Current Events: Identifying Webpages by Tapping the Electrical Outlet

Shane S. Clark[1], Hossen Mustafa[2], Benjamin Ransford[3], Jacob Sorber[4], Kevin Fu[5], and Wenyuan Xu[⋆⋆2,6]

[1]University of Massachusetts Amherst [2]University of South Carolina
[3]University of Washington [4]Clemson University [5]University of Michigan
[6]Zhejiang University

Abstract. Computers plugged into power outlets leak identifiable information by drawing variable amounts of power when performing different tasks. This work examines the extent to which this side channel leaks private information about web browsing to an observer taking measurements at the power outlet. Using direct measurements of AC power consumption with an instrumented outlet, we construct a classifier that correctly identifies unlabeled power traces of webpage activity from a set of 51 candidates with 99% precision and 99% recall. The classifier rejects samples of 441 pages outside the corpus with a false-positive rate of less than 2%. It is also robust to a number of variations in webpage loading conditions, including encryption. When trained on power traces from two computers loading the same webpage, the classifier correctly labels further traces of that webpage from either computer. We identify several reasons for this consistently recognizable power consumption, including system calls, and propose countermeasures to limit the leakage of private information. Characterizing the AC power side channel may help lead to practical countermeasures that protect user privacy from an untrustworthy power infrastructure.

1 Introduction

Computer users commonly assume that software mechanisms, such as in-browser encryption, protect their private information. Research on side channels has challenged this assumption by showing that computer components such as the CPU [18] and the keyboard [32] can leak private information. Along the same lines, this paper examines the feasibility of inferring private information from a general-purpose computer's AC power consumption, despite significant additive noise from the power grid [6].

Past work has exploited AC power side channels for information leakage, but at the level of an entire household [24] or a device with a constrained state space [8, 6]. For example, a television that is dedicated to displaying videos produces relatively consistent power consumption over multiple plays of the same video. Given a small number of candidate videos, it is possible to identify which of them is playing [8]. A general-purpose computer, on the other hand, exhibits a tremendous state space because of its practically unconstrained operation. Executing the same computing task at

⋆⋆ Corresponding faculty author

Published in Proceedings of European Symposium on Research in Computer Security, 2013

different times may result in different power consumption patterns because of different background tasks or I/O workloads (e.g., network activity). Nevertheless, we find that system-wide traces of AC power consumption leak enough information about the operation of a general-purpose computer to identify the webpage that the computer is loading (out of a set of known pages). Because browsers use a diverse subset of the available hardware components, our results suggest that this technique may generalize to other computing workloads.

Several factors work to our advantage. Web browsers increasingly take advantage of hardware to improve the user's experience (by, e.g., executing native code [35]), resulting in resource consumption that scales with the webpage's complexity. Another factor is that modern computers and operating systems aggressively try to reduce power consumption [30], resulting in energy-proportional computing in which the power consumption tightly fits the workload [6]. Both of these factors increase the system's dynamic range, which in turn increases the information available in power traces.

There are also challenges to the task of identifying webpages from AC power consumption. The fundamental challenge is separating interesting activity from baseline power consumption, which a computer's power supply aggregates. Other challenges stem from the dynamic nature of the Internet and modern websites. The round trip time for fetching webpages may change over time; many websites include scripts that run long after the page loads, and many customize content for each visitor.

This paper's contribution is an attack on privacy that identifies specific web-browsing activities via an AC power side channel. We characterize and measure this side channel by designing methods to extract patterns of power consumption as a computer loads a webpage. These patterns, which are obscure in the time domain but more apparent in the frequency domain, act as power signatures that allow an eavesdropper to determine which webpage is being loaded. Additionally, these power signatures are robust against a variety of changes to the computing environment, including background processes, changes in network location, the use of a VPN, or even the use of a different computer. Because most of the identifiable information occurs at under 10 kHz in the frequency domain, capturing and exfiltrating power measurements is within reason for a simple embedded device that could fit discreetly inside a power outlet.

Using a covertly modified electrical outlet to record power consumption, we trained and tested a classifier with over 100 hours of traces representing more than $13,000$ page loads from a set of 51 webpages from sites representing over 30% of global page views. Given a power trace with 51 possible labels, the classifier identified the correct match from the training set with 99% precision (resistance to false positives) and 99% recall (resistance to false negatives). Given an unlabeled trace from one of 441 webpages *not* in the training set, the classifier's false positive rate is less than 2%. In some cases, the classifier succumbs to overfitting when trained on traces from a single computer, confusing other consistent power activity for browsing activity. However, when trained on traces from two computers, the classifier identifies the common patterns that are due to browser activity and can correctly label unseen traces from either computer.

This work conceptually bridges the gap between previous work on circuit-level direct-current (DC) power analysis [18] and coarser-grained, household-level activity

recognition via AC power measurements [12, 26, 11]. This paper also proposes several hardware and software countermeasures to minimize information leakage.

Threat model. Focusing on the AC power side channel, this paper considers an attacker with physical access to a power outlet the victim might use. Possible easy targets include outlets in coffee shops and airports. A simple modification to a power outlet enables discreet data recording. Because most users implicitly trust power outlets, an attacker may gain easy, *persistent* access to a victim's power-consumption patterns.

2 Background and Challenges

This section distinguishes AC power measurements from DC power measurements, which have been studied extensively in literature about side channels [18, 4]. It breaks down the power budget of a computer and discusses how the actions of a web browser influence power consumption. Finally, it explains some of the challenges inherent in our attempts to classify AC power traces.

