Instruction Level Disassembly through Electromagnetic Side-Chanel: Machine Learning Classification Approach with Reduced Combinatorial Complexity

V.M.Vaidyan
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa, USA
vaidyan@iastate.edu

Akhilesh Tyagi
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa, USA
tyagi@iastate.edu

ABSTRACT

EM side-channel can be quite effective at instruction level disassembly of the executing program. This leaks IP from Internet of Things (IoT) networks. This may also serve as a benign capability to reverse engineer IoT malware binaries. Power Side Channel instruction level disassembly state-of-the-art is capable of identifying instructions in a 2-3 stage pipeline at 50-200 MHz clock frequency with reasonable accuracy by grouping instructions. EM side-channel works at distance unlike power side-channel. Machine Learning models for instruction identification, Principal Component Analysis (PCA) for feature selection, Gaussian Process Classifiers (GPC), Adaptive Boosting (AB), Quadratic Discriminant Analysis (QDA), Naïve Bayes (NB), Support Vector Machines (SVM) and Convolutional Neural Network (CNN) for instruction classification were developed. Our results of implementation on a 2-stage pipelined architecture demonstrate that the EM side-channel classification approach identifies instructions in flight with 99% accuracy.

CCS Concepts

• Computing methodologies→ Machine learning •Computing methodologies→ Machine learning approaches→ Kernel methods→ Support vector machines •Computing methodologies→ Machine learning approaches→ Factorization methods→ Principal component analysis •Computing methodologies→ Machine learning algorithms→ Ensemble methods→ Boosting •Security and privacy→ Security in hardware→ Hardware reverse engineering

Keywords

Instruction Disassembly; Machine Learning; Hardware Security; IoT Devices; Computer Architecture; Electromagnetics

1. INTRODUCTION

Analysis and transformation of commercial-off-the-shelf and legacy software have many applications [1], [2], [3], [4], [5],[6], [7], [8], [9], [10], [11] like reverse engineering, security hardening, bug finding, code clone detection and refactoring. In addition, IoT malware is emerging as a new battleground in cybersecurity. With 18 billion devices related to the Internet of Things (IoT) [12], recent attacks like Mirai and Moose highlight the need to defend these devices. In fact, one of the fundamental problems is precisely disassembling the software in a black-box environment. In these cases, hardware side-channel methods are useful as they do not need access to the binary. They can disassemble instructions with side-channel Power signatures [13]. Even though, the task is seemingly simple, it is indeed highly challenging due to the complexity of compilation and optimizations.

However, while the state-of-the-art hardware disassembler performs well by grouping of instructions, disassembly is challenging on modern multicore/manycore processors due to combinatorial complexity of feature space. It is best captured by dimensionality of the problem $\boldsymbol{p_{cc}} * \boldsymbol{n_p} * \boldsymbol{n_i}$, where $\boldsymbol{p_{cc}}$ is number of power samples per clock cycle, $\boldsymbol{n_p}$ is number of pipeline stages, $\boldsymbol{n_i}$ is number of instructions in flight. For a demonstration of dimensionality of the problem, a single core with conservative number of power samples per clock cycle, number of pipeline stages, and number of instructions in flight parameters of 5, 14, and 128 respectively, has 8960 dimensions. In contrast, for a 4-core processor, it translates to 35840. Consequently, it translates to nonlinear time for Machine learning classifiers in this dimensionality. In fact, this is significantly more challenging than the dimensionality 128 problem of extraction of 128-bit AES key. Additionally, this needs to be done in one clock cycle of the order of 0.25ns. Full disassembly of binaries is a challenge in modern processors. To reduce combinatorial complexity in this case, we propose an alternate Electromagnetic spectrum-based Instruction Disassembly to predict instructions in pipeline. In particular, our approach is to first analyze Electromagnetic characteristics in spectral domain using Power spectral density. Subsequently, we reduce the dimensions of the data with Principal Component Analysis. Based on the reduced dimensionality data, a machine learning model is trained with Gaussian Process Classifier, Adaptive Boosting, Quadrative Discriminant Analysis, and Support Vector Machines. The resulting disassembler has high accuracy (e.g., the likelihood of mispredicting a true positive instruction is near 0%). In our empirical study with different instructions, it never misses any true positive instruction. It also has much lower requirements in terms of physical access to the device. It also needs smaller instruction model library compared to Power side-channel based Disassembly.

Our contributions are summarized as follows:

- We propose a novel Electromagnetic Instruction disassembly to reduce the combinatorial complexity of feature space

SPML 2020, October 22–24, 2020, Beijing, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7573-3/20/10…$15.00
DOI: https://doi.org/10.1145/3432291.3432300

captured by dimensions $(p_{cc} * n_p * n_i)$. This consequently reduces the nonlinear time for Machine learning classifiers in this dimensionality, thereby making it an ideal candidate for disassembly.

- We develop an Electromagnetic spectrum domain framework, with dimensionality reduction and feature selection using PCA for a set of features for the individual instructions as a training library first. This model is extended for multiple instructions in the pipeline where adjacent instruction signature interference needs to be modeled. Machine learning and deep learning classifiers like Gaussian Process Classifier, Adaptive Boosting, Quadratic Discriminant Analysis, Support Vector Machines, Multilayer Perceptron and Convolutional Neural Network are able to identify instruction sequences in flight.
- Our experiments on ATMeg328 demonstrate that our technique can predict instructions in flight with 99% accuracy

The rest of the paper is organized as follows. In Section 2, CMOS switching and EM characteristics are discussed. In Section 3, details about EM Characterization of DUT are discussed. Section 4 presents feature selection. Adaptive Boosting is presented in section 5. In Section 6, Quadratic Discriminant Analysis is discussed. Gaussian Process Classifiers are given in Section 7. In Section 8, details of Support vector Machines are discussed. Naïve Bayes classifier is given in Section 9. In Section 10, Multilayer Perceptron is discussed. Section 11 discusses Convolutional Neural Networks. Hardware implementation details are given in Section 12. Evaluation and performance of classifiers are discussed in Section 13. Section 14 summarizes the work.

