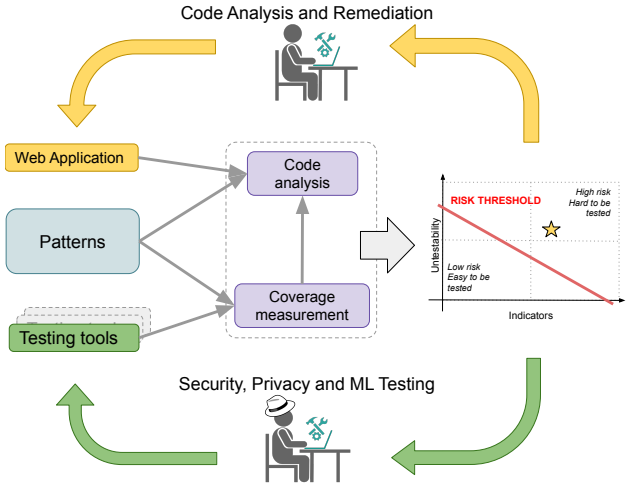


Figure 1: Our approach

the application, the number of commits over time, or the total number of developers—all elements that are difficult to modify in a running project. We propose instead a new orthogonal dimension measuring the **testability** of the application into the current notion of risk (cf. two-dimensional graph in the right-hand side of Fig. 1).

As a simple example, we can imagine two applications, A1 with a vulnerability indicator V=40% and A2 with V=63%. While lowering these values can improve the security and privacy of the applications, it can be very difficult to do that in practice. For instance, reducing code size or changing the number of developers are difficult aspects to act on. Moreover, it is possible that A1 (V=63%) avoids certain coding and practices (we refer to them as **testability patterns** in our work), thus making the application easier to analyze by testing tools. As a result, even if the likelihood of containing a vulnerability is higher for A1, the developers can expect that most of these problems will be detected during the testing phase—thus resulting in a final product that is more secure than the one corresponding to A2 (V=40%).

For space limitation, we illustrate in the next subsections the three main activities of our approach when applied to SAST, providing a summary of the results published in [2]. However the approach is more general and can be applied beyond SAST (see Section 3).

2.1. Testability patterns creation

Core to our research idea is the identification of the crucial testability patterns for specific testing approaches of web applications.

In the context of SAST, we created hundreds of testability patterns for both PHP and JavaScript (JS), two of the most popular web programming languages. A SAST testability pattern captures a few code instructions in language X that, when present, may impede the ability of state-of-the-art tools to analyze an application developed in X. Those few code instructions are referred to as the "tarpit". For instance, let us consider the following pattern instance for PHP:

```
1 function F($var) {
2    return $var;
```

```
3 }
4 $a = $_GET["p1"]; // source
5 $b = call_user_func("F", $a); // tarpit
6 echo $b; // sink
```

First, note that all instances include, in purpose, a simple cross-site scripting (XSS) vulnerability to set an expected result when measuring SAST tools: in line 4 the attacker controls the input parameter `"p1"` (the source) that is printed, without any sanitization, into the HTML web page in line 6 (the sink). Second, in between this source-sink data-flow there is the *tarpit*. The tarpit is the code area that may confuse the SAST tools. The tarpit in line 5 is a form of *dynamic dispatching* (reflection) that allows the programmer to invoke a function specified by passing its name inside a string.

This specific pattern instance, that we will refer to as *simple dispatching*, is hardcoding a constant parameter `"F"`, thus making the target function resolvable from a static analysis perspective. Other similar pattern instances could be created. For instance, that constant could be first assigned to a variable and that variable could be then used as first parameter of `call_user_func`. Similarly, a concatenation instruction could be used as first parameter of the dynamic dispatching making the tarpit even more challenging and so on and so forth.

To be comprehensive, the internal specifications, and the APIs of both PHP and JS were reviewed and distilled into many SAST testability patterns, around several categories (e.g., object-oriented programming, security, etc): 120 instances were created for PHP and 150 for JS. We made all these patterns available to the entire web and SAST community [2].

2.2. Measurement and advancement of security, privacy and ML testing

Based on our patterns a comprehensive measurement of current testing approaches will be conducted to identify strenghts and weaknesses. Advanced techniques can be then designed and implemented to overcome some of these weaknesses (cf. lower green loop in Figure 1).

For SAST, we considered so far only the measurement part. An arsenal of 11 commercial and open-source SAST tools (6 for PHP and 5 for JS) were selected and measured against the created pattern instances. The best commercial tools were only able to handle 50% of the PHP and 60% of the JS tarpits, thus potentially leaving large parts of an application code unexplored. For instance, only 2 tools over 6 were able to detect the XSS in the simple dispatching pattern instance outlined above. The measurements, in [2] are detailed over the pattern categories showing that certain SAST tools may exceed in a category and fall short toward others. All these are precious information for SAST tools' owners and for the research community trying to advance SAST. You can know very precisely which pattern instances are blocking your tool and invest enginnering effort to support them in a future release.

2.3. Web Application Analysis and Remediation

For our risk measurements and iterative testing process, it is essential to reliably identify testability patterns in the source code of a web application under test. Thus, analysis techniques (mainly at the code level, but not only) will be designed to detect the testability patterns. Furthermore, following up on the results of our web application analysis, we will investigate remediation techniques such as code debloating and refactoring methods to remove problematic code patterns or replacing them with alternatives that are better handled by testing tools (cf. upper yellow loop in Figure 1).

In the context of SAST, to measure the impact on the unsupported tarpits, automated *discovery rules* were implemented and run to analyse more than 3000 open-source applications (the Testbed, in short). The experiments demonstrate that these tarpits are very common in the real world: the average project contains 21 different tarpits and even the best SAST tool cannot process more than 20 consecutive instructions without encountering a tarpit that prevents it from correctly analyzing the code. For instance, the dispatching pattern instance was discovered in more than 500 projects in the Testbed. The ability to automatically discover each tarpit brings many benefits. First, it can provide immediate and precise feedback to the developers about the tarpits in their code (e.g., by integrating the discovery rules into an IDE plugin). This information can then be used to make an informed decision about which combination of SAST tools are better suited to analyze the code, which parts of the application are *blind spots* for a static analyzer and thus may require a more extensive code review process, and which region of code could be *refactored* into *more testable* alternatives.

A few experiments were executed to evaluate the power of code refactoring as a mean to make an application more testable for SAST tools. By running SAST tools both before and after the transformations, a significant improvement in the overall testability of the application was observed. More than 400 new true positives emerged upon transformations and 188 have been already confirmed by the respective team, 55 of which affected very popular projects with more than 1000 stars in Github.

3. Conclusion and future directions

The outcomes from the previous section confirm the added-value of the overall approach of Figure 1, when applied to SAST, not only in measuring testing tools, but also and foremost in the impact of removing testability patterns' tarpits as a means to increase the testability of web applications.

Besides maturing the work for SAST (e.g., by computing also vulnerability indicators and combining that with testability to derive an overall risk score), we aim to apply our approach to DAST, privacy and machine learning. The core idea is to create testability patterns for all these areas so to measure the *available* state-of-the-art testing tools, advance/develop the techniques underlying these tools, and mitigate these patterns whenever possible to increase the testability of the application. For instance, we expect many testability patterns to emerge as a means to probe DAST crawlers. A failure in the crawling phase may leave a large portion of the web application uncovered. On the other hand, aware that privacy testing, when compared to security testing, is still in its infancy, our research in that area will focus on defining the scope of privacy

testing and implement privacy testing techniques to address relevant and well-known web related privacy issues. Last, but not least, we want to create testability patterns which are hindering security and privacy testing of ML-based components so to measure these patterns against the many emerging tools to detect adversarial machine learning attacks.

References

[1] Edgescan. 2020 vulnerability statistics report. 2020.

[2] Feras Al Kassar, Giulia Clerici, Luca Compagna, Fabian Yamaguchi, and Davide Balzarotti. Testability Tarpits: the Impact of Code Patterns on the Security Testing of Web Applications. In *Network and Distributed System Security (NDSS) Symposium*, NDSS 22, April 2022.

[3] TESTABLE consortium. TESTABLE: TestabiliTy Pattern-driven Web Application Security and Privacy Testing. https://testable.eu/, Accessed August 31, 2022.