2.1 AC Versus DC Power Traces

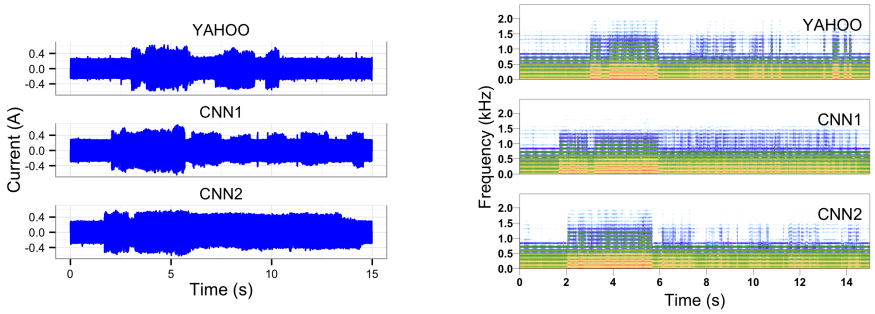
DC power measurements explored in previous research involve tracing individual components' power consumption on a circuit board. Such techniques require access to internal hardware elements and are overly intrusive from the viewpoint of our threat model. An attacker conforming to our threat model seeks a power-analysis technique that is relatively nonintrusive and does not involve opening or modifying the victim's computer. Every computer operates on power drawn from a power grid. A laptop user may avoid drawing power from the grid by relying on the battery, but this is a temporary solution. Monitoring AC power consumption affords a *system-wide* view of the computer's activity, although individual activities may be difficult to identify because all components' signals are added together.

Challenge: Noise. Both the "upstream" (e.g., household) circuit and the power supply itself introduce noise onto the power line. Since this noise is omnipresent, we make no attempts to remove it, unlike previous work [8]; instead, we provide noisy inputs to classifiers and rely on the presence of a non-noise signal to influence classification. Eschewing filtering also simplifies the hardware we use to gather traces.

Challenge: Periodicity. Whereas DC signals often feature prominent level shifts and other artifacts that are amenable to time-domain analysis, the *alternating* nature of AC current essentially convolves sets of sinusoids, making time-domain analysis difficult. We therefore perform analysis in the frequency domain. Before classification, we transform traces into the frequency domain (Figure 1b) using the Fourier transform. In addition to making certain signals easier to identify (e.g., 60 Hz utility power in the U.S.), this approach enables meaningful comparisons despite misaligned traces, or traces of different lengths, and the additive nature of the SMPS's power consumption preserves frequency information from individual components' power consumption.

2.2 Tracking Hardware Components' Power Consumption

To develop intuition about what kinds of tasks are likely to induce identifiable activity patterns on the power line, we measured the power consumption of a laptop (MacBook-1, Appendix A) under a variety of workloads designed to stress individual subsystems.



(a) Time-domain plots **(b)** Spectrogram plots

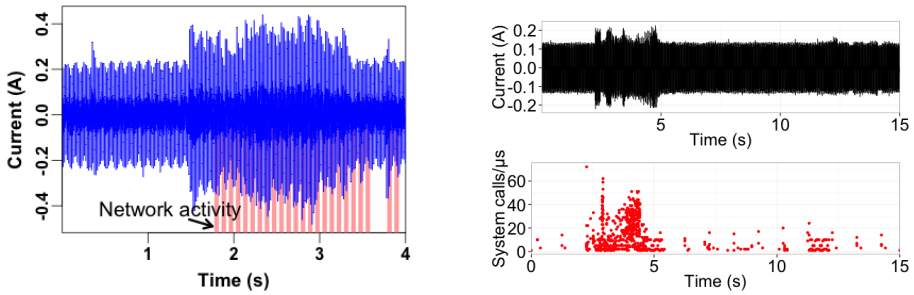
Fig. 1: Time- and frequency-domain plots of several power traces as a MacBook loads two different pages. In the frequency domain, brighter colors represent more energy at a given frequency. Despite the lack of obviously characteristic information in the time domain, the classifier correctly identifies all of the above traces.

We used a P3 Kill A Watt power monitor [25] to measure power consumption. Table 1 summarizes the results, which suggest that the MacBook's CPU and GPU dominate power consumption under load. The network interfaces and solid-state storage draw comparatively little power.

Prior side-channel work has leveraged network characteristics such as packet timings [28] or lengths [33, 34] to classify webpages according to their network traffic. A reasonable question to ask is whether network classifiers are likely to apply to the problem of webpage identification. An experiment suggests that AC power traces in fact do not map exactly onto network traffic. We tapped the activity LED of a network switch port to capture a representation of a computer's network traffic while also tracing the computer's AC power line. Figure 2a shows an example from our tests. The computer consumes power performing other tasks before the network interface actually begins to

Condition	**Power (W) vs. Baseline**
Baseline (idle, screen off)	**8**
One core at 100%	+7
Two cores at 100%	+11
GPU at 100%	+11
Wired network saturated	+2
Wireless network saturated	+3
File copy, SSD to SSD	+6
Screen at maximum brightness	+6

Table 1: MacBook power consumption under various types of load. Numbers beginning with + are relative to the baseline of 8 W.



(a) The network activity is correlated with high current consumption, but is not the only cause. Spikes before and after network activity show that local computation dominates the consumption.

(b) The system call activity (as measured by DTrace) is also correlated with high current consumption, and our results suggest that systems exercised by system calls are a major cause of consumption.

Fig. 2: Time-domain plots as a MacBook loads webpages. Both network activity and system calls appear to correlate with energy consumption.

send and receive packets. Furthermore, the AC power provides insight into client scripts and rendering loads unavailable in a network trace.

Power consumption appears to be more strongly correlated with system calls than with network activity as shown by Figure 2b. Tracking the number of system calls initiated by the browser process with DTrace captures memory allocation and disk I/O in addition to network activity, enabling monitoring of all of the components we have identified as major power consumers.