2. CMOS SWITCHING AND ELECTROMAGNETIC CHARACTERISTICS

The power dissipated by digital logic gates can be characterized as static and dynamic. The input is kept at "1" or "0" in static case. Static power consumption then can be written as: $P_{static} = V_{DD} I_{supply}$. This is often leakage from V_{DD} to Ground. However, in case of dynamic switching, input switching causes charging or discharging of various capacitances. This in turn makes dynamic power consumption dependent on signal frequency. With small rise and fall times, the dynamic power consumption is entirely related to energy for charging and discharging load capacitances. Dynamic power can be given as, $P_{dynamic} = C_L V_{DD}^2 f$, where C_L is denotes total load capacitance and f being signal frequency. Hence, Dynamic power and in turn Electromagnetic waves are a reliable source to identify the instructions being executed inside the processor – particularly because power signature is correlated with data value or switching.

3. EM CHARACTERIZATION OF DUT AND FIELD RESPONSE

The level of field above antenna factor can be defined as ratio of electric or magnetic field of device under test to voltage induced on the probe, $AF = H(dB) - V(dB)$. Consider an antenna with radius a, total loop perimeter l, with s as a coordinate around the perimeter of loop [14]. Assuming a is small and loop is symmetrical about axis and incident field varies as $e^{j\omega t} = e^{jkct}$, we have $-jk\int_s cB.dS = \oint_s E.ds$. Here cB is the magnetic field and has same dimensions as E [50]. Magnetic field is sum of incident field B^i and the reradiated field B^τ. Using Helmholtz integrals and Ohm's law, after breaking up current to zero and first phase sequences, we get $I(s) = I^0(s) + I^1(s)$, $I\left(s + \frac{l}{2}\right) = I^0(s) - I^1(s)$. Then general integral equation for zero phase-sequence current is $-jk\iint_s cB^i.dS = \oint_s I^0(s) Z^i ds + \frac{j\omega\mu}{4\pi}\oint\oint I^0(s)\frac{e^{-jkR}}{R} ds.ds$ where Z^i is the internal impedance/unit length, $k = 2\pi/\lambda$, and R, the distance from source point to field point. From this relation, we can identify that in a very general case zero-phase sequence current alone is related to the incident magnetic field, $\boldsymbol{B^i}$.

If loop is small enough, integral of incident magnetic field can be written as B_{z0}^i, its normal component at center of loop times S the area of loop which is equivalent to neglecting even derivatives in Taylor series expansion of $\boldsymbol{B^i}$ at center of loop. Additionally, zero phase-sequence current can be approximated to first order by constant current I_0. As we know, low-frequency input admittance of the loop in constant current case is $Y_0 = \left[\oint Z^i ds + \frac{j\omega\mu}{4\pi}\oint\oint I^0(s)\frac{e^{-jkR}}{R} ds.ds\right]^{-1}$ and $h_b = -jkS$.

Defining unloaded magnetic sensitivity constant K_B dependent on probe geometry as $K_B = Y_0 h_b/\boldsymbol{\lambda}$ and approximations mentioned in last paragraph. Then $I^0(0)$ can be solved as: $I^0(0) \approx I_0 = \boldsymbol{\lambda} K_B(cB_{se}^i)$. Besides, as electric field in the plane of loop will not enter into I_0, consequently, I_0 is magnetic field at center of loop.

Consider dipole mode which is the first-phase-sequence current $\boldsymbol{I^{(1)}}$, directly dependent on electric field can be broken up into two parts: symmetric across x-axis and y-axis as $I_x^{(1)}$ and $I_y^{(1)}$ respectively. In fact, $I_x^{(1)}$ and $I_y^{(1)}$ are related to E_{z0}^i and E_{y0}^i. We can ignore $I_x^{(1)}$while determining load currents as the loads can be restricted to ones concentrated at $s = 0$ or $l/2$ only. Thereby, the current at $s = 0$ can be given as: $I_y^{(1)}$ (0) = $h_{eI} Y_I E_{y0}^i$, where Y_I, the input admittance about center of antenna can be found out directly by solving antenna problem directly. h_{eI} can be obtained by using Rayleigh-Carson reciprocal theorem in a two-port passive system. Consequently, leading to the expression $kh_{eI} = F_I$, where F_I is the far-zone field factor of each half at midplane of broadside direction. Assuming I_y to be the y component while transmitting and I_{y0} being value at driving point, F_I can be defined as $F_I = \frac{k}{I_{y0}}\int_{-\boldsymbol{l/4}}^{\boldsymbol{l/4}} I_y(s)ds$. Then unloaded electric sensitivity can be written as: $h_{eI} = \frac{2}{I_{y0}}\int_0^{l/4} I_y(s)ds$. And relation for $I_y^{(1)}$ (0) can be rewritten to be $I_y^{(1)}$ (0) $= \lambda K_E E_{y0}^i$, where $K_E = h_{eI} Y_I/\lambda$. As average magnetic field perpendicular to the plane does not enter $I_y^{(1)}$, $I_y^{(1)}$ can be considered reasonably as a measure of parallel component of electric field. In case of a singly loaded receiving loop, we can replace load by an equivalent generator $V = -IZ_L$ based on compensation theorem. Then, effective current is sum of the currents of unloaded receiving loop from external fields and transmitting current from equivalent generator. Let $I^T(s) = Vv(s)$, which implies: $I(0) = I^{(0)}(0) + I_y^{(1)}(0) - I(0)Z_L v(0)$. However, $v(0)$is total input admittance Y for loop while driven at $s = 0$. We can find the load current as: $I_L = I(0) = \lambda K_B^{(1)} cB_{z0}^i + \lambda K_E^{(1)} E_{y0}^i$, having single loaded sensitivity constants as: $K_B^{(1)} = \frac{Y_L}{Y_L + Y} K_B$ and $K_E^{(1)} = \frac{Y_L}{Y_L + Y} K_E$. For loop loaded at $s = \frac{l}{2}$, equivalent to rotating 180^0 in its plane, load current becomes $I_L' = I(l/2) =$

$\lambda K_B^{(1)} c B_{z0}^i - \lambda K_E^{(1)} E_{y0}^i$. Hence, both readings of I_L and I_L' are required to measure the magnetic field and neither reading is adequate alone.