Poster: The Beauty and the Beast (40 years of process algebra and cybersecurity)

1st Silvia De Francisci
SySMA Unit
IMT School for Advanced Studies
Lucca, Italy
silvia.defrancisci@imtlucca.it

2nd Gabriele Costa
SySMA Unit
IMT School for Advanced Studies
Lucca, Italy
gabriele.costa@imtlucca.it

3rd Rocco De Nicola
SySMA Unit
IMT School for Advanced Studies
Lucca, Italy
rocco.denicola@imtlucca.it

***Abstract*—Process algebras provide the mathematical foundation for several formal verification techniques, and they profoundly influenced many fields, from correct design to testing. Process algebras were greatly influential also for the security community. One of the main reasons for their success is their compact, yet expressive and flexible syntax, which allows to model the relevant aspects of computation while abstracting away the secondary ones. Although most authors acknowledge the importance of process algebras for the security community, it is not trivial to estimate how they shaped the past and present researches.**

The goal of this work is to provide a comprehensive outlook on some prominent works about process algebras and security. These include both the application of process algebras to security problems and process algebras inspired by security-related aspects of computation. To achieve this, we consider three fundamental fields of cybersecurity, i.e., *secure development*, *threat modeling* and *vulnerability assessment*.

***Index Terms*—process algebra, cybersecurity, formal methods in security.**

1. Introduction

Process Algebras (PA) are formal languages commonly used to model the behavior of a computational agent. Roughly speaking, PA put most of the emphasis on the control flow structure of the modeled agent, rather than on its data flow. In most PA, computational steps also emit *observable* actions. This allows to elegantly introduce the role of an external observer, i.e., someone who aims at understanding an agent's behavior, but has no control on its internal structure. Under reasonable assumptions, an attacker is, in fact, an external observer. As a matter of fact, an attacker is generally represented by its *capabilities* and *goals*. In terms of capabilities, the possibility to partially interact with the agent's I/O mechanism is a typical setting (e.g., think of information flow [19]). In terms of goals, the attacker's aim can be modeled as a target state, e.g., denoting a failure of the agent, she wants to reach. Formal models of both the attacker and the system are the fundamental building blocks of most automated, formal reasoning techniques. Not surprisingly, a vast literature about the adoption of PA in cybersecurity exists. As a result, determining the overall influence of PA in cybersecurity and its sub-fields is difficult.

This poster proposes a systematization of the existing PA and their applications to the cybersecurity fields. Our contributions are the following.[1]

- We show significant PA from a genealogy standpoint, as well as their unique characteristics.
- We discuss the impact of some PA and its genealogy in three application scenarios for cybersecurity classification: Secure Development, Threat modeling, and Vulnerability assessment.
- We show the state-of-the-art of PA for secure Development Life Cycle (DLC).

2. Genealogy of PA

The genesis of PA goes back to the 80s, when a few authors independently proposed different calculi for the specification of processes: Milner's Calculus of Communicating Systems (CCS) [16], Brookes, Hoare and Roscoe's Communicating Seqential Processes (CSP) [8], and Bergstra's Algebra of Communicating Processes (ACP) [6], which inspired a considerable number of other PA. Typically, derived PA introduce some new elements w.r.t. their archetypes. These elements often aim to model some specific aspects of computation. In order to better highlight these extensions we label PA with icons denoting their peculiar features. Such features include the ability to model security aspects (🔒), timed computation (🕒), stochastic behaviors (🎲), quantum computing (⚛), imperative statements (❗), cyber-physical systems 🎛 and wireless networks 📶. Arguably, the main reason behind this diversification is the needing of considering specific aspects of the computation of distributed agents. In this landscape, security is no exception and several security-related PA (🔒) emerged. However, the main difference w.r.t. other domain-specific PA is that security does not refer to some peculiar aspect of the computation. In general, security has to do with all the things that might go wrong during the execution of a process. These behaviors of interest can be modeled with specific, secure PA or even with general purpose ones. As a consequence, PA used in this field require particular attention for understanding which security concerns they deal with, as explained below.

1. Tables and figures highlighting our results are omitted in this abstract and will be included in the poster above.

3. Secure PA

We now focus on PA that have been specifically proposed for tackling security aspects. Security is a multifaceted issue. It is common knowledge that there is no "silver bullet" for security and that several security tasks must be carried out to improve the robustness of a system, e.g., as in Security-by-Design [11].

Here we put forward a classification based on the following three areas.

- ⚙ Secure development, i.e., PA supporting the secure design, implementation and execution of a system.
- 🕵 Threat modeling, i.e., PA used for modeling and analyzing the behavior of an attacker and her strategies.
- 🐞 Vulnerability assessment, i.e., methods employing PA for spotting out actual flaws in existing systems.

Needless to say, precisely measuring the impact of a certain PA in these areas is extremely hard or even impossible. However, a rough estimation can be obtained by considering the scientific literature. In particular, we propose the following scale that, to the best of our knowledge, we can measure for a any given PA and security area.

- 0 No literature exists applying PA in the area.
- $\frac{1}{2}$ Some papers exist, but their authors belong to a single clique.[2]
- 1 Some papers exist and their authors belong to two or more cliques.

Below we discuss some observations we consider more interesting.

Secure Development According to the considered literature, most PA-based proposals focus on secure design and development. According to our analysis, for each considered PA, ⚙ ¿ 0. However, when only considering security PA (🔒), the trend changes significantly.

Secure PA Secure PA are usually employed for modeling threats and identifying vulnerabilities. Interestingly, SPA with its extensions and especially applied π-calculus with its extensions are used for the secure development of systems.

4. PA for Secure Development

A cornerstone of Security-by-Design is that security should be considered from the very early stages of the design process. In this respect, thanks to their abstract and compact syntax, PA have often been proposed as a design formalism. Also, their formal semantics permits to carry out verification procedures which are not natively supported by other design languages, e.g., UML [7] and BPMN [26]. Often, formal verification occurs via model checking [10].

We reports the adoption of PA in DLC w.r.t. some major application domains, specifying whether there exists at least one implementation among PA-based tools.

We consider the following DLC macro-phases:

- *Planning*, i.e., the initial conceptualization of the system and its requirements.
- *Design*, i.e., the architectural modeling of the system and its components.
- *Implementation*, i.e., the actual development of the system.
- *Testing*, i.e., for validating the implementation against the expected requirements.
- *Maintenance*, i.e., for monitoring, updating and eventually disposing the system.

To the best of our knowledge, we conclude the following use of PA related to DLC phases:

No usage for planning. No author proposes PA for the earliest stage of the development process. Reasonably, this happens because during this phase there is no information about the modules that will constitute the system to be implemented.

Design, implementation and testing. Many authors propose approaches employing PA during design, implementation and testing. This is somehow expected since PA are particularly suitable for modeling a system and its components. Moreover, their formal semantics provide the foundation for refinement methods, useful at implementation time for driving the development process from its initial specification. Also, at testing time, model checkers' counterexamples can be converted to test cases.

Almost no maintenance. Not surprisingly, PA are scarcely used for maintenance, with a few, interesting exceptions mostly related to runtime enforcement. In particular, this topic is recurrent in the field of policy specification. The main reason is that PA provide a theoretical background for policy enforcement. As a matter of fact, some authors [1], [4], [5], [15], [14] found it convenient to model policy monitors as agents that run in parallel with a target system. In this context, action authorization amounts to synchronous transitions between the two agents, i.e., the monitor and its target.

Types of PA When only the Design phase is studied, the PA employed are frequently new, generated for the article's purpose. While, when the Design, Implementation, and Testing phases (D-I-T) are considered together, the PA used are typically those associated with a tool. An ad hoc PA could better model the architecture under consideration, but at the same time, it could be challenging to verify the model's correctness. As a result, sometimes authors translate a PA or a programming language into another PA to take advantage of a model checker. For example, on [20], CHP is translated into LOTOS in order to allow the application of CADP; Ferrara [13] also shows a translation from BPEL into LOTOS for the same reason.