3 Approach: Supervised Learning Classifier

To distinguish among webpages, we adopt a supervised learning approach, in which we train a classifier on labeled AC power traces and then attempt to match unlabeled traces. An AC power trace contains artifacts of every powered computer component, each of which may have its own clock rate or power signature, and each of which processes information differently. We assume that disentangling these signals (multicore CPU, multicore video card, multiple drives, etc.) from a single AC power trace is intractable with current techniques, and instead focus on coarser-grained, *system-level* questions, such as which popular webpage the user is loading. Because it is prohibitively difficult to build a generative model of how computing tasks will map to power consumption—a discriminative modeling approach is more appropriate. Our supervised-learning approach fits this requirement; it requires only a labeled training set and learns its own model of how feature values map to class labels. Specifically, we train *support vector machines* (SVMs) using the open-source library libsvm [5].

3.1 Feature selection

Classification requires extracting feature vectors on which to train a classifier. A naïve classifier might simply consider the *length* feature of a sample in the time domain, defined as the length of time for which power consumption remains above a predetermined

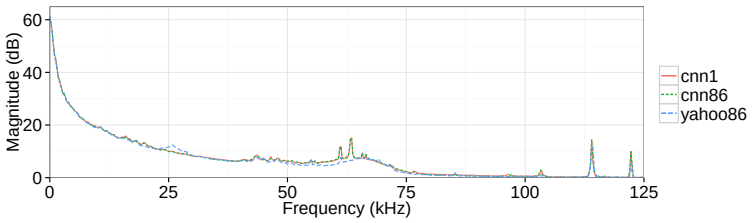


Fig. 3: Plots of three of our Fourier transform feature vectors. While the pages are difficult to separate visually in the time domain, the two `cnn.com` samples are indistinguishable to the eye in the frequency domain, whereas `yahoo.com` diverges around 25 and 65 kHz.

threshold. However, background tasks add confounding noise in the time domain, which may obscure the true endpoints of a specific task, and tasks often include periods of both high and low power consumption. Mean, minimum, and maximum power are equally unsuitable choices for similar reasons. A more robust approach is to classify traces based on features from the frequency domain.

We transform traces into the frequency domain by first calculating the spectrogram using rectangular sliding windows 1000 samples wide with 50% overlap. We then collapse the spectrogram into a single Fourier transform by summing over all of the time steps. As a base set of features for classification, we divide the Fourier transform of each power trace into 500 segments, each 250 Hz wide, starting at 0–250 Hz and ending at 124.75–125 kHz, half the 250 kHz sample rate at which we recorded traces. This process yields 500 features, each of which represents the power present within one 250 Hz slice of spectrum over the duration of the trace.

As described in Section 2.1, classifying in the frequency domain allows meaningful comparisons between misaligned traces or traces of different lengths. Using the output directly from the Fourier transform as a feature vector is a simple approach and yields excellent results. It is also straightforward to visualize differences in the Fourier transform, as shown in Figure 3. Plotting the feature vector reveals consistent features between the two traces of `cnn.com` and recognizable differences between a trace of `yahoo.com` and the `cnn.com` traces, which are not obvious in the time domain.

3.2 Classification

We train a binary classifier for each page in our corpus. After training all 51 SVMs, we use each of them to classify test samples. A test sample is an *unlabeled* 500-dimensional feature vector, obtained in the same way as the training samples, that is *not* in the training set. Each SVM determines whether the test sample was an instance of the webpage it was trained to recognize. In the interest of simplicity, we do not implement a multi-class labeling solution in which all 51 SVMs collectively generate a single output, but there are a variety of well-studied techniques for this purpose and libsvm implements several of them [15].

There are three notable details of the training process. First, we linearly normalize feature values across all samples. This prevents features with large values (e.g., the

low frequency elements) from dominating features with smaller values. Second, we use standard 10-fold cross-validation to avoid overfitting at training time. By repeatedly splitting the training set and retraining each time, the classifier avoids the possibility of biases in a small number of training examples producing a biased model. Finally, we use a radial basis function (RBF) kernel, as recommended by the authors of libsvm [5].

4 Methods and Metrics

To cover a diverse set of webpages representing typical Internet traffic, we chose 48 webpages drawn from Alexa's list of the top $1,000,000$ websites [2], discarding duplicates and adult websites. By Alexa's estimates, these top 48 websites represent over 30% of global page views. We added the top Google result for "cheap Viagra" as an example of a potentially embarrassing (or malicious) page. To include a page that loads with negligible latency, we added two authors' department's home pages, a < 1 ms round trip from one of our measurement points, bringing the number of webpages in our training set to 51.

The Alexa rankings list *websites*, but it is more meaningful to collect traces of individual *webpages*. Each of our traces represents an automated load of the front page of one of the 51 websites. To record realistic power traces of user browsing, we used a custom Chrome extension (see §4.1) to collect at least 90 consecutive traces of each page. For webpages that require users to log in before displaying useful information, we logged in as the first or second author. We believe that our choice to consider front pages is reasonable because users are likely to visit the front page of a given website and then follow links to other pages. The notable exceptions to this tendency are bookmarked pages and direct links from other people or sites.

Evaluation metrics. Because our classifier uses standard machine-learning techniques, we use standard metrics from machine learning to evaluate its performance. In the following definitions, *tp* and *tn* refer to true positives and true negatives (correct labelings), and *fp* and *fn* refer to false positives and false negatives (incorrect labelings).

Precision, $tp/(tp+fp)$, is the fraction of positively labeled examples whose labels are correct. It measures the classifier's ability to exclude negative examples.

Recall, $tp/(tp+fn)$, is the fraction of all the examples that *should* have been positively labeled that *are* correctly positively labeled. It measures the classifier's ability to identify positive examples.