4. FEATURE SELECTION

Owing to the fact that EM traces have large number of sampling points, the dimensionality of the EM signature is large. Accordingly, a mapping, $x \rightarrow Wx$, of the large dimension EM signature with $Wx \in R^n$ being the lower dimensionality representation of x, and matrix $W \in R^{n,d}$, where $n < d$, reduces the dimensionality of vectors $x_1, x_2, \ldots\ldots, x_m$. Principal Component Analysis (PCA) can be used to find the compression matrix W and the recovering matrix U, ensuring total squared distance between both being minimal [15]. Further, this can be implemented by solving the problem, $\underset{W\in R^{n,d}, U \in R^{n,d}}{\operatorname{argmin}} \sum_1^m \| \|x_i - UWx_i\| \|_2^2$. The solution to PCA optimization problem is to set U to be the matrix whose columns are $u_1, u_2, \ldots\ldots, u_n$, where $u_1, u_2, \ldots\ldots, u_n$ are eigenvectors of matrix $A = \sum_{i=1}^m x_i x_i^T$ and set $W = U^T$. In this paper, certain number of principal components based on significance were chosen by dividing each absolute value by the sum of all eigen values.

5. ADAPTIVE BOOSTING

Adaptive Boosting (AdaBoost), which finds a hypothesis with low empirical risk, receives a training set of EM signatures, $S = (x_1, y_1), \ldots\ldots\ldots, (x_m, y_m)$, with each i, $y_i = f(x_i)$, for a labelling function, f [15]. Boosting process proceeds with sequence of consecutive rounds, with booster defining a distribution $\mathbf{D}^{(t)}$ in S at round t. *i.e,* $\mathbf{D}^{(t)} \in R_+^m$ and $\sum_{i=1}^m D_i^{(t)} = 1$. Then distribution and sample are passed on to the weak learner. The weak learner then constructs *i.i.d.* examples according to $\mathbf{D}^{(t)}$ and f. Weak learner returns a "weak" hypothesis, h_t with error,

$\epsilon_t \overset{\text{def}}{=} L_{\mathbf{D}^{(t)}}(h_t) \overset{\text{def}}{=} \sum_{i=1}^m D_i^{(t)} 1_{[h_t(x_i) \neq y_i]} \leq \frac{1}{2} - \Upsilon$. Then AdaBoost assigns a weight which is inversely proportional to the error of h_t as $w_t = \frac{1}{2} log\left(\frac{1}{\epsilon_t} - 1\right)$. During end of round, higher probability mass is assigned if h_t errs and lower probability mass if h_t is correct. This means, weak learner focuses on the problematic examples in the next round. Output of the algorithm is based on weighted sum of all the weak hypotheses. To prove that training error of the output hypothesis decreases exponentially fast with the number of boosting rounds, let us assume that for each t, denote $f_t = \sum_{p \leq t} w_p h_p$. Therefore, f_T is the output of AdaBoost. Additionally, $Z_t = \frac{1}{m} \sum_{i=1}^m e^{-y_i f_t(x_i)}$. For any hypothesis, $1_{[h_t(x_i) \neq y_i]} \leq e^{-yh(x)}$. Thereby, $L_s f_T \leq Z_T$, and it suffices to show that $Z_T \leq e^{-2\Upsilon^2 T}$. Upper bound Z_T can be written as, $Z_T = \frac{Z_T}{Z_0} = \frac{Z_T}{Z_{T-1}} \cdot \frac{Z_{T-1}}{Z_{T-2}} \ldots\ldots\ldots\ldots \frac{Z_2}{Z_1} \cdot \frac{Z_1}{Z_0}$. However, we use the fact that as $f_0 \equiv 0, Z_0 = 1$. Therefore, it suffices to show that in every round, $\frac{Z_{t+1}}{Z_t} \leq e^{-2\Upsilon^2}$. In order to do that, an inductive argument, for all t and i, $D_i^{(t+1)} = \frac{e^{-y_i f_t(x_i)}}{\sum_{j=1}^m e^{-y_j f_t(x_j)}}$. Then, $\frac{Z_{t+1}}{Z_t} = \frac{e^{-y_i f_{t+1}(x_i)}}{\sum_{j=1}^m e^{-y_j f_t(x_j)}}$ $= 2\sqrt{\epsilon_{t+1}(1 - \epsilon_{t+1})}$. But, in our assumption, $\epsilon_{t+1} \leq \frac{1}{2} - \Upsilon$, and since function $g(a) = a(1 - a)$ monotonically increases in [0,1/2], we get, $2\sqrt{\epsilon_{t+1}(1 - \epsilon_{t+1})} \leq 2\sqrt{\left(\frac{1}{2} - \Upsilon\right)\left(\frac{1}{2} + \Upsilon\right)} = \sqrt{1 - 4\Upsilon^2}$. Lastly, using inequality $1 - a \leq e^{-a}$ we have $\sqrt{1 - 4\Upsilon^2} \leq e^{-\frac{4\Upsilon^2}{2}} = e^{-2\Upsilon^2}$ showing $\frac{Z_{t+1}}{Z_t} \leq e^{-2\Upsilon^2}$ holds and hence training error is at most $L_S(h_s) = \frac{1}{m} \sum_{i=1}^m 1_{[h_t(x_i) \neq y_i]} \leq e^{-2\Upsilon^2 T}$. Every iteration of AdaBoost has $O(m)$ operations and a single call to the weak learner. Consequently, if the weak learner is implemented efficiently, the total training process will be efficient.