5. Related Work

Some authors revised the history of PA and the relevant application domains. For instance, Baeten [3] surveys PA in a general, he summarizes the history of CCS, CSP, and ACP and he presents the developments of time and stochastic features. More recently, Brookes and Roscoe [9] approach from a historical point of view to CSP and in particular to the FDR tool. Wang presents ATCP and its variants in [25] and [24], considering cryptographic properties, abstraction and introducing guards. In the first

2. Cliques are computed by considering the co-authoring relation induced by the literature considered in this poster.

article, he uses ATCP to analyze several protocols, while in the second one, ATCP is used to model Map-Reduce, Google File System, cloud resource management, Web Service Composition, and QoS-aware Web Service orchestration. Aldini et al. [2] survey the application of PA to software architecture, emphasizing on component-oriented modeling. Beek et al. [21] compare formal methods used on web service composition considering three features: connectivity, correctness, and quality of services. In the same domain Eddine [12] compares the different approaches, among which are PA, to design and implement Web services focusing on choreography and orchestration. Tuan Anh et al. [22] look into the issues surrounding the security and privacy of the Internet of Mobile Things utilizing PA, with a focus on the mobile PA π-calculus. Wan et al. [23] survey composition mechanisms and then models for cyber-physical systems, concluding that "CPS development must be supported from the design phase by process algebras to achieve strong results on correctness,performance, cost and efficiency." Related to protocol specification and verification, Ryan et al. [18] model and analyze protocols and properties through CSP, using FDR and Casper tools. Ryan and Smyth [17], present the applied pi-calculus, in which areas it was used, and how to use it to model protocols and properties. Wideł et al. [27] investigate the different generation and analysis approaches for Attack Trees; among the formal methods studied are PA, while among the analysis methods is the tool UPPAAL and its variants.

References

[1] Kamel Adi, Lamia Hamza, and Liviu Pene. Automatic security policy enforcement in computer systems. *computers & security*, 73:156–171, 2018.

[2] Alessandro Aldini, Marco Bernardo, and Flavio Corradini. *A process algebraic approach to software architecture design*. Springer Science & Business Media, 2010.

[3] Jos CM Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2-3):131–146, 2005.

[4] David Basin, Samuel J Burri, and Günter Karjoth. Dynamic enforcement of abstract separation of duty constraints. *ACM Transactions on Information and System Security (TISSEC)*, 15(3):1–30, 2012.

[5] David Basin, Samuel J Burri, and Günter Karjoth. Obstruction-free authorization enforcement: Aligning security and business objectives. *Journal of Computer Security*, 22(5):661–698, 2014.

[6] Jan A Bergstra and Jan Willem Klop. Process algebra for synchronous communication. *Information and control*, 60(1-3):109–137, 1984.

[7] Grady Booch. *The unified modeling language user guide*. Pearson Education India, 2005.

[8] Stephen D Brookes, Charles AR Hoare, and Andrew W Roscoe. A theory of communicating sequential processes. *Journal of the ACM (JACM)*, 31(3):560–599, 1984.

[9] Stephen D Brookes and AW Roscoe. Csp: a practical process algebra. In *Theories of Programming: The Life and Works of Tony Hoare*, pages 187–222. 2021.

[10] Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model checking*. MIT press, 2018.

[11] Daniel Deogun, Dan Johnsson, and Daniel Sawano. *Secure by Design*. Manning Publications, 2019.

[12] Meftah Mohammed Charaf Eddine et al. A comparative study of formal approaches for web service oriented architecture. *Netw. Commun. Technol.*, 5(2):15–33, 2020.

[13] Andrea Ferrara. Web services: a process algebra approach. In *Proceedings of the 2nd international conference on Service oriented computing*, pages 242–251, 2004.

[14] Mahjoub Langar, Mohamed Mejri, and Kamel Adi. Formal enforcement of security policies on concurrent systems. *Journal of Symbolic Computation*, 46(9):997–1016, 2011.

[15] Fabio Martinelli, Ilaria Matteucci, and Charles Morisset. From qualitative to quantitative enforcement of security policy. In *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, pages 22–35. Springer, 2012.

[16] Robin Milner et al. A calculus of communicating systems. *Springer Verlag*, 1980.

[17] Mark D Ryan and Ben Smyth. Applied pi calculus. In *Formal Models and Techniques for Analyzing Security Protocols*, pages 112–142. Ios Press, 2011.

[18] Peter Ryan, Steve A Schneider, Michael Goldsmith, Gavin Lowe, and Bill Roscoe. *The modelling and analysis of security protocols: the CSP approach*. Addison-Wesley Professional, 2001.

[19] Andrei Sabelfeld and Andrew C Myers. Language-based information-flow security. *IEEE Journal on selected areas in communications*, 21(1):5–19, 2003.

[20] Gwen Salaun, Wendelin Serwe, Yvain Thonnart, and Pascal Vivet. Formal verification of chp specifications with cadp illustration on an asynchronous network-on-chip. In *13th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'07)*, pages 73–82. IEEE, 2007.

[21] Maurice H Ter Beek, Antonio Bucchiarone, and Stefania Gnesi. Formal methods for service composition. *Annals of Mathematics, Computing & Teleinformatics*, 1(5):1–10, 2007.

[22] Vu Tuan Anh, Pham Quoc Cuong, and Phan Cong Vinh. Context-aware mobility based on π-calculus in internet of thing: A survey. In *Context-Aware Systems and Applications, and Nature of Computation and Communication*, pages 38–46. Springer, 2019.

[23] Kaiyu Wan, Danny Hughes, Ka Lok Man, and Tomas Krilavičius. Composition challenges and approaches for cyber physical systems. In *2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications*, pages 1–7. IEEE, 2010.

[24] Yong Wang. Actors–a process algebra based approach. *arXiv preprint arXiv:2104.05438*, 2021.

[25] Yong Wang. Secure process algebra. *arXiv preprint arXiv:2101.05140*, 2021.

[26] Stephen A White. Introduction to bpmn. *BPTrends*, 2004.

[27] Wojciech Wideł, Maxime Audinot, Barbara Fila, and Sophie Pinchinat. Beyond 2014: Formal methods for attack tree–based security modeling. *ACM Computing Surveys (CSUR)*, 52(4):1–36, 2019.

Poster: The impact of data sampling in the anonymization pipeline

Jenno Verdonck, Kevin De Boeck, Michiel Willocx, Jorn Lapon, Vincent Naessens
imec-DistriNet
KU Leuven
Ghent, Belgium
firsname.lastname@kuleuven.be

***Abstract*—An increasing number of companies are selling data as an additional source of revenue, or acquire data from other parties to optimize their business. In many cases, the shared data contains sensitive personal records. According to the GDPR regulation, personal data should be anonymized before it is released to third parties. A frequently applied technique is the k-anonymity metric, which ensures that every record in the dataset becomes indistinguishable from K other records through data generalization. This work combines generalization techniques with sampling. By adding a sampling step in the anonymization pipeline, additional uncertainty is introduced towards a potential attacker. As attackers can no longer be sure that an individual is in the sampled dataset, the re-identification risk is mitigated. This work proposes and evaluates multiple sampling techniques. Both the privacy and the utility properties of the anonymized datasets are embraced. The utility of the anonymized datasets is further evaluated in a machine learning use-case.**

***Index Terms*—anonymization, sampling, privacy, utility**

1. Introduction

Data collection and processing have become aspects of increasing importance in the daily operation of many businesses and organizations. Amongst others, data is employed for optimizing production processes and to increase the effectiveness of marketing campaigns. Hence, sharing (or trading) data can be very lucrative, and might even bootstrap cooperation between organizations. While data sharing exposes great opportunities for companies, caution should be taken. First, the European GDPR regulation states that datasets containing personal information should be anonymized before they are released. This means that a record in the shared dataset can no longer be linked to an individual afterwards. Second, thoughtless release can undermine the competitiveness of companies. Anonymization techniques can be applied to mitigate these threats.

An often cited and applied strategy for data anonymization relies on privacy metrics such as k-anonymity [1]. The goal of this metric is to generalize attribute values in such a way that each individual becomes indistinguishable from at least $k-1$ other individuals in the dataset. However, constructing a feasible anonymized dataset is no sinecure for most organizations. Some information is inevitably lost during anonymization caused by generalization. Hence, companies often struggle to find a satisfactory balance between the privacy and the utility of an anonymized dataset. Moreover, solely applying k-anonymity does not always protect against attackers with membership knowledge, a key assumption in the prosecutor attacker model [2]. This work argues that the aforementioned risk is mitigated by applying an additional sampling step in the anonymization pipeline. An attacker can no longer be sure that the target is in the dataset, making it harder to re-identify individuals. The prosecutor becomes a journalist, corresponding to the also well-known but less powerful journalist attacker model.