We present experimental results in terms of *precision* and *recall* because the standard *accuracy* metric is often misleading. In most of our experiments, the number of negative examples is roughly 50 times the number of positive examples because, for each webpage, there are more traces of *other* webpages in the testing set than there are of that webpage. Because of this disparity between positive and negative examples, the classifier could achieve greater than 98% accuracy by simply classifying all examples as negative. A perfect classifier would achieve 100% accuracy, 100% precision, and 100% recall. For any imperfect classifier, there is a tradeoff between precision and recall.

4.1 Experimental Setup

This section describes the experimental setup we used to capture AC power traces.[1]

Safety note: This paper is not a complete manual for electrical safety. Measuring "hot" terminals is potentially fatally dangerous and should be conducted only under qualified supervision. Do not try this in a non-laboratory setting.

Using an instrumented outlet (described next), we measured the power consumption of two Apple MacBook computers, a Lenovo T410 laptop, and a Dell Vostro desktop PC, all running different operating system versions. (For detailed hardware and software specifications, see Appendix A.) To approximate the environments of typical users, we used a stock installation of each operating system. In particular, we allowed default background processes to run. Experiments with the two MacBook computers were carried out approximately one year apart using similar but non-identical instrumented outlets.

To record each workload's power signature, we monitored electrical current between the power supply and an instrumented outlet. A modern AC outlet has three terminals: *hot*, *neutral*, and *ground*. To measure a power supply's instantaneous current on the hot–neutral circuit, we placed a 0.1 Ω *sense resistor* (part #13FR100E-ND) in series with one terminal of a standard outlet. We attached an Agilent U2356A data acquisition unit (DAQ) to the terminals of the sense resistor. The DAQ samples the voltage across its probes and sends the data via USB to another PC (not the computer being measured). We recorded 16-bit samples at a rate of 250 kHz to capture workload artifacts occurring at up to 125 kHz.

Finally, we developed a Chrome extension to automate the repeated loading of a target webpage. The extension repeatedly: opens a new window, pauses, loads the page, pauses again, and finally closes the window. For webpages that did not require user credentials, the script opened browser windows in a *private browsing* mode to purge the browser environment of confounding data. To compare webpage identifiability across browsers, we also used the iMacros extension for Firefox [1] to mimic our Chrome extension. We recorded continuously with the DAQ while running experiments. A script with knowledge of the browser extensions' timings chopped the DAQ's output into separate trace files to be used with our classifier. While the majority of the webpages we profiled show no changes within the measurement period, there are notable exceptions. A number of high-turnover webpages including `cnn.com`, `cnet.com`, and `reddit.com` underwent content changes during our measurements.

5 Evaluation

This section summarizes our experimental results over a wide range of conditions. While there are a limitless number of questions to ask about how well a classifier works under different conditions, we have distilled them down to the following six questions regarding causality and intuition:

[1] We use the terms *power trace*, *voltage trace*, and *current trace* interchangeably. What we actually record is a *voltage trace* that maps trivially to current ($I_{sense} = V_{sense}/R_{sense}$, with R_{sense} constant) and therefore power ($P = I_{sense}V_{RMS}$, with V_{RMS} a constant 120 volts (RMS) in North America).

- How effectively can the SVM classifier differentiate webpages from one another? (Section 5.1)
- How robust is the classifier in the presence of content distribution services, anonymity services, encryption, and caching, as well as changes in network location, type, or interface? (Section 5.2)
- How is classifier performance affected by changes in operating system or hardware? (Section 5.3)
- How does the classifier's performance change when the test traces include background activities? (Section 5.4)
- How does sampling rate affect classification? (Section 5.5)
- How well does the classifier exclude samples of pages outside the corpus? (Section 5.6)

We find that our classifier can differentiate webpages with high precision and recall rates (each averaging 99%) and that it is robust against many of the variations we tested, including the use of a VPN, and changes in the location or network interface. It is not robust against changes of machine or operating system. Where our classifier performs poorly, we find in most cases that increasing the diversity of the training set improves its performance along all metrics. The total number of power traces we tested across all experiments was $13,110$, chronicling over 100 hours of 250 kHz trace recordings.

5.1 Page differentiation

Our SVM classifier effectively differentiates among the 51 popular webpages we tested. As a baseline for classifier performance, we varied only the webpage under test and held all other variables constant. These other variables include machine under test, network location, network interface, operating system, and web browser. By varying only the webpage under test, we minimize differences that are not actually the result of variation among webpages.

We gathered all of the data for this experiment twice, using slightly different MacBooks and instrumented outlets built to the same specifications in different locations. Here we present the results of combining the two data sets into a single corpus. After gathering $\sim$90 traces for each of the 51 webpages on each of the MacBooks (for a total of $\sim$180 traces), we used the experimental protocol described in Section 3.2 to label each trace. Each SVM labels a trace as either matching or not matching the page for which it was trained. The total size of the corpus for this experiment was $9,240$ traces. We used half of these traces for training and the remaining traces for testing. With $\sim$90 training examples per label, the SVM classifier achieves an average 99% precision and 99% recall over all webpages in the data set.

The classifier's performance varied among the tested webpages. The precision and recall were both 100% for 18 of the 51 webpages. The lowest precision for any page was 93% for `skype.com` and the lowest recall for any page was 88% for `slashdot.org`. A plausible explanation for the classifier's relatively poor performance on `skype.com` is that the front page underwent a significant design change between the capture of our two data sets, which we confirmed by inspecting snapshots from the Internet Archive Wayback Machine [16]. The poor recall for `slashdot.org` could be explained by the high turnover of the front page or inconsistent load times. `godaddy.com`, which uses a commercial content distribution service, also yielded lower recall results.

5.2 Diverse browsing conditions

We varied the conditions under which our browser operated and found that the SVM classifier is robust against local network connection type, use of cache, VPN encryption, and the passage of time for most webpages. It is not robust against the use of a caching content-distribution network (CDN) such as Coral [10].