According to the definition of weak learner, failure probability is δ. The probability of non-failure of weak learner in all the iterations, based on union bound is $1 - \delta T$. By way of contrast, as the sample complexity dependence on δ can be logarithmic in 1/δ, it is not problematic to invoke weak learner with very small δ. Consequently, we can assume δT to be small. Moreover, weak learner is applied on distributions over training set, and thereby can be implemented with zero probability of failure.

6. QUADRATIC DISCRIMINANT ANALYSIS

Let, $p_0(x) = p(x|Y = 0)$ and $p_1(x) = p(x|Y = 1)$ be multivariate Gaussians: $p_k(x) = \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} exp\left\{-\frac{1}{2}(x - \mu_k)^T\right\}$, $k = 0,1$ and Σ_1, Σ_2 are $d \times d$ covariance matrices. Hence, $X|Y = 0 \sim N(\mu_0, \Sigma_0)$ and $X|Y = 1 \sim N(\mu_1, \Sigma_1)$. By definition, Bayes rule is $h^*(x) = I(\pi_1 p_1(x) > (1 - \pi_1) p_0(x))$. When specific forms of p_0 and p_1 are plugged-in and logarithms are taken, we obtain $h^*(x) = 1$ If and only if $(x - \mu_1)^T \Sigma_1^{-1} (x - \mu_1) - 2log\pi_1 + \log(\Sigma_1) < (x - \mu_1)^T \Sigma_0^{-1} (x - \mu_0) - 2log(1 - \pi_1) + \log(|\Sigma_1|)$. Then Bayes rule follows to be:

$$h^*(x) = \begin{cases} 1 \; if \; r_1^2 < r_0^2 + 2log\frac{\pi_1}{1 - \pi_1} + log\left(\frac{|\Sigma_0|}{|\Sigma_1|}\right) \\ 0 \qquad\qquad\qquad otherwise \end{cases} \quad (1)$$

Where $r_1^2 = (x - \mu_1)^T \Sigma_1^{-1} (x - \mu_1)$ is Mahalanobis distance. Assuming $\pi_0 = 1 - \pi_1$, an equivalent way of writing Bayes rule is $h^*(x) = \underset{k \in \{0,1\}}{\operatorname{argmax}} \delta_k(x)$ where, $\delta_k(x) = -\frac{1}{2}\log(|\Sigma_k|) - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)$ is the Gaussian discriminant function. The decision boundary can be characterized as the set $\{x \in \chi : \delta_1(x) = \delta_0(x)\}$ which is quadratic and so Quadratic Discriminant Analysis.

7. GAUSSIAN PROCESS CLASSIFIERS

Assuming an electromagnetic signature library D of data points x_i with binary class labels $y_i \in \{-1, 1\}$: $D = \{(x_i, y_i) | i = 1, 2, \ldots\ldots\ldots, n\}, X = \{y_i | i = 1, 2, \ldots\ldots\ldots, n\}$[16]. Even more, with this data set, our goal is to identify correct class label for a new data point $\hat{x}$. This can be obtained by computing the class probability $p(\hat{y}|\hat{x}, D)$. On assuming class label to be obtained from some real valued latent variable $\hat{f}$ (value of some latent function $f(\,.\,)$ evaluated at x), a Gaussian prior can be put on this function, meaning any number of points evaluated have a multivariate Gaussian density. Assuming GP prior is parameterized by hyperparameters Θ, the probability of interest can be written as:

$$p(\hat{y}|\hat{x}, D, \Theta) = \int p\left(\hat{y}|\hat{f}, \Theta\right) p(\hat{f}|D, \hat{x}, \Theta) d\hat{f} \quad (2)$$

Second part of the relation is obtained by further integrating over $f = [f_1 f_2 \ldots\ldots f_n]$, the values of latent function over data points.

$$p\left(\hat{f}|D, \hat{x}, \Theta\right) = \int p\left(f, \hat{f}\,|\, D, \hat{x}, \Theta\right) df$$
$$= \int p\left(\hat{f}|\hat{x}, f, \Theta\right) p(f|D, \Theta) df \quad (3)$$

where $p\left(\hat{f}|\hat{x}, f, \Theta\right) = p\left(f, \hat{f}|\hat{x}, \{x_i\}, \Theta\right) / p(f|\{x_i\}, \Theta)$ and $p(f|D, \Theta)$ is $\{\prod_{i=1}^n p(y_i|, f_i, \Theta)\} p(f|, X, \Theta)\}$. The primary term is likelihood for each observed class given the latent value and latter term is GP prior over functions evaluated at data and $\hat{x}$. A form of likelihood term $p(y_i|f_i,\Theta) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{y_i f(x_i)} \exp(-\frac{z^2}{2})dz = \text{erf}(y_i f(x_i))$. GP prior over functions can be given as

$$p(f|\Theta) = \frac{1}{(2\pi)^{\frac{N}{2}}(C_\Theta)^{\frac{1}{2}}} \exp(-\frac{1}{2}(f - \int_{-\infty}^{y_i f(x_i)} \exp(-\frac{1}{2}(f-\mu)^T C_\Theta^{-1}(f-\mu)) \quad (4)$$

While f depends on x implicitly, and mean μ is usually assumed as zero vector and each term of a covariance matrix C_{ij} is a function of x_i and x_j. In general, class probability can be calculated by $p(\hat{y}|\hat{x},D) = \int p(\hat{y}|\hat{x},D,\Theta)p(\Theta|D)d\Theta$.