Contributions. This paper presents preliminary results of our research on combining traditional anonymization techniques (i.e. k-anonymity) and sampling. Three different sampling strategies are presented, implemented and assessed. The impact of each strategy on both the utility and the privacy properties of the remaining data is evaluated in a practical machine-learning use-case.

The remainder of this work is structured as follows. Section 2 points to related work. Section 3 describes different sampling strategies. Thereafter, Section 4 details the evaluation methodology. Lastly, Section 5 outlines the conclusions and points to future work.

2. Related work

Sampling has been the subject of previous research. Rocher et al. [3] demonstrate that solely applying random sampling does not sufficiently protect the privacy of individuals in the dataset. Hence, this work combines sampling with techniques for k-anonymity. Other research shows that random sampling in combination with k-anonymity can achieve differential privacy [4], [5]. Their main focus is redefining differential privacy in this context. Our work is complementary as it considers multiple sampling strategies and focuses on the utility-privacy balance. Still others do focus on the privacy-utility balance. [6] and [7] assess this balance in an optimization and an association rule mining case respectively. They do however solely focus on anonymization metrics and do not embrace sampling. Previous work [8] combined k-anonymity and sampling, and focused on the privacy-utility balance in an optimization use-case. This paper expands on this work by assessing this balance in a machine-learning setting, and by applying multiple sampling strategies. When training machine learning models on anonymized data, additional steps are required in order to apply the machine

learning models with non-anonymized data. Inan et al [9] elaborate on this challenge.

3. Sampling strategies

This work combines sampling with attribute generalization (i.e. k-anonymity). In the anonymization pipeline, the sampling step can be executed either before (*pre-sampling*) or after (*post-sampling*) the generalization step. Both approaches have advantages. *Pre-sampling* reduces the size of the dataset, and hence the complexity of the generalization step. *Post-sampling* enables more intelligent sampling based on the output of the generalization algorithm. Note that – especially in larger datasets – both strategies can be combined.

This work evaluates and compares three sampling strategies, namely *random sampling*, *stratified sampling* and *balanced stratified sampling*. Note that the latter two are only applied in *post-sampling*, as they require the output of the generalization algorithms (i.e. the equivalence classes of size $\geq k$):

- **Random sampling.** The main advantage of this straightforward sampling technique is that, by definition, no bias is introduced. Nevertheless, this technique has one major disadvantage. Because the sampling is completely random, there are no guarantees that records are removed from each equivalence class. Therefore, some equivalence classes can remain complete, giving an advantage to attackers with membership knowledge of the original dataset.
- **Stratified sampling** tackles the aforementioned problem by forcing that records are removed from each equivalence class. This method first calculates the amount of records that need to be removed from each equivalence class (based on the sizes of the equivalence classes), after which the required amount of records are removed from each equivalence class randomly.
- **Balanced stratified sampling.** Data is increasingly acquired for use in machine learning applications. Many machine learning techniques heavily benefit from a balanced target attribute. However, by nature, many datasets are unbalanced. The *balanced stratified sampling* is a variation on the *stratified sampling* technique. Instead of applying full random sampling within each equivalence class, priority is given to removing a record that is over-represented in the target attribute.

4. Evaluation methodology

The goal of this work is to assess the impact of anonymization on the utility of datasets when applied in machine learning. Fig. 1 illustrates the evaluation methodology. The original dataset is first anonymized in a two-step process, combining sampling (*pre- or post-sampling*) and generalization (for achieving k-anonymity). Thereafter, a machine learning model is created using the gradient boosted decision tree algorithm by scikit-learn. Finally, the privacy and utility of the anonymized dataset are evaluated. The experiments are executed on census data extracted by the *Folktables* tool [10]. The dataset of 1.6M records contains attributes such as age, place of birth, sex, marital status and income. The goal is to create an accurate machine learning model to predict whether the income of citizens reaches a certain threshold. The tests are repeated for a set of different values for k, sampling strategies and sample sizes. Each test is executed multiple times in order to rule out sample-dependent results. The remainder of this Section first outlines both the utility and privacy measurements used for the evaluation, after which some preliminary results are presented and discussed.

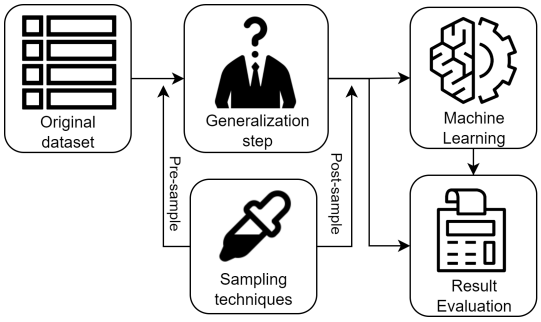


Figure 1. Evaluation workflow

4.1. Privacy measurements

The privacy properties of the anonymized datasets are evaluated by means of two metrics, namely *re-identification risk* and *certainty*. For calculating the risks, the *journalist risk* formula is applied [2], as the sampling step during anonymization ensures that an adversary is uncertain about the presence of the target in the anonymized dataset. The formula for calculating the journalist risk is presented in Equation (1), with f_j and F_j being the sizes of the equivalence classes in the sampled and the non-sampled dataset respectively. Note that the journalist risk is independent of the sample size. The *certainty* is calculated using Equation (2). It represents the certainty of an attacker that the target is in the sampled dataset.

$$Risk_j = \frac{f_j}{F_j} \cdot \frac{1}{f_j} = \frac{1}{F_j} \quad (1) \qquad Certainty_j = \frac{f_j}{F_j} \quad (2)$$

4.2. Utility measurements

This work focuses on two aspects of utility, namely case-agnostic utility and utility when applied for a specific machine learning purpose. In order to assess the former, this work analyses the selected generalization levels. Harsher generalizations typically imply increased information loss. It also compares the distribution (*chi-square* and *F-test*) of the data in the original dataset to the anonymized dataset. The machine learning utility is measured using the $accuracy$ (3) and F_1-$score$ (4). The models are evaluated by applying a 5-fold cross-validation on the dataset. In the equations below, TP and TN represent the amount of true positive and true negative predictions respectively, FN represents the amount of false negatives.

$$accuracy = \frac{TP + TN}{\#records} \quad (3)$$

$$F_1\text{-}score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (4)$$

4.3. Preliminary results

While this research is ongoing, some preliminary results are already available.

Fig. 2 presents a violin plot of the (journalist) risk for random pre- and post-sampling with k=20 and sample size 1/8. First of all, this plot demonstrates that results can vary between pre- and post-sampling. This difference can be attributed to the fact that pre-sampling reduces the amount of records in the dataset, and thereby the likeliness that records can be grouped in groups of size k. Therefore, the generalization algorithm is forced to impose harsher generalization levels to reach k-anonymity. Because the pre-sample has undergone harsher generalizations, the risk is lower compared to the post-sample. However, it should be noted that the post-sample dataset will score higher in utility metrics.

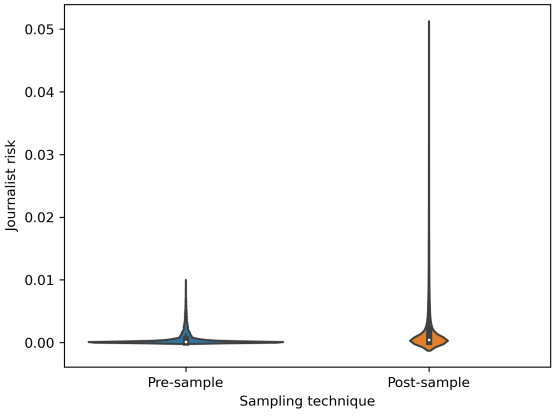


Figure 2. Journalist risk violin plots for random pre- and post-sampling with k = 20, samplesize = $\frac{1}{8}$

Fig. 3 presents a violin plot of the *certainty* for both the *full random* and *stratified* post-sample technique (k=5, sample size 0.5). This figure demonstrates that, while for most records the *random sampling* achieves its goal of introducing uncertainty for attackers, a set of records is underaffected by the sampling step (all records >0.5 in the left graph). Almost 50% of the records have a certainty over 0.5 and 0.2% (i.e. more than 2K records) have a certainty of over 80%. The *stratified sampling* strategy on the other hand boasts a maximal certainty of 0.5 (= the sample size).