Our training and testing setup was as follows. We repeated this process for three webpages: one simple page (`google.com`) and two complex pages (`cnn.com` and `cnet.com`). For each page, we gathered the following sets of traces on one of our MacBooks:

- **Time:** Traces gathered a month later, to test how fresh the training set must be.
- **Cache:** Traces recorded with a warm browser cache, to test whether the classifier depends on specific network traffic patterns.
- **VPN:** Traces recorded while connected to a VPN concentrator a 1.5 ms round trip away (essentially measuring only cryptographic overhead), to test whether encrypting normal traffic would be an effective countermeasure.
- **WiFi:** Traces recorded while connected to our lab network wirelessly instead of via wired Ethernet, to test whether the training phase overfits the SVMs to "clean" low-latency wired traffic.
- **CDN:** Traces recorded with web traffic passing through the Coral CDN, to test whether a caching proxy sufficiently disguises traffic.

To test each one of these sets, we trained an SVM on all of the *other* sets using only samples from the same MacBook. For example, to study whether the SVM could correctly label traces in the *WiFi* set, we trained the SVM on the *Time*, *Cache*, *VPN*, and *CDN* sets in addition to the *Base* set of samples. In contrast to training only on the *Base* set, this training approach avoids overfitting the SVM to that set. After training the classifier, we instructed it to classify the traces in the test set.

The only condition that hurt performance, for two of the three webpages, was the use of the Coral CDN. For `cnn.com`, the classifier incorrectly labeled all traces from the *Coral* set as negatives; for `google.com`, the classifier incorrectly labeled 45 of 50 traces as negatives, resulting in a 10% recall rate. The classifier's poor performance on Coralized pages illustrates that, while the network interface's power consumption may not uniquely determine how a trace is classified, the network interface may still alter the timing of characteristic consumption patterns for *downstream* devices such as the CPU and GPU that act on its outputs. Coralizing `cnet.com` likely made little difference in its classifiability because `cnet.com` is already distributed via a commercial CDN.

The classifier's performance on traces from the *VPN* set deserve special attention. It suggests that encryption and decryption, at least as implemented by our MacBook's PPTP VPN, have little effect on power-trace classification—i.e., the SVM classifier is robust against VPN encryption. Our MacBook's processor does not include hardware support for the AES-NI instruction set, which is designed to improve AES performance [27]. With AES-NI support, performance and energy efficiency should both be improved, reducing the encryption and decryption impact even further.

Network location. The Coral results suggest that changes in latency or throughput alter packet arrival times enough to thwart the classifier. To test this hypothesis, we created two static pages that do not have any running scripts or asynchronous content and

gathered traces of one MacBook loading each page in different locations. One location had a residential connection and the other a university research network connection. We then trained on all pages in the corpus, including samples of the static pages using only one of the two locations. We tested on samples from the untrained location.

When trained on the residential connection and tested on the university connection, the classifier's precision and recall were 100% and 73% respectively. This result shows that the training did not lead to any false negatives for other pages, but was not able to identify all samples. When we reversed the training and testing locations, the precision and recall were both 100%. This experiment demonstrates that it is not always necessary to use or simulate a potential victim's connection to train an effective classifier, but that the network connection's impact is determined largely by page content.

Tor. Based on the classifier's reduced performance when using Coral, we evaluated Tor [7] configured as a local SOCKS proxy. While Coral introduces higher latencies and less consistent throughput, Tor adds additional complications. Not only does the Tor network add latency and reduce throughput, but Tor includes encryption and decryption operations and the exit node changes unpredictably, which can change the language or content of some pages. We gathered 91 traces of MacBook-1 loading `google.com` and tested them against an SVM trained on samples from all of the other browsing conditions. Tor proved to be a very effective countermeasure, with only 2 of the 91 Tor samples correctly identified, yielding 2% recall.

5.3 Operating system and machine diversity

Operating system. Training on traces from a single operating system limits the classifier's effectiveness to that operating system. However, training on traces from two operating systems allows the classifier to correctly identify both OSes. To test this behavior, we gathered traces of `google.com` and `cnn.com` using Windows 7 and Linux (Ubuntu 10.04) on the desktop PC. For both sets of traces, we used the same versions of the Chrome browser and our custom Chrome extension for test automation. When trained only on examples from one operating system, the classifier failed to correctly label traces from the other. The only exception was a single trace of `cnn.com` loaded from Linux, which the classifier identified correctly despite the having been trained only on examples from Windows 7. When we rearranged the input sets so that each contained an equal number of traces from each OS, then trained on one of these mixed sets, the classifier *correctly labeled* all unlabeled traces from the other mixed set.

Differences among OSes include system timers, drivers, memory management, GUI characteristics, and performance tuning. All of these differences may play roles in differentiating power-consumption patterns. The above result suggests that a prospective attacker should collect traces under as many different operating systems as possible.

Machine diversity. When we varied both machine *and* operating system, the SVM classifier failed to correctly label any traces. We trained an SVM on the MacBook (running Mac OS 10.7.3) with 50 webpages and tested on the Lenovo laptop (running Windows 7) for 5 webpages (`google.com`, `cnn.com`, `espn.com`, `live.com`, and `youtube.com`). Then switched the roles and trained and tested again. For all webpages, the SVM failed to correctly label traces from one machine when trained only on examples from the other. The precision and recall never exceeded 10%.

Training on examples from *both* machines allowed the SVM to classify traces from both machines accurately: 98.4% precision and 98.2% recall on average for the 5 webpages. This result suggests that, as in the operating system experiment, the problem lies in the lack of training diversity. In the future, we intend to test this hypothesis by training an SVM on a small, but diverse, set of machines and then testing traces from machines that are not represented in the training set.