8. SUPPORT VECTOR MACHINES

Consider an EM training library, $S = (x_1,y_1),\dots\dots,(x_m,y_m)$ with $x_i \in R^d$ and $y_i \in \{\pm 1\}$ and can be said to be linearly separable if a half space (w,b) exists such that $y_i = sign(\langle w,x_i\rangle + b)$ for all i [15]. Condition can be rewritten to be, $\forall i \in [m],\ y_i(\langle w,x_i\rangle + b) > 0$. The distance between a hyperplane and point x is defined as, $\min\{\|x-v\|: \langle w,v\rangle + b = 0\}$.

Assuming $v = x - (\langle w,x_i\rangle + b)w$,

$$\langle w,x\rangle + b = \langle w,x\rangle - (\langle w,x\rangle + b)\|w\|^2 + b = 0 \quad (5)$$

$$\text{and } \|x-v\| = |\langle w,x\rangle + b|\|w\| = |\langle w,x\rangle + b| \quad (6)$$

Distance will utmost be then, $|\langle w,x\rangle + b|$. Now, taking any point u on the hyperplane, $\langle w,u\rangle + b = 0$.

$$\begin{aligned}\|x-u\|^2 &= \|x-v+v-u\|^2 \\ &= \|x-u\|^2 + 2(\langle w,x\rangle + b)\langle w,v-u\rangle \\ &= \|x-v\|^2 \quad (7)\end{aligned}$$

The equality is because $\langle w,v\rangle = \langle w,u\rangle = -b$. Therefore, distance between x and u is at least the distance between v and x. On the basis of this claim, closest point in training set to separating hyperplane is $\min_{i\in[m]}|\langle w,x_i\rangle + b|$. Therefore, the SVM rule is: $\underset{(w,b):\|w\|=1}{\text{argmax}} \min_{i\in[m]}|\langle w,x_i\rangle + b|$ s.t. $\forall i \in [m],\ y_i(\langle w,x_i\rangle + b) > 0$. It will be a separable case if there is a solution to the preceding problem, and equivalent problem can be written as: $\underset{(w,b):\|w\|=1}{\text{argmax}} \min_{i\in[m]} y_i(\langle w,x_i\rangle + b)$. Another equivalent formulation as a quadratic optimization problem can be written as: $\underset{(w,b):\|w\|=1}{\text{argmax}} \min_{i\in[m]} y_i(\langle w,x_i\rangle + b)$. Another equivalent formulation as a quadratic optimization problem can be written as:

Input: $(x_1,y_1),\dots\dots,(x_m,y_m)$

Solve: $(w_0,b_0) = \underset{(w,b)}{\text{argmin}}\|w\|^2$

$$\text{s.t. } \forall i \in [m],\ y_i(\langle w,x_i\rangle + b) \geq 1$$

Output: $\hat{w} = \frac{w_0}{\|w_0\|}, \hat{b} = \frac{b_0}{\|w_0\|}$

Intuitively, SVM searches for w of minimal norm among all vectors that separate the data and for which $|\langle w,x\rangle + b| \geq 1$. Hence, calculating largest margin halfspace is finding w whose norm is minimal.

9. NAIVE BAYES

From probability perspective, based on Bayes Rule, probability of an event, $E = (x_1,x_2,\dots\dots,x_n)$ in class c is $p(c|E) = [p(E|c)p(c)]/p(E)$. E is classified as class $C = +$ if

$$f_b(E) = \frac{p(C=+|E)}{p(C=-|E)} \geq 1, \quad (8)$$

Where $f_b(E)$ being the Bayesian classifier. Assuming independence of attributes, given value of class variable, $p(E|c) = p(x_1,x_2,\dots\dots,x_n|c) = \prod_{i=1}^n p(x_i|c)$, Naïve Bayes classifier can be defined as,

$$f_{nb}(E) = \frac{p(C=+|E)}{p(C=-|E)}\prod_{i=1}^n \frac{p(x_i|C=+|E)}{p(x_i|C=-|E)} \quad (9)$$

10. MULTILAYER PERCEPTRON

Assuming use of an input layer with n_0 neurons $X = (x_0,x_1,\dots\dots,x_n)$, sigmoid activation function $f(x) = 1/(1+e^{-x})$, hidden layers $(h_1,h_2,\dots\dots,h_N)$ and n_i is number of neurons in hidden layer h_i, for output of first hidden layer, $h_i^j = f(\sum_{k=1}^{n_{i-1}} w_{k,j}^0 x_k)$, where $j = 1,2,\dots,n_i$, outputs can be computed as:

$$h_i^j = f\left(\sum_{k=1}^{n_{i-1}} w_{k,j}^{i-1} h_{i-1}^k\right) \quad (10)$$

for $i = 1,2,\dots,N$ and $j = 1,2,\dots,n_i$

Where $w_{k,j}^l$ is the weight of neuron in hidden layer and i and $i+1$. Then, output of i^{th} layer is $h_i T = (h_i^1, h_i^2, h_i^3,\dots, h_i^{n_i})^T$ and network output for output layer Y can be defined as: $y_i = f(\sum_{k=1}^{n_N} w_{k,j}^N h_N^k)$ with $Y = (y_1,y_2,\dots,y_j,\dots,y_{N+1}) = F(W,X)$ where $w_{k,j}^N$ is the weight of neuron in N^{th} hidden layer and output layer, n_N is number of neurons in N^{th} hidden layer, F is transfer function, and $W = [W^0,W^1\dots,W^j,\dots W^N]$ is the matrix of weights. Objective function to be optimized is the error calculated between the obtained output and desired output.