5. Future work and conclusions

This work presented an overview of work in progress research on the effects of applying an additional sampling step in an anonymization pipeline. Different sampling strategies are presented and evaluated with respect to privacy and utility. However, many more experiments are required to finish this work. Firstly, we suspect that different combinations of sampling and generalization levels will achieve similar privacy properties but varying utility levels. Completing our experiments will lead to guidelines

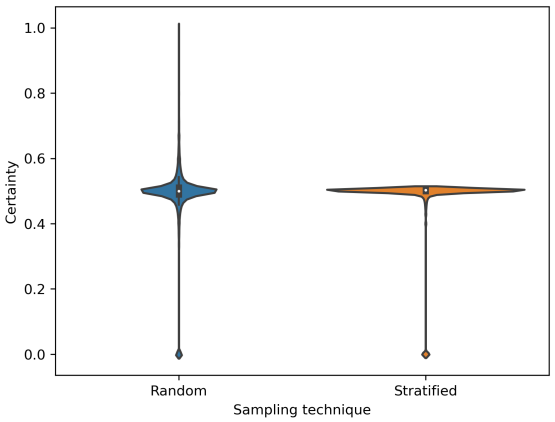


Figure 3. Certainty violin plots for random and stratified post-sampling with k = 5 and samplesize = $\frac{1}{2}$

for achieving optimal utility with respect to a certain privacy level. Moreover, based on our initial tests, our privacy and utility measurement metrics will undergo fine-tuning. Lastly, our tests currently cover one dataset and one specific machine learning problem. Expanding these tests to other data and algorithms will undoubtedly provide us with more complete and generic results.

References

[1] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[2] K. El Emam, *Guide to the de-identification of personal health information*. CRC Press, 2013.

[3] L. Rocher, J. M. Hendrickx, and Y.-A. De Montjoye, "Estimating the success of re-identifications in incomplete datasets using generative models," *Nature communications*, vol. 10, no. 1, pp. 1–9, 2019.

[4] R. Bild, K. A. Kuhn, and F. Prasser, "Safepub: A truthful data anonymization algorithm with strong privacy guarantees," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 1, pp. 67–87, 2018. [Online]. Available: https://doi.org/10.1515/popets-2018-0004

[5] N. Li, W. Qardaji, and D. Su, "On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, 2012, pp. 32–33.

[6] R. Hoogervorst, Y. Zhang, G. Tillem, Z. Erkin, and S. Verwer, "Solving bin-packing problems under privacy preservation: Possibilities and trade-offs," *Information Sciences*, vol. 500, pp. 203–216, 2019.

[7] T. Li and N. Li, "On the tradeoff between privacy and utility in data publishing," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 517–526.

[8] K. De Boeck, J. Verdonck, M. Willocx, J. Lapon, and V. Naessens, "Dataset anonymization with purpose: a resource allocation use case," in *2021 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*. IEEE, 2021, pp. 202–210.

[9] A. Inan, M. Kantarcioglu, and E. Bertino, "Using anonymized data for classification," in *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 2009, pp. 429–440.

[10] F. Ding, M. Hardt, J. Miller, and L. Schmidt, "Retiring adult: New datasets for fair machine learning," *arXiv preprint arXiv:2108.04884*, 2021.

Poster: The impact of public data during de-anonymization: a case study

Kevin De Boeck, Jenno Verdonck, Michiel Willocx, Jorn Lapon, Vincent Naessens
imec-DistriNet
KU Leuven
Ghent, Belgium
firstname.lastname@kuleuven.be

***Abstract*—Many companies, non-profit organizations and governmental bodies collect personal information during service interactions. However, releasing sensitive personal data may impose huge privacy risks. First, an increasing amount of sensitive personal information becomes publicly available online after user consent. Moreover, data breaches may result in huge data dumps that can contain personal records of millions of individuals. Hence, malicious entities are able to scrape, collect and combine personal data from multiple sources in order to compile detailed profiles of many individuals. This paper demonstrates the impact of publicly available data during de-anonymization by means of a concrete case study. Journalists are often reluctant or even prohibited to release the identity of suspects or victims in criminal cases. They do, however, often release initials and background (such as their age and residential location). Through a large scale study of over 132.000 news articles, this paper demonstrates that currently applied privacy measures are often insufficient and straightforward re-identification strategies can de-anonymize individuals.**

***Index Terms*—Privacy, re-identification, data breaches**

1. Introduction

Today, many companies and services we interact with in our daily lives collect data about individuals. Based on the acquired data, companies are able to offer personalized advertisements to customers and to build models to improve their service. Online service providers store preferences and personal interests for recommendations. While this is very convenient to end-users, huge data sources pose serious risks to our privacy. Besides data sources that are already publicly available online (e.g. social media and the whitepages), huge data breaches occur frequently. These breaches often result in huge amounts of personal data collected by services and companies being dumped on the internet.

Entities with doubtful or malicious intents can compile these dumps and scrape publicly available data in order to gain information on a large set of individuals. These data sources can in their turn be applied as background knowledge to perform large-scale re-identification attacks. For example, an anonymized dataset can be linked to the aforementioned publicly available datasets to support de-anonymization.

This work presents a practical use-case in which the negative impact of public data on the privacy of individuals is demonstrated. In news articles about crime-related cases, journalists are often reluctant or even prohibited to release the exact name and details of potential subjects or victims. They do however often opt to provide the reader with contextual information about these individuals. Examples are (partial) initials, the age and their residential location. If only the information in the newspaper was publicly available, only relatives would be able to de-anonymize individuals based on the info included in the article. However, due to the multiple publicly available data sources, interested readers can often quite easily re-identify a large set of individuals.

Contribution. This paper assesses the impact of public data for re-identification purposes. Firstly, this work categorizes publicly available data sources. Next, the negative impact of these data sources is assessed by means of a large-scale case study, namely the automatic re-identification anonymized individuals in news articles. Experimental results are presented and evaluated. The remainder of this work is structured as follows. Section 2 points to related work. Section 3 provides a general overview of public data sources. Next, our case study is presented in more detail. Preliminary results are discussed in section 5. This paper ends with conclusions and hints at future work.

2. Related work

Many research can be found on de-anonymization attacks. Pontes et al. [1] identify individuals based on the location information contained in reviews on the Foursquare platform. No data from external sources was required for this attack. Other papers, however, do rely on external data. Narayanan et al. [2] were able to link anonymized Netflix profiles to public IMDB profiles. Douriez et al. [3] demonstrated that drivers can be re-identified in an anonymized dataset by linking them to their unique cab ID. These works demonstrate that background knowledge and external data should be taken into consideration during anonymization. Several papers attempt to model the background knowledge of attackers [4]–[6] in order to improve the anonymization process. These papers approach background knowledge from a theoretical point of view, in which the knowledge of an attacker is fixed and pre-defined. In many real-life scenarios however, attackers have access to numerous data sources (e.g. data breaches and publicly available data). This work targets a real-life practical scenario in which the background knowledge of an adversary is not limited to a narrow, fixed amount of knowledge.

3. Public datasets

This work distinguishes three ways in which data can be acquired: data sharing, data scraping and data breaches.

Data sharing. Companies increasingly opt to share data with third parties. Data release can either occur in a bilateral agreement with a third party, or by publishing the data online.

Data scraping. This implies the collection of data beyond its intended use. Well-known examples of online data sources are the whitepages and social networks. Exploiting these online data sources by large-scale retrieval mechanisms allows malicious actors to optimize searches in the collected data, and additionally enables more complex and in-depth analysis of the data. Moreover, repeated crawling enables the construction of an historical view of an individual.

Data breaches. These occur when non-public data is accessed by an unauthorized person or party. Nowadays, this occurs frequently, as demonstrated by UpGuard [7] with their collection of the largest publicly-disclosed data breaches. Many well-known companies such as Facebook and Twitter have previously experienced multiple data breaches. The Facebook data breach [8] from 2019 is an example of a data breach impacting more than 500 million individuals, containing people's full names, phone numbers, locations, email addresses, dates of birth and more. The Facebook dataset is an example of a data breach that was dumped online for free. Other, more exclusive and harmful data is sold for premium prices.