5.4 Background activities

Noting the tendency of users to browse multiple webpages at the same time and running background processes, we measured the classifier's sensitivity to background noise. We randomly chose one of the 51 webpages in our training set—`live.com` —and loaded it with combinations of background processes. We collected traces for `live.com` on a MacBook when a combination of the following 4 processes were running: `gmail.com`, iTunes radio, `pandora.com`, and a word processor. We collected traces for 8 combinations in total, e.g., only `gmail.com`; `gmail.com`, `pandora.com`, and word processor together, etc. We trained the SVM with examples for all 51 webpages without any background process and tested it using the background examples. In all cases, the classifier was able to classify `live.com` accurately with 100% precision and 100% recall.

Even though we only tested one webpage and a small set of background processes, the result suggests that the classifier can be robust against combinations of background processes. A possible explanation for the classifier's robustness is that the background processes do not saturate the CPU load and have little effect on the GPU load because they are literally in the background and do not draw to the screen. Quantifying the limits of this robustness will require further investigation.

5.5 Sampling rate differentiation

Decreasing the sampling rate at which an instrumented outlet records voltage would allow for tracing and exfiltration using simple, low-cost hardware. To understand how robust the classifier is to changes in sampling rate, we repeated the set of page differentiation tests, but simulated lower sampling rates by restricting the set of input features to those representing lower-frequency components. Figure 4 compares the results with the original sampling rate against results with simulated lower sampling rates. Each reduction in sampling rate is by a factor of two.

Reducing the sampling rate by a factor of more than 30 (from 250 kHz to 7.8 kHz) incurs only a 9% reduction in average precision and recall. These results show that the lower frequency bands alone contain enough information to accurately classify webpages. An attacker could likely produce a compact and inexpensive measurement device capable of mounting successful attacks.

5.6 Exclusion of unknown pages

Our classifier reliably identifies pages appearing in the training set, but a practical attack would require the classifier to reject pages *not* appearing in the training set as well. With training and testing sets that resembled each other, a classifier could perform equally well in the previous experiments whether it learned to cluster positive or negative examples. To test the hypothesis that the SVMs learned to cluster only *negative* examples

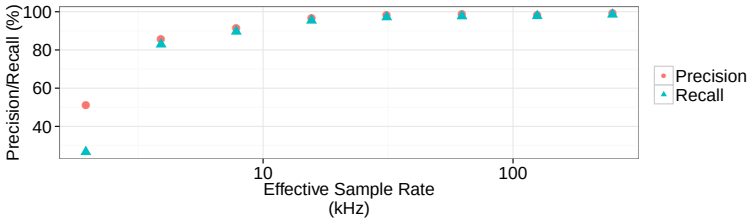


Fig. 4: The average precision and recall across all 51 pages with exponentially increasing sample rate. The classifier's performance decreases with sampling rate, but the precision and recall do not drop below 90% until the sampling rate is less than 4 kHz, a 60x reduction.

during training, we tested them with a set of previously unseen webpage samples that were not in the training set.

We gathered one trace from each of 441 webpages randomly selected from a list of 1 million popular pages published by Alexa [2], making sure to remove pages that were already in the training set. We then tested all 441 pages against all 51 trained SVMs and measured their false-positive rates. The total false positive rate over all classifiers was 1.6%, leading us to reject the above hypothesis and conclude that the SVMs correctly learned to cluster positive examples.

6 Countermeasures to Limit Information Leakage

This section sketches several countermeasures to mitigate the threats described in Section 1. Hardware and software countermeasures both present inherent tradeoffs. Hardware mechanisms that increase design complexity or cost may not find traction with high-volume manufacturers. Software countermeasures that increase computational work may vitiate energy-efficiency measures. Altering workloads to disguise activity may negatively affect usability or user satisfaction. Effective yet usable countermeasures remain an open problem.

Software countermeasure: Delays and throttling. The classifier's poor performance with traces gathered using Coral or Tor suggests that delays complicate classification. Tor in particular is a very effective countermeasure according to our experiments. For users unable or unwilling to use Tor, it may be possible to leverage similar changes in activity patterns. A defensive idea from Song et al. [28] is to introduce random delays in the time domain, which will cause changes in the frequency domain that may confuse our classifier. The problem with random delays, as Song et al. point out, is that different instances of the same private signal, with different random delays added to each, give an attacker enough information to learn the true timing of the signal by simply averaging the delays. The same problem afflicts the defensive strategy of randomizing the order in which the browser loads page elements.

Hardware countermeasure: Current filtering. Filtering circuitry that damps current fluctuations could prevent workload-dependent information from leaking onto the AC power line. SMPSes internally implement low-pass filters to remove high-frequency

noise and meet government EMI standards [9]. Our experiments reveal that, for the SMPSes we tested, the frequencies useful for classification are below the internal filter's cutoff. A more aggressive low-pass filter or a high-pass filter could remove additional information, but would likely increase the cost and physical size of an SMPS. Our sampling rate experiments show that the classifier is effective until the maximum observable frequency drops below 4 kHz, so much more filtering would likely be required.

7 Related Work

AC power event recognition. This work focuses on classifying run-time events on the order of seconds on a general-purpose computer, in contrast to previous work that measured on–off transitions at a large granularity from a household vantage point. Unclassified research on recognizing activity by measuring AC power goes back to at least 1989, when Hart proposed *nonintrusive load monitoring* (NILM) to map changes in total household power consumption to appliance activations [12, 13]. Hart also recognized the potential for abuse of NILM techniques. Recently, Gupta et al. proposed ElectriSense, a nonintrusive system that uses a single-point monitor to detect electromagnetic interference (EMI) generated by consumer electronics' switched-mode power supplies [26, 11]. Analyzing frequency-domain power signatures, ElectriSense advanced prior work by detecting nearly simultaneous device activation. Both NILM and ElectriSense effectively capture and identify *on* and *off* events at the device level, but neither aims to infer the internal states of integrated commodity devices such as personal computers, as our work does.