11. CONVOLUTIONAL NEURAL NETWORK

The feature value at (i,j) in the kth feature map of lth layer, $z_{i,j,k}^l$, is $z_{i,j,k}^l = \boldsymbol{w}_k^{l\,T}\boldsymbol{x}_{i,j}^l + b_k^l$, with $\boldsymbol{w}_k^l$ and b_k^l being weight and bias vectors respectively, $\boldsymbol{x}_{i,j}^l$ is input patch at (i,j) of lth layer [17]. Assuming $a(\cdot)$ is nonlinear activation function, $a_{i,j,k}^l$ of convolutional feature $z_{i,j,k}^l$ then can be computed as $a_{i,j,k}^l = a(z_{i,j,k}^l)$. In this paper, sigmoid ReLU ($a_{i,j,k} = \max(z_{i,j,k}, 0)$) is the activation function. Furthermore, denoting pooling function as pool$(\cdot)$, for every feature map $a_{:,:,k}^l$: $y_{i,j,k}^l = \text{pool}(a_{m,n,k}^l)$, $\forall (m,n) \in \Re_{i,j}$, where $\Re_{i,j}$ is local neighborhood around (i,j). In fact, max pooling was used in identifying instruction traces. Subsequently, several convolutional and pooling layers will result in one or more fully-connected layers which perform high-level reasoning. Output layer is a softmax operator. Assuming N desired input-output relations $\{(x^{(n)},y^{(n)}); n \in [1,\dots,N]\}$, where $x^{(n)}$ is the n-th input data, $y^{(n)}$ is corresponding target label, $o^{(n)}$ is output of CNN and θ, the loss in CNN will be $\mathcal{L} = \frac{1}{N}\sum_{n=1}^N \ell(\theta; y^{(n)}, o^{(n)})$. Minimization of loss function is mainly by optimizing CNN network with estimation of the gradients based on single randomly picked $(x^{(t)},y^{(t)})$ mainly with

Stochastic Gradient Descent from the training set: $\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \mathcal{L}(x^{(t)}, y^{(t)})$. Momentum update is usually, $v_{t+1} = \gamma v_t - \eta_t \nabla_\theta \mathcal{L}(x^{(t)}, y^{(t)})$ and $\theta_{t+1} = \theta_t + v_{t+1}$, with v_{t+1} being the current velocity vector and γ is usually the momentum term set to 0.9.

12. HARDWARE IMPLEMENTATION

Pipeline operation the DUT is given in Fig.1. The feature vectors obtained for training and classification of opcode of instructions is mainly due to the EM spectrum variations from CMOS switching in the Execution stage of individual instructions. In Fig.2., details of the single instruction execution are outlined showing register operand fetch, ALU operation execute and Result write back stages. All the operations are single cycle for the DUT. As we can see from Fig.1., Instruction Execute of first instruction and Instruction Fetch of next instruction overlap due to the pipeline. This is a challenge in accurately identifying the instruction. As identifying two instructions in pipeline is the major challenge, in this work we mainly focused on the overlapping area and tried to separate first instruction from the following instructions. We assumed arbitrary register operands. Hardware setup developed for Instruction Disassembly experiments is given in Fig.3. The EM traces of $i_1, i_2, ...$ and instructions in pipeline $i_1 i_2, ...$ of Atmega328 were first received through TPBS01 EM probe and the traces were identified on DPO 4032 oscilloscope with 350MHz bandwidth and 1.5GS/s sampling rate. The probe was kept at a 10cm distance from DUT as a receiver. Oscilloscope and probe were calibrated with a sleep and trigger mechanism on Atmega. The EM trace was obtained through UART interface and preprocessed for EM signal analysis. Further, EM spectral analysis was done to identify probability of instruction in the frequency spectrum. Due to the dimensionality of feature space, features were selected using Eigen division and Principal Component Analysis thereby feature space was reduced to 200 dimensions. Subsequently, the processed signal was trained with feature vector of individual instructions $i_1 i_2, ...$ to 100% accuracy into different classes. Furthermore, instructions in pipeline were classified into the equivalent classes using Gaussian Process Classifiers (GPC), Adaptive Boosting (AdaBoost), Quadratic Discriminant Analysis (QDA), Support Vector Machines (SVM) and Convolutional Neural Network. Finally, the classification accuracy of instruction in pipeline was evaluated using ML classifiers.