4. Case study: News articles

The remainder of this work focuses on the data leakage by news articles. The goal is to investigate up to what extent individuals can be re-identified based on information given in news articles. News articles expose dates, locations and information about the persons involved such as their age, gender by means of pronouns, and surname. Journalists often replace full names with initials to protect an individual's privacy. The focus of this case-study is to reconstruct the full name of de-identified individuals based on the information in the news articles combined with publicly available data. This Section first presents the attacker model, after which the research methodology is outlined.

4.1. Attacker model

This study assumes a curious or malicious actor attempting to learn the full name of an individual based on the information included in a news article. The attacker attempts to single out one individual (or a small set of possible individuals) by searching through public datasets with key attributes contained in the article. Depending on the resources of the attacker, different capabilities are assumed. Weaker adversaries solely have access to freely available datasets (publicly shared data, data breach dumps and querying online platforms such as search engines), while stronger adversaries are able to scrape data and have access to a collection of datasets after a paywall.

Small scale attacks in which the identity of a few individuals in one of a limited numbers of articles must be exposed can be executed manually by an attacker. However, for the purpose of this paper, the whole process is automated and executed on a large set of news articles in order to correctly assess the broader privacy impact.

4.2. Research methodology

This research was executed in three steps. Firstly, a large set of news articles was crawled. Next, all relevant information was extracted from the crawled articles. Lastly, a re-identification attempt is made by mapping the extracted information to two public datasets.

Crawling news articles. A script was developed that automatically retrieves the title, abstract, content, author(s) and the date of all articles from an archive of news articles. Firstly, all articles present in the archive were crawled, starting from the year 1998 until January 2022. Next, articles were filtered based on two conditions: They had to contain keywords matching lawsuits (e.g. lawyer, trial, arrested) and keywords such as *theft*, *murder* or *extortion* in the abstract. Four prevalent news sites in Belgium were crawled for nearly 7 million articles. Based on the prerequisites, $\pm$132.000 articles were selected. Afterwards, the data contained in the articles was extracted.

Data extraction. For the data extraction, a hybrid approach is employed, combining natural language processing and regular expressions. Two NLP models were used, namely Spacy [9] and Flert [10]. The Spacy model was used for sentence extraction while the Flert model allowed named entity recognition. This means identification of names, organizations, locations,... The data extraction process works as follows. Firstly, sentences were extracted from the article. Next, names and locations were identified. Afterwards, using regular expressions, ages were extracted. Finally, all found information per individual was merged. This process resulted in approximately 292.000 individuals. For our research, only news articles from 2017 and later are considered since these are more likely to match on our public data sources. This research focuses on reconstructing full names from (partial) initials or a partial name (only first or last name). Hence, full names are omitted from the test dataset. This leaves a total of 13.865 individuals in the test dataset, of which 2712 and 2904 include a location and age respectively. For 1128 individuals, both a location and age are known.

Re-identification. During the re-identification phase, the known attributes (initials, age, location) are matched to two different datasets, namely the aforementioned publicly available Facebook data breach dataset and the whitepages. The whitepages dataset is an extract from the Belgian whitepages[1]. It consists of 1.912.055 entries, and contains a name and a location (full address) for each record. The Facebook dataset (Belgian extract) consists of 3.183.529 entries, of which 1.546.421 entries have a location and 96.649 records have an age associated. Only a small subset of 71.512 records contain both a location and an age. Naturally, a more complete dataset (ideally the full population with all attributes), would inevitably result in more accurate re-identifications.

1. Belgian PhoneBook: https://www.whitepages.be

5. Preliminary results

While this research is ongoing, it is possible to share some preliminary results. In the results presented below, a distinction is made between matches solely based on the names of individuals and matches based on the combination of name with another attribute (location, age). The more matching attributes, the higher the certainty of a correct match. Often, one individual in the test dataset matches to multiple individuals in the matching dataset (false positive results). A large amount of matches defines a low certainty for the attacker. Therefore, a second distinction is made between individuals with less than five matches and individuals with five or more matches. Note that the correct individual is not necessarily in the matching dataset.

The results from the experiment with the Belgian whitepages are presented in Table 1. For over half of the individuals in our test dataset, at least one match was found. Approximately a quarter of them had less than five matches. When matching a combination of name and location, only 787 individuals were matched, however half of them had less than five matches. When only considering the partial initials (complete first name and abbreviated last name or vice versa), 3666 individuals were matched (444 when matching name + location). The majority of these individuals (77%) had less than five matches.

TABLE 1. WHITE PAGES MATCHES WITH OUR TEST DATASET

Attribute	Matches	Initials	Partial name	Partial initials
Name	< 5	0	1198	778
	≥ 5	830	2567	2888
Name + Location	< 5	26	53	344
	≥ 5	245	19	100

The results from the experiment with the Facebook dataset are displayed in Table 2. 9600 individuals matched but only 19% had fewer than five matches. When also matching on location, 991 individuals were matched with half of them having fewer than five matches. Matching on age gives similar results. Matching on age and location often resulted in unique matches. However, due to the small number of records in the matching dataset that contain both an age and a location, the total amount of matched individuals for this category is low.

TABLE 2. FACEBOOK MATCHES WITH OUR TEST DATASET

Attribute	Matches	Initials	Partial name	Partial initials
Name	< 5	0	1249	618
	≥ 5	830	3240	3663
Name + Location	< 5	36	52	433
	≥ 5	238	65	167
Name + Age	< 5	25	58	481
	≥ 5	234	60	96
Name + Age + Location	< 5	20	2	7
	≥ 5	2	0	0

6. Conclusions and future work

This poster paper presented preliminary research results of a case study on the impact of public data for re-identification purposes. While the current results support our hypothesis that re-identification by employing publicly available data is a realistic threat when publishing anonymized data, it should be noted that the accuracy of the attack strongly depends on the quality of the datasets available for the attacker. For example, because neither matching datasets used in the experiments are complete, our current results display a large number of false positives. However, even in its current form, our demonstrator eases the re-identification of individuals significantly by narrowing the search-space to a small set of individuals. Moreover, additional improvements to our current work can be made. Firstly, when other more complete matching datasets (more attributes and/or more individuals) are available, the quality of our matches will undoubtedly increase. Secondly, the experiments with different matching datasets are currently executed separately. Combining all public data sources into one combined matching dataset could also significantly improve our results.

After the offensive line of research presented in this paper, a next step could be to apply our research results to create guidelines for taking into account public data when leaking anonymized information on individuals. In this case specifically, a software tool could warn journalists when they unintentionally compromise the privacy of an individual.

References

[1] T. Pontes, M. Vasconcelos, J. Almeida, P. Kumaraguru, and V. Almeida, "We know where you live: Privacy characterization of foursquare behavior," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 898–905.

[2] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 111–125.

[3] M. Douriez, H. Doraiswamy, J. Freire, and C. T. Silva, "Anonymizing nyc taxi data: Does it matter?" in *2016 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 2016, pp. 140–148.

[4] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 3–es, 2007.

[5] T. Li, N. Li, and J. Zhang, "Modeling and integrating background knowledge in data anonymization," in *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 2009, pp. 6–17.

[6] W. Du, Z. Teng, and Z. Zhu, "Privacy-maxent: integrating background knowledge in privacy quantification," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 459–472.

[7] "The 63 biggest data breaches (updated for february 2022)," Feb 2022. [Online]. Available: https://www.upguard.com/blog/biggest-data-breaches

[8] "Facebook data on 533 million users reemerges online for free," Apr 2021. [Online]. Available: https://www.bloomberg.com/news/articles/2021-04-03/facebook-data-on-533-million-users-leaked-business-insider

[9] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python," 2020.

[10] S. Schweter and A. Akbik, "Flert: Document-level features for named entity recognition," 2020.

