Measuring Security Practices and How They Impact Security

Louis F. DeKoven
University of California, San Diego
ldekoven@cs.ucsd.edu

Audrey Randall
University of California, San Diego
aurandal@eng.ucsd.edu

Ariana Mirian
University of California, San Diego
amirian@cs.ucsd.edu

Gautam Akiwate
University of California, San Diego
gakiwate@cs.ucsd.edu

Ansel Blume
University of California, San Diego
ablume@ucsd.edu

Lawrence K. Saul
University of California, San Diego
saul@cs.ucsd.edu

Aaron Schulman
University of California, San Diego
schulman@cs.ucsd.edu

Geoffrey M. Voelker
University of California, San Diego
voelker@cs.ucsd.edu

Stefan Savage
University of California, San Diego
savage@cs.ucsd.edu

ABSTRACT

Security is a discipline that places significant expectations on lay users. Thus, there are a wide array of technologies and behaviors that we exhort end users to adopt and thereby reduce their security risk. However, the adoption of these "best practices" — ranging from the use of antivirus products to actively keeping software updated — is not well understood, nor is their practical impact on security risk well-established. This paper explores both of these issues via a large-scale empirical measurement study covering approximately 15,000 computers over six months. We use passive monitoring to infer and characterize the prevalence of various security practices in situ as well as a range of other potentially security-relevant behaviors. We then explore the extent to which differences in key security behaviors impact real-world outcomes (i.e., that a device shows clear evidence of having been compromised).

CCS CONCEPTS

• **Security and privacy** → *Intrusion detection systems*; • **Networks**;

ACM Reference Format:
Louis F. DeKoven, Audrey Randall, Ariana Mirian, Gautam Akiwate, Ansel Blume, Lawrence K. Saul, Aaron Schulman, Geoffrey M. Voelker, and Stefan Savage. 2019. Measuring Security Practices and How They Impact Security. In *Internet Measurement Conference (IMC '19), October 21–23, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3355369.3355571

1 INTRODUCTION

Ensuring effective computer security is widely understood to require a combination of both appropriate technological measures and prudent human behaviors; e.g., rapid installation of security updates to patch vulnerabilities or the use of password managers to ensure login credentials are distinct and random. Implicit in this status quo is the recognition that security is not an intrinsic property of today's systems, but is a byproduct of making appropriate choices — choices about what security products to employ, choices about how to manage system software, and choices about how to engage (or not) with third-party services on the Internet. Indeed, the codifying of good security choices, commonly referred to as security policy or "best practice", has been a part of our lives as long as security has been a concern.

However, establishing the value provided by these security practices is underexamined at best. First, we have limited empirical data about which security advice is adopted in practice. Users have a plethora of advice to choose from, highlighted by Reeder et al.'s recent study of expert security advice, whose title — "152 Simple Steps to Stay Safe Online" — underscores both the irony and the variability in such security lore [35]. Clearly few users are likely to follow all such dicta, but if user behavior is indeed key to security, it is important to know which practices are widely followed and which have only limited uptake.

A second, more subtle issue concerns the efficacy of security practices when followed: Do they work? Here the evidence is scant. Even practices widely agreed upon by Reeder's experts, such as keeping software patched, are not justified beyond a rhetorical argument. In fact, virtually all of the most established security best practices — including "use antivirus software", "use HTTPS/TLS", "update your software regularly", "use a password manager", and so on — have attained this status without empirical evidence quantifying their impact on security outcomes. Summarizing this state of affairs, Herley writes, "[Security] advice is complex and growing, but the benefit is largely speculative or moot", which he argues leads rational users to reject security advice [17].

To summarize, our existing models of security all rely on end users to follow a range of best practices. However, we neither understand the extent to which they are following this advice, nor do we have good information about how much this behavior ultimately impacts their future security.