Enev et al. refined the ElectriSense concept by studying the correlation of EMI with video signals being played on modern high-definition televisions [8]. They classified signals (video segments) more than 15 minutes long. In comparison, we focus on classifying signals containing shorter periods of activity and monitor comparatively complex general-purpose hardware. Our sensing apparatus is also somewhat simpler, relying on a single sense resistor and no hardware filtering.

Our prior work has argued that ever-increasing energy proportionality has positive and negative consequences for security and privacy and presented preliminary webpage classification results with a smaller corpus [6]. This work instead focuses specifically on attacks against user privacy and presents a much more in-depth empirical analysis.

Network traffic analysis. Past work has, like ours, exploited side channels to learn sensitive information from traffic that may be encrypted. From previous work we borrow the intuition that webpages induce characteristic activity patterns that are robust against encryption and the passage of time. Several researchers have trained classifiers on encrypted or obfuscated web traffic and observed that they could match webpages against their training set using only packet-length information [14, 22, 23, 29]. Our classifier uses AC power traces as input rather than network traces, and so observes a noisier, aggregate side channel.

Parasitic modulation. Our work focuses on leakage via a wired channel, unlike many past works that focus on leakage via parasitic modulation. Looking at CRT monitors, van Eck published the first unclassified side channel analysis work, demonstrating that the screen image could be reconstructed remotely using a TV receiver and hand-tuned

oscillators [31]. Kuhn further analyzed leakage from CRT and LCD monitors based on parasitic modulation [19–21]. More recently, Vuagnoux and Pasini also investigated leakage via parasitic modulation, though they targeted keyboards rather than monitors and detached their laptop power supplies to avoid interference [32]. Barisani and Bianco independently demonstrated keystroke recovery for PS/2 keyboards by attaching a resistor to the AC power cable, as in our work. They focus only on information from the keyboard and rely on the observation of high-speed switching specified by the PS/2 protocol [3].

DC power analysis. Our methods are *not* designed to find key material, unlike past work studying DC circuits that required pin-level access to components or detailed knowledge of the circuits under test. Kocher et al. summarize much of the abundant research on timing and power side channels [17, 18]. The most straightforward of these attacks measures a small portion of the complete system and uses domain knowledge to infer the information being processed. This type of attack requires physical access to the system, knowledge of the cryptosystem under attack, and thousands of accurate measurements of the same process.

8 Extensions and Future Work

Alternative tracing methods. In our experiments, we physically connect probes to an AC circuit to trace electrical activity. An ancillary goal of this work is to demonstrate that it is possible to covertly modify a power outlet, so physical contact with the computer's power cord is a reasonable expectation under our threat model. However, less-invasive methods exist to measure the current along the power cable. In particular, a *Hall effect sensor*, which measures current via the magnetic field around a wire, could provide a way to trace power consumption if modifying the outlet is infeasible. Such an eavesdropper could easily be removed when not in use. We have not tested our classifier against traces captured with a Hall effect sensor, but we have confirmed that Hall effect sensors matching our sense resistor's sensitivity exist.

Another possibility is indirect measurement similar to that of Enev et al. [8]: connecting measurement equipment in parallel with the victim on the same electrical circuit but on a different outlet. We expect classification performance to decline because of the higher noise floor, but measurements might reveal that traces from outside the victim's outlet are *qualitatively* good enough for an attacker to use.

Adding classification features. The current SVM classifier relies solely on a coarse-grained Fourier transform to learn unique webpage features. There are many promising extensions to the feature space that could improve classification performance. One simple extension would be to increase the resolution of the Fourier transforms used to train and test the classifier. Doing so would increase the dimensionality of the feature space, and possibly the classifier's ability to distinguish among webpages.

An extension that takes advantage of SMPS load characteristics would be to simultaneously sample both voltage and current. As Section A discusses, SMPSes are nonlinear loads that pull the voltage and current waveforms out of phase in a way that is related to the workload. The changing relationship between the voltage and current waveforms over time may reveal more information about the state of the system that is orthogonal to raw current consumption.

Detecting other activities. As we have emphasized in this work, computers are complex, general-purpose devices. The space of possible actions that a user might take is vast. While we have focused on web activity in order to address a well-defined, tractable problem, future work could address a broad range of other activities. Tasks as simple as counting keystrokes, regardless of whether different keys can be recognized, may reveal sensitive information. Song et al. have demonstrated that counting keystrokes can reduce the search space for brute-force password cracking by a factor of 50 [28].

9 Conclusions

This work demonstrates that a computer's AC power consumption reveals sensitive information about computing tasks, specifically the webpage that the computer is loading. We designed methods for webpage identification that extract power consumption signatures that are obscure in the time domain but more apparent in the frequency domain. With a data set of over $13,000$ power traces of 51 popular webpages, our trained classifier can correctly label unseen traces with 99% precision and 99% recall. The power trace signatures are robust against several variations including the use of an encrypting VPN, background processes, changes in network location, or even the use of a different computer. This is the first paper that quantifies the degree to which information about browsing activity leaks via the AC power supply. We believe it represents an early step in understanding this side channel. The increasing dynamic range of hardware power consumption will lead to further information leakage. Open research problems include the design and evaluation of countermeasures to mitigate the privacy risks of using untrusted power infrastructure.

Acknowledgments

We thank the members of the SPQR group and Erik Learned-Miller for discussions and feedback on drafts. This material is based upon work supported by the National Science Foundation under Grant No. GEO-1124657, CNS-0845671, CNS-0923313, CNS-1331652, and S121000000211. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This publication was made possible by Cooperative Agreement No. 90TR0003/01 from the Department of Health and Human Services. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the HHS.