POSTER: Towards Cyber Resilience of Cyber-Physical Systems using Tiny Twins

Fereidoun Moradi*
Mälardalen University
Västerås, Sweden
fereidoun.moradi@mdu.se

Sara Abbaspour Asadollah
Mälardalen University
Västerås, Sweden
sara.abbaspour@mdu.se

Marjan Sirjani
Mälardalen University
Västerås, Sweden
marjan.sirjani@mdu.se

***Abstract*—We propose a method to detect attacks on sensor data and control commands in cyber-physical systems. We develop a monitor module that uses an abstract digital twin, Tiny Twin, to detect false sensor data and faulty control commands. The Tiny Twin is a state transition system that represents the observable behavior of the system. The monitor observes the sensor data and the control commands transmitted in the network, walks over the Tiny Twin and checks whether the observed data and commands are consistent with the transitions in the Tiny Twin. The monitor produces an alarm when an attack is detected. The Tiny Twin is built automatically based on a timed actor code of the system. We demonstrate the method and evaluate it in detecting attacks using a temperature control system.**

***Index Terms*—Monitoring, Model Checking, Cyber-Physical Systems, Attack Detection and Prevention, Cyber-Security**

1. Introduction

Cyber-Physical Systems (CPSs) are safety-critical systems that integrate physical processes in the industrial plants (e.g., thermal power plants or smart water treatment plants) with sensors, actuators and controller components. Since these components are integrated via a communication network (usually wireless), a CPS is vulnerable to malicious cyber-attacks that may cause catastrophic damage to the physical infrastructure and processes. Cyber-attacks may be performed over a significant number of attack points and in a coordinated way. So, detecting and preventing attacks in CPSs are of significant importance.

Intrusion Detection Systems (IDSs) are deployed in communication networks to defend the system against cyber-attacks. Regular IDSs cannot easily catch complex attacks. They need to be equipped with complicated logic, based on human (safety and security engineers) reasoning [1]. In rule-based IDSs [1], a set of properties that are extracted from the system design specification are considered as rule-sets to detect attacks. Indeed, if an IDS finds a deviation between the observed packets in the network and the defined rules, it produces an alarm and takes a predefined action such as dropping the packets. The key challenge is the effort required to specify the correct system behavior as rules.

2. Overview

We propose a method to detect cyber-attacks on sensor data and control commands in CPSs. The overview of our method is shown in Figure 1. The model of a CPS is developed in Timed Rebeca [2] and the Afra model checker [3] is used to verify the model (step (1) in Figure 1). In [4], it is shown that how entities of a CPS, i.e., sensors, actuators, controllers, and physical plant are modeled as actors, and interactions between them are modeled as messages passed between the actors. We develop an abstraction tool (step (2) in Figure 1) that abstracts the state space of the Timed Rebeca model (generated by Afra model checking tool) to create a Tiny Twin (TT) [5]. Digital Twins (DT) [6] are used as digital representation of the system to advance the system monitoring. We develop a monitor module that uses the created Tiny Twin to track the order and the timing of events (step (3) in Figure 1).

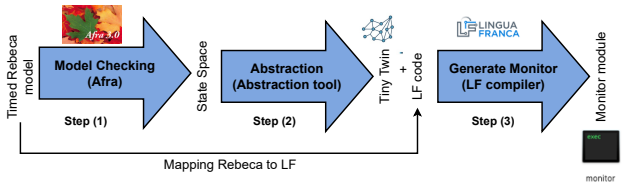


Figure 1: The overview of our method. Step (1), we generate the state space of the Timed Rebeca code of a CPS by the Afra model checker (see Sec. 3). Step (2), the state space is abstracted by our abstraction tool (see Sec. 4). The result of the abstraction is a Tiny Twin that is used within a monitor module (see Sec. 5) to detect the attacks. Step (3), we develop the monitor module in LF language and use the LF compiler to generate an executable file.

3. Model Checking and Security Analysis

We assess the security aspects of a CPS by verifying its security properties. Afra supports LTL, TCTL and assertions for property specification. By model checking we analyze security of the CPS design to recognize where the potential attack scenarios can successfully cause a failure in the system. We utilize the STRIDE [7] model as a reference for classifying potential attacks on a CPS. In Table 1, we classify the significant attacks on CPS based on the STRIDE categories. The cyber and physical attacks exploit emerging CPS-related vulnerabilities in the two aspects of *communication* and *component*, and are shown in Table 1 as *Scheme-A* and *Scheme-B*. *Scheme-A* consists of the attack scenarios which are secretly recording or modifying the data transmitted over the channels (e.g., eavesdropping, MITM and injection attack). *Scheme-B* includes the attacks that inject malicious code into the software components or perform a malicious alteration on a physical component (e.g., malware and physical attack).

TABLE 1: Attack Classification according STRIDE threat modeling [4].

Threat Type	*Cyber or Physical Attack*	*Scheme-A*	*Scheme-B*
Spoofing (Authentication)	Masquerade attack		[8]
	Packet spoofing attack	[9]	
Tampering (Integrity)	Man-in-the-middle (MITM)	[8]	
	Injection attack	[9]	[9]
	Replay attack	[8]	
	Malware (Virus or Worms)		[9]
	Physical attack	[9]	[10]
Reputation (Non-Repudiation)	On-Off attack		[10]
Information Disclosure (Confidentiality)	Eavesdropping	[8]	
	Malware (Spyware)		[9]
	Side-channel attack		[9]
	Physical attack	[9]	[10]
Denial of Service (Availability)	Resource exhaustion attack	[8]	[9]
	Interruption attack	[8]	
	Malware (Ransomware)		[9]
	Physical attack	[9]	[10]
Elevation of Privilege (Authorization)	Malware (Rootkit)		[9]

4. Abstraction and Tiny Twin

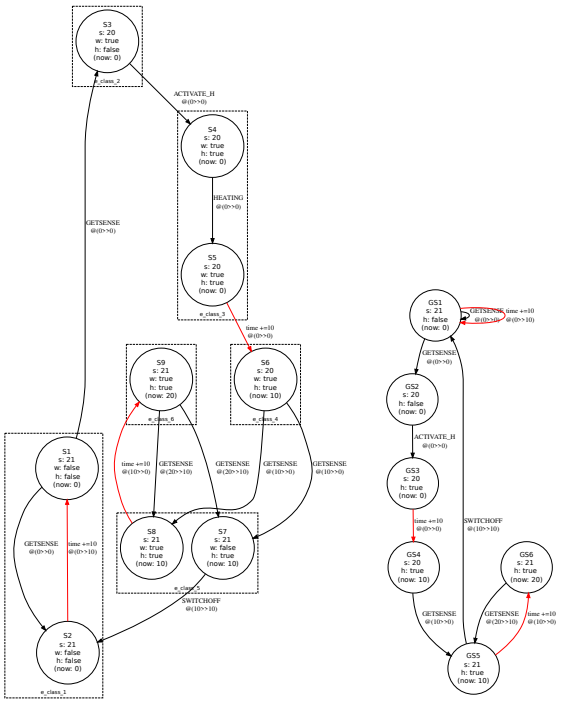
To create a Tiny Twin, we abstract the given state space with respect to the sensor data and control commands. We abstract the state space generated by the model checker in order to preserve sequences of observable actions while hiding internal actions. Our abstraction method is implemented in a tool by considering the reduction algorithm in [11]. It is applied to the original state space where it iteratively refines indistinguishable states, i.e., the classes containing equivalent states, while hides transitions that are called silent transitions. Figure 2 illustrates how the abstraction tool performs on an example.

The Tiny Twin defines the observable behavior of the system in the absence of an attack and contains the order and the time at which the sensor data and control commands are communicated. Transitions in Tiny Twin are tagged by a label that indicates an action or the progress of time.

5. Monitoring and Attack Detection

We develop a monitor module that observes the sensor data and the control commands transmitted in the network and decides to drop or pass the control commands to the actuators (see Figure 3). The Tiny Twin is placed within the module and serves as a baseline for detecting attacks. The module starts its observation when the system executes. Upon receiving sensor data, the module compares it with the state variables in the Tiny Twin. If the module finds no differences between them, it proceeds and checks whether the commands are consistent with the corresponding transitions in the model. If this is the case, the module sends the commands to the actuators. Otherwise, the module produces an alarm and terminates the process of monitoring.

We implement our monitor module in Lingua Franca (LF) [12] that is a language for programming CPSs. In principle, a Lingua Franca code can connect to the physical plant and the controller through the input/output communication channels in the actual system. We use Lingua Franca to simulate the system at runtime and evaluate the detection capability of our method by defining compromised components.



(a) (b)

Figure 2: (a) The transition system of an example Timed Rebeca model with the equivalence classes created by the abstraction tool. (b) The Tiny Twin of the transition system depicted in Figure 2(a) [5].