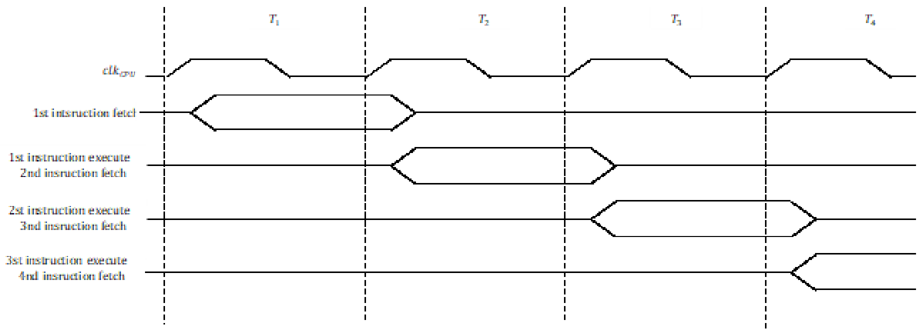


Figure 1. Pipeline Diagram of the DUT

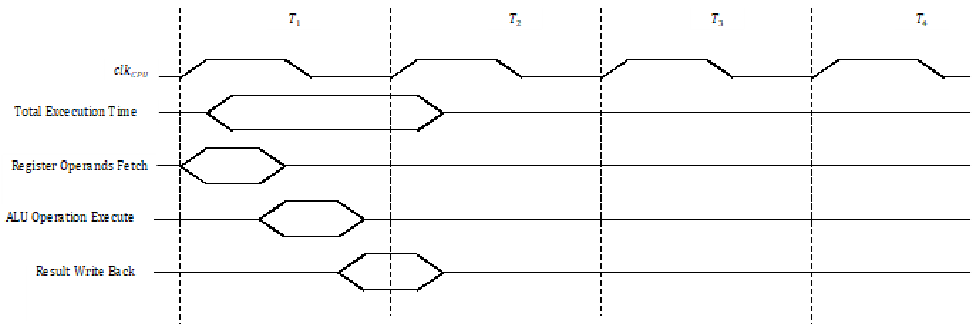


Figure 2. Single Cycle ALU operation of DUT

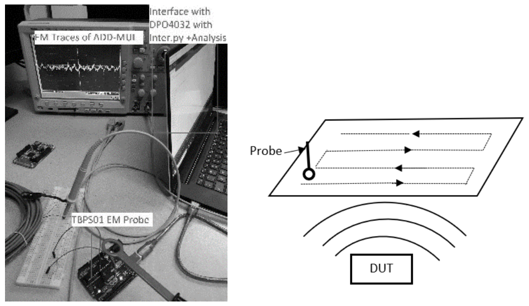


Figure 3. EM Instruction Disassembly Hardware

13. RESULTS AND EVALUATION

We recorded a stream of two instructions in flight in 2-stage pipeline of ATmega328. Figure 4 illustrates a typical Electromagnetic spectrum. Fig. 5b shows an illustrative instruction stream in pipeline. The onset of every EX stage of new instruction manifested onto the EM spectrum. For each instruction, it was classified based on models of the pre-trained independent instructions. To illustrate the effectiveness of EM based instruction disassembly in pipelined architectures, we targeted instructions ADD and MUL stream. The feature set used is the amplitude and time in plot. Initially, EM spectrum had 200 dimensions for every instruction. Fig.5a shows the important principal components with highest variance to be the first 8 components. Principal component analysis and Eigen division approaches were used for dimensionality reduction and feature selection to identify the features with highest variance and hence important features. In the experiment, a loop of individual instructions in stream was run with loop count, $n = 200$. Hence, 200 instructions of individual instruction classes were obtained.

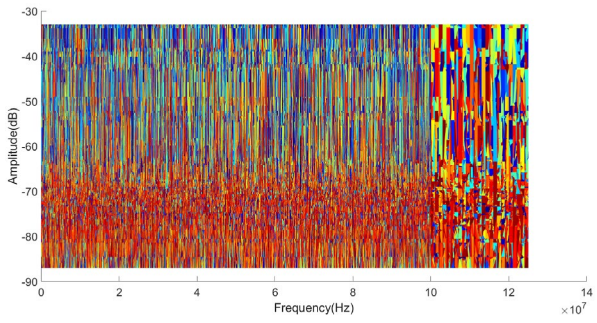
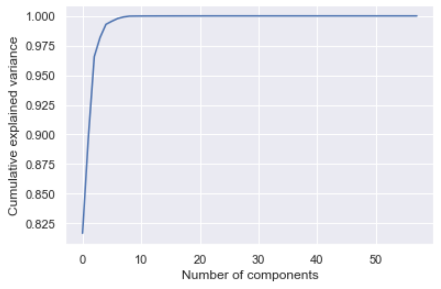


Figure 4. Electromagnetic Spectrum of ADD-MUL pipeline



```
"add r29, r24 \n\t"
"mul r29, r24 \n\t"
"add r29, r24 \n\t"
"mul r29, r24 \n\t"
"add r29, r24 \n\t"
"mul r29, r24 \n\t"
"add r29, r24 \n\t"
"mul r29, r24 \n\t"
"add r29, r24 \n\t"
"mul r29, r24 \n\t"
```

Figure 5. (a) Cumulative Explained Variance of Training data of Instructions (b) Instruction stream in pipeline

To test the accuracy of prediction, a loop with loop count of $n = 200$ was run with two different instructions in pipeline. Then, the Machine learning classifiers were used to identify each instance as classify it into separate instruction classes. We characterized the performance of different Machine Learning classifiers based on prediction capability of overlapping instructions due to the pipeline which is summarized in Figure 6.

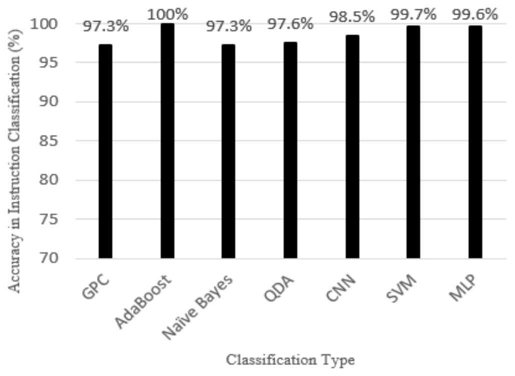


Figure 6. Performance evaluation of classification approaches for two instruction stream in 2-stage pipeline architecture

For evaluation of performance, Gaussian Process Classifier, AdaBoost, Naïve Bayes, Quadratic Discriminant Analysis, Support Vector Machine, and Multilayer Perceptron were implemented in Python. The addition of feature selection based on Eigen division increased the accuracy of instruction prediction in our problem. Overall, on average across all the classifiers, 99% of instructions in flight in the pipeline were detected accurately. However, for our disassembly dataset with overlapping EM spectrum, AdaBoost, SVM and MLP showed the best predictions of instructions being executed. MLP with 'lbfgs' solver, alpha=1e-5, hidden_layer_sizes=(5, 2) showed the best accuracy. AdaBoost with 70 n_estimators and a learning_rate of 1 showed the best performance.