References

1. iMacros for Firefox. `http://www.iopus.com/imacros/firefox/`, Loaded Sept. 2011.
2. Alexa Internet, Inc. Top 1,000,000 sites (updated daily). `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`, Loaded Feb. 12, 2012.
3. A. Barisani and D. Bianco. Sniffing keystrokes with lasers/voltmeters. CanSecWest, Mar. 2009. Presentation slides.
4. G. T. Becker, D. Strobel, C. Paar, and W. Burleson. Detecting software theft in embedded systems: A side-channel approach. *IEEE Transactions on Information Forensics and Security*, 7(4), Aug. 2012.

5. C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 2011.
6. S. S. Clark, B. Ransford, and K. Fu. Potentia est scientia: Security and privacy implications of energy-proportional computing. In *HotSec '12*, Aug. 2012.
7. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, Aug. 2004.
8. M. Enev, S. Gupta, T. Kohno, and S. Patel. Televisions, video privacy, and powerline electromagnetic interference. In *ACM Conference on Computer and Communications Security (CCS)*, Oct. 2011.
9. Federal Communications Commission. Code of Federal Regulations, Title 47, Part 15, Sections 101–103, Oct. 2010.
10. M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, Mar. 2004.
11. S. Gupta, M. S. Reynolds, and S. N. Patel. ElectriSense: Single-point sensing using EMI for electrical event detection and classification in the home. In *International Conference on Ubiquitous Computing (UbiComp)*, Sept. 2010.
12. G. W. Hart. Residential energy monitoring and computerized surveillance via utility power flows. *IEEE Technology and Society Magazine*, June 1989.
13. G. W. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12), 1992.
14. A. Hintz. Fingerprinting websites using traffic analysis. In *Workshop on Privacy Enhancing Technologies (PET)*, Apr. 2002.
15. C.-W. Hsu. Multi-label classification. `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/#multi_label_classification`, Nov. 2011.
16. Internet Archive. Internet archive wayback machine. `http://archive.org/web/web.php`, Loaded Mar. 2013.
17. P. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, Aug. 1996.
18. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO*, Aug. 1999.
19. M. G. Kuhn. Electromagnetic eavesdropping risks of flat-panel displays. In *Workshop on Privacy Enhancing Technologies (PET)*, May 2004.
20. M. G. Kuhn. Security limits for compromising emanations. In *Workshop on Cryptographic Hardware and Embedded Systems 2005 (CHES)*, Aug. 2005.
21. M. G. Kuhn and R. J. Anderson. Soft tempest: Hidden data transmission using electromagnetic emanations. In *Information Hiding (IH)*, Apr. 1998.
22. M. Liberatore and B. N. Levine. Inferring the source of encrypted HTTP connections. In *ACM Conference on Computer and Communications Security (CCS)*, Oct. 2006.
23. L. Lu, E. C. Chang, and M. C. Chan. Website fingerprinting and identification using ordered feature sequences. In *European Symposium on Research in Computer Security (ESORICS)*, Sept. 2010.
24. A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, Nov. 2010.
25. P3 International. P3 — Kill A Watt. `http://www.p3international.com/products/special/P4400/P4400-CE.html`, Loaded Feb. 13, 2012.
26. S. N. Patel, T. Robertson, J. A. Kientz, M. S. Reynolds, and G. D. Abowd. At the flick of a switch: Detecting and classifying unique electrical events on the residential power line. In *International Conference on Ubiquitous Computing (UbiComp)*, Sept. 2007.
27. J. Rott. Intel Advanced Encryption Standard instructions (AES-NI). `http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni`, Feb. 2012.

28. D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on SSH. In *USENIX Security Symposium*, Aug. 2001.
29. Q. Sun et al. Statistical identification of encrypted web browsing traffic. In *IEEE Symposium on Security and Privacy*, May 2002.
30. United States Environmental Protection Agency. ENERGY STAR program requirements for computers. `http://www.energystar.gov/ia/partners/prod_development/revisions/downloads/computer/Version5.0_Computer_Spec.pdf`, July 2009.
31. W. van Eck. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers & Security*, 4(4), Dec. 1985.
32. M. Vuagnoux and S. Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *USENIX Security Symposium*, Aug. 2009.
33. A. M. White, A. R. Matthews, K. Z. Snow, and F. Monrose. Phonotactic reconstruction of encrypted VoIP conversations: Hookt on Fon-iks. In *IEEE Symposium on Security and Privacy*, May 2011.
34. C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson. Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob? In *USENIX Security Symposium*, Aug. 2007.
35. B. Yee, D. Sehr, G. Dardyk, B. Chen, R. Muth, T. Ormandy, S. Okasaka, N. Narula, and N. Fullagar. Native Client: A sandbox for portable, untrusted x86 native code. In *IEEE Symposium on Security and Privacy*, May 2009.

A Hardware Specifications

MacBook-1: 2008 model, dual-core Intel Core 2 Duo processor, 4 GB of RAM, Intel GMA X3100 GPU, 80 GB Corsair SATA II MLC solid-state drive, and Mac OS 10.6.8. We removed the battery from MacBook-1 during all experiments.

MacBook-2: 2008 model, dual-core Intel Core 2 Duo processor, 4 GB of RAM, Intel GMA X3100 GPU, 320 GB SATA magnetic drive, and Mac OS 10.7.3. The battery remained in MacBook-2 during all experiments.

Dell Vostro desktop PC: quad-core Intel Core i5 processor, 4 GB of RAM, AMD Radeon 6450 GPU, and 250 GB SATA magnetic drive. We tested the desktop PC under Windows 7 and Ubuntu 10.04.

Lenovo T410 Laptop: Intel Core i5 processor, 4 GB of RAM, 300 GB SATA magnetic drive, and Windows 7 OS. The battery remained in the laptop during all experiments.