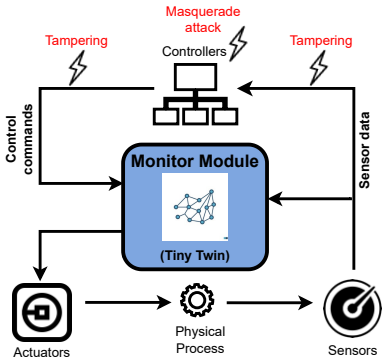


Figure 3: The monitor module observes the input/output of the controllers and drops faulty control commands if it identifies a mismatch between the state transitions in Tiny Twin and the observed sensor data and control commands.

6. Case Study: a Temperature Control System

We evaluate our method in detecting and preventing cyber-attacks using a temperature control system. The temperature control system is responsible for maintaining the temperature of a room at a desired range (e.g., the values between 21 and 23). This system includes a sensor, a hc_unit (heating and cooling unit) as an actuator, and a controller. The controller receives sensor data from the sensor and transmits the activate_c, activate_h or switch off command to the hc_unit to respectively activate the cooling or heating process, or switch the heating/cooling process off. We use the Afra model checker to produce the state space of the developed Timed Rebeca model and exploit our abstraction tool to generate the Tiny Twin. We implement both the system and the monitor module in LF.

6.1. Tiny Twin

We create the Tiny Twin of the state space of the developed Timed Rebeca model for the temperature control system. The Tiny Twin is generated by the abstraction tool based on the list V={*sensedValue*, *cooler_on*, *heater_on*} of state variables. The original state space of the model includes 76 states and 103 transitions while the generated Tiny Twin contains 21 states (i.e., equivalence classes) and 36 transitions. The Tiny Twin is trace equivalent to the original state space (projected on the variables containing sensors data and control commands).

6.2. Attack Types and Detection Capability

We evaluate the capability of the developed monitor module in detecting the attacks. We consider three types of attacks that target the *integrity* aspect of CPS (see Figure 3). (1) Attackers have the ability of tampering sensor data or injecting any arbitrary values into the vulnerable channel between controller and sensors, i.e., *replay or tampering attack*, (2) attackers are able to manipulate the controller through malicious code injection into the software of the controller, i.e., *fabrication or masquerade attack*, and (3) one or more attackers can perform a coordinated attack to force the system to change its correct functionally.

We consider the number of false sensor data and faulty control commands sent by the compromised components as the number of attacks. In our experiments, we simulate 20 false sensor data and 12 faulty control commands as listed in Table 2. We also simulate 240 coordinated attacks (combination of the false sensor data and the faulty control commands). We calculate the detection rate of the monitor with respect to the detected/undetected attacks. In this case study, the detection rate is around 68.8 percent.

TABLE 2: Attacks and detection capability of the monitor [5].

System States	*# Attacks*	*False sensor data/ Faulty control commands*	*Detection Capability* (DS/DC)
GS1 and GS2	4	Sensor data (20, 21, 23, or 24)	DS (20 and 24)
GS3 and GS5	4	Sensor data (20, 21, 22, or 24)	DS (20 and 21)
GS4 and GS6	4	Sensor data (20, 22, 23, or 24)	DS (23 and 24)
GS8	2	Command (*activate_h* or *switchoff*)	DC (*activate_h* and *switchoff*)
GS9	2	Command (*activate_c* or *switchoff*)	DC (*activate_c* and *switchoff*)
GS11 and GS13	4	Sensor data (20, 21, 22, or 23)	DS (20 and 21)
GS12 and GS14	4	Sensor data (21, 22, 23, or 24)	DS (23 and 24)
GS15, GS16, GS17, GS18	2	Command (*activate_h* or *activate_c*)	DC (*activate_h* and *activate_c*)

#Attacks.: Number of simulated attacks, *DS*: Detect false sensor data, *DC*: Detect faulty control commands.

Table 3 shows the alarms list returned by the monitor module when a false sensor data or a faulty control command is detected. The alarm is comprised of a time value, a false sensor data or a faulty control command, the status of the physical plant reported by the sensor and the value of the state variables in the state where the monitor module terminated the system execution.

TABLE 3: Alarms of the monitor module in case of attacks [5].

System States	*False sensor data/ Faulty control commands*	*Alarms list*
GS1 and GS2	Sensor data (20)	$[time, y^i : 20, y : 23, s : 22, c : false, h : false]$
GS3 and GS5	Sensor data (21)	$[time, y^i : 21, y : 22, s : 23, c : false, h : false]$
GS4 and GS6	Sensor data (23)	$[time, y^i : 23, y : 22, s : 23, c : false, h : false]$
GS8	Command (*activate_h*)	$[time, u^d : activate\_h, y : 24, s : 24, c : false, h : false]$
GS9	Command (*switchoff*)	$[time, u^d : switchoff, y : 20, s : 20, c : false, h : false]$
GS11 and GS13	Sensor data (21)	$[time, y^i : 21, y : 22, s : 24, c : true, h : false]$
GS12 and GS14	Sensor data (24)	$[time, y^i : 24, y : 22, s : 20, c : false, h : true]$
GS16	Command (*activate_c*)	$[time, u^d : activate\_c, y : 22, s : 22, c : true, h : false]$

time: The logical time which is derived using Lingua Franca code.

In a CPS, there may be several variables involved in the physical process as well as various sensors and actuators. The monitoring approach using the Tiny Twin enables us to consider only variables are affected during an attack (i.e., violation of properties). Tiny Twin provides relevant information about attacks that can be employed in mitigation techniques, backtracking and recovering the system after attacks. We have developed the models and the LF codes of two case studies (Pneumatic Control System and Secure Water Treatment system), for which the *monitor* module can properly detect the attacks on the system.

7. Conclusion and Future Work

In this work, we proposed a method for detecting cyber-attacks on CPSs. In particular, we used a Tiny Twin to detect the attacks on sensor data and control commands. We developed an abstraction tool to build the Tiny Twin, which is an abstract version of a state transition system representing the system correct behavior in the absence of an attack. In our method, we built a monitor module that executes together with the system. It produces an alarm if the sensor data or the control commands are not consistent with the state transitions in the Tiny Twin. We evaluated the capability of our method in detecting attacks on a temperature control system. As the future work, we aim to build a module to mitigate impacts of the attacks based on the predefined mitigation plans.

References

[1] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–29, 2014.

[2] M. Sirjani and E. Khamespanah, "On time actors," in *Theory and Practice of Formal Methods*, pp. 373–392, Springer, 2016.

[3] "Afra: an integrated environment for modeling and verifying rebeca family designs." https://rebeca-lang.org/alltools/Afra, 2022. [Online; accessed Feb 09, 2022].

[4] F. Moradi, S. A. Asadollah, A. Sedaghatbaf, A. Čaušević, M. Sirjani, and C. Talcott, "An actor-based approach for security analysis of cyber-physical systems," in *International Conference on Formal Methods for Industrial Critical Systems*, pp. 130–147, Springer, 2020.

[5] F. Moradi, M. Bagheri, H. Rahmati, H. Yazdi, S. A. Asadollah, and M. Sirjani, "Monitoring cyber-physical systems using a tiny twin to prevent cyber-attacks," in *International Symposium on Model Checking of Software (SPIN)*, 2022.

[6] M. Eckhart and A. Ekelhart, "A specification-based state replication approach for digital twins," in *Proceedings of the 2018 workshop on cyber-physical systems security and privacy*, pp. 36–47, 2018.

[7] A. Shostack, *Threat modeling: Designing for security*. Wiley, 2014.

[8] S. Choi, J.-H. Yun, and S.-K. Kim, "A comparison of ics datasets for security research based on attack paths," in *International Conference on Critical Information Infrastructures Security*, Springer, 2018.

[9] J.-M. Flaus, *Cybersecurity of industrial systems*. J. Wiley & Sons, 2019.

[10] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[11] D. N. Jansen, J. F. Groote, J. J. Keiren, and A. Wijs, "A simpler o (m log n) algorithm for branching bisimilarity on labelled transition systems," *arXiv preprint arXiv:1909.10824*, 2019.

[12] M. Lohstroh, C. Menard, A. Schulz-Rosengarten, M. Weber, J. Castrillon, and E. A. Lee, "A language for deterministic coordination across multiple timelines," in *2020 Forum for Specification and Design Languages (FDL)*, pp. 1–8, IEEE, 2020.