14. CONCLUSION

A novel Electromagnetic spectrum based Instruction disassembly of pipelined architectures based on Machine Learning with reduced combinatorial complexity is introduced. The training models can be built without tampering the device unlike for power side-channel where physical power tap is needed. Compared to the best related existing work, there is no requirement to tamper with the device. The training feature vectors are for individual instructions rather than for instruction combinations or groups, thereby reducing combinatorial complexity of hierarchical classification. Performance evaluation with Gaussian Process Classifiers (GPC), Adaptive Boosting (AB), Quadratic Discriminant Analysis (QDA), Naïve Bayes (NB), Support Vector Machines (SVM) and Convolutional Neural Network (CNN) for two instruction stream and three instruction stream was conducted. Over 99% accuracy in instruction identification despite adjacent instruction interference in the pipeline is achieved. Since this EM Disassembler can operate at distance DUT without needing any tampering of the device for side-channel taps, it opens up many more possibilities in code reverse engineering and IoT security.

15. FUTURE WORK

The present electromagnetic spectrum based instruction disassembly showed promise in predicting and identifying instructions in two instruction stream in a 2-stage pipeline. In future, we expect to do an exhaustive evaluation of its performance for entire Instruction Set Architecture. In addition, another future direction of research is to evaluate performance for cascaded instructions of more than two classes in flight.

16. ACKNOWLEDGMENTS

The work was supported in part by DARPA/AFRL through Contract FA8750-20-1-0504.

17. REFERENCES

[1] Abadi, M., Budiu, M., Erlingsson, U., and Ligatti, J. 2009. Control-flow integrity: principles, implementations, and applications. ACM Transactions on Information and System Security, Vol. 13, 1 (Oct. 2009), 4:1-4:40.

[2] Davi, L., Dmitrienko, A., Egele, M., Fischer, T., Holz, T., Hund, R., Nurnberger, S., and Sadeghi, A.-R..2012. MoCFI: A framework to mitigate control-flow attacks on smartphones. In Network and Distributed System Security Symposium (NDSS) (San Diego, CA, Feb. 2012).

[3] Wartell, R., Mohan, V., Hamlen, K., and Lin, Z. 2012. Securing untrusted code via compiler-agnostic binary rewriting. In Proceedings of the 28th Annual Computer

Security Applications Conference (ACSAC'12) (Orlando, FL, Dec. 2012). 299–308.

[4] Chen, X., Slowinska, A., Andriesse, D., Bos, H., and Giuffrida, C. 2015. Stackarmor: Comprehensive protection from stack-based memory error vulnerabilities for binaries. In Network and Distributed System Security Symposium (San Diego, CA, Feb. 2015). 8-11.

[5] Van der Veen, V., Andriesse, D., Goktas, E., Gras, B., Sambuc, L., Slowinska, A., Bos, H., and Giuffrid, C. 2015. Practical context-sensitive cfi. In Proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS), (Denver, Colorado, USA, October 12–16, 2015). ACM, New York, USA. 927–940.

[6] Van der Veen, V., Goktas, E., Contag, M., Pawlowski, A., Chen, X., Rawat, S., Bos, H., Holz, T., Athanasopoulos, E., and Giuffrida, C. 2016. A tough call: Mitigating advanced code-reuse attacks at the binary level. In Proceedings of the 37th IEEE Symposium on Security and Privacy (Oakland) (San Jose, CA, USA, 22-26 May 2016). IEEE, USA. 934-953.

[7] Chen, X., Bos, H., and Giuffrida, C. 2017. CodeArmor: Virtualizing the Code Space to Counter Disclosure Attacks. In Proceedings of 2017 IEEE European Symposium on Security and Privacy (Paris, France, 26-28 Apr. 2017). IEEE, USA. 514-529.

[8] Sæbjørnsen, A., Willcock, J., Panas, T., Quinlan, D., and Su, Z. 2009. Detecting code clones in binary executables," in Proceedings of the Eighteenth International Symposium on Software Testing and Analysis (Chicago, IL, USA, July, 2009). ACM, New York, USA. 117–128.

[9] Gopan, D., Driscoll, E., Nguyen, D., Naydich, D., Loginov, A, and Melski, D. 2015. Data-delineation in software binaries and its application to buffer-overrun discovery. In Proceedings of the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (Florence, Italy, 16-24 May 2015). IEEE, USA. 145-155.

[10] Tilevich, E., and Smaragdakis, Y. 2015. Binary refactoring: Improving code behind the scenes. In Proceedings of the 27th International Conference on Software Engineering (ICSE'05) (St. Louis, Missouri, USA, May 15-21, 2005). 264-273.

[11] Ericsson. 2018. November 2018, the connected future - internet of things forecast, May 2018.

[12] Bilodeau, O., and Dupuy, T. 2015. Dissecting Linux/Moose the Analysis of a Linux Router-based Worm Hungry for Social Networks. Technical Report. May, 2015.

[13] Park, J., and Tyagi, A. 2017. Using power clues to hack IoT devices: The power side channel provides for instruction-level disassembly. IEEE Consumer Electronics Magazine. 6, 3, (Jul. 2017), 92–102.

[14] Whiteside, H. and King. R. 1964. The loop antenna as a probe. IEEE Transactions on Antenna and Propagation, 12, 3, (May 1964). 291-297

[15] Shwartz, S.S., David, S, B. 2014. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, UK

[16] Kim, H.-C., and Ghahramani, Z. 2003. The EM-EP Algorithm for Gaussian Process Classification. In Proceedings of Workshop on Probabilistic Graphical Models for Classification (ECML), (2003). 1-8

[17] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., et al., "Recent advances in convolutional neural networks," Pattern Recognition,Vol. 77, pp. 354–77, Oct. 2017