This paper seeks to make progress on both issues — the prevalence of popular security practices and their relationship to security outcomes — via longitudinal empirical measurement of a large population of computer devices. In particular, we monitor the online

IMC '19, October 21–23, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-6948-0/19/10…$15.00
https://doi.org/10.1145/3355369.3355571

behavior of 15,291 independently administered desktop/laptop computers and identify per-device security *behaviors*: what software they are running (e.g., antivirus products, password managers, etc.), is the software patched, and what is their network usage (e.g., does the machine contact file sharing sites), etc., as well as concrete security *outcomes* (i.e., whether a particular machine becomes compromised). In the course of this work, we describe three primary contributions:

- Large-scale passive feature collection. Our results are based on large-scale measurement using passive monitoring. In doing so, we develop and test a large dictionary of classification rules to indirectly infer software state on monitored machines (e.g., that a machine is running antivirus of a particular brand, or if its operating system has been updated). In addition, to ensure that features are consistently associated with particular devices, we describe techniques for addressing a range of aliasing challenges due to DHCP and to DNS caching.
- Outcome-based analysis. We use a combination of operational security logs and network intrusion detection alerts to identify the subset of machines in our data set that are truly compromised. This outcome data allows us to examine the impact of adopted security practices in terms of individual security outcomes and with respect to concrete time periods surrounding the likely time of compromise.
- Prevalence and impact of security practices. For our user population, we establish the prevalence of a range of popular security practices as well as how these behaviors relate to security outcomes. We specifically explore the hypotheses that a range of existing "best practices" are negatively correlated with host compromise or that "bad practices" are positively correlated. We consider both behaviors that could *directly* lead to compromise and those which may *indirectly* reflect a user's attentiveness to security hygiene.

Finally, while we find a number of behaviors that are positively correlated with host compromise, few "best practices" exhibit the negative correlations that would support their value in improving end user security.

2 BACKGROUND

This study follows a large body of prior work that empirically relates user activity to various risk factors, which we highlight in five categories below.

Small scale studies of individuals. In 2008, Carlinet et al. [6] analyzed three-hour long packet traces of ADSL customers (from 200–900 customers) and correlated hosts that experienced at least one Snort IDS alert with other factors. Their study revealed a relationship between those machines raising alerts, and their use of the Windows operating system as well as heavy web browsing habits. Our study is similarly based on passive network data collection, but we operate at a significantly larger scale (in number and diversity of hosts as well as duration) and we also explicitly try to control for a range of confounding factors.

Aggregate studies of user behavior. Others have studied risk factors in aggregate across large organizations. Notably, Yang et al. [23] correlated publicly-declared data breaches and web site hacks with external measurements (e.g., misconfigured DNS or HTTPS certificates). They found that evidence of organizational failures to police security is predictive of attacks. Similarly, recent papers have focused on exploring how differences in deployed defenses (e.g., across ISPs or web sites) relate to the occurrence of particular attacks [40, 42], and Xiao et al. [49] showed that user patterns of security activity can be a predictor of future malware outbreaks in an ISP.

Web access behavior. Other researchers have investigated how a user's web browsing habits reveal risk factors. Levesque et al. [22] monitored web browser activity for 50 users over four months and found that the likelihood of visiting a malware hosting site was correlated with the other kinds of sites a machine visited (e.g., with peer-to-peer (P2P) and gambling sites). Canali et al. [5] replicated this study using antivirus telemetry (100,000 users), and Sharif et al. [38] describe a similar analysis for 20,000 mobile users. Both found that frequent, nighttime, and weekend browsing activity are correlated with security risk.

Software Updates. Another vein of research has correlated poor software update habits with indicators of host compromise. Kahn et al. [21] used passive monitoring of roughly 5,000 hosts to infer software updates and used the Bothunter traffic analysis tool [15] to infer likely infected hosts based on suspicious traffic patterns (e.g., based on outbound scanning). They found a positive correlation between infection indicators and a lack of regular updating practice.

At a larger scale, Bilge et al. [4] used antivirus logs and telemetry from over 600,000 enterprise hosts to retrospectively relate software updates to subsequent infections. They found that devices that do not patch correlate with those that were at some point infected. Finally, Sarabi et al. [36] used a similar data set of 400,000 Windows hosts and found that patching faster provides limited benefit if vulnerabilities are frequently introduced into product code.

Human factors. Finally, there is an extensive literature on the human factors issues involved in relating security advice to users, the extent to which the advice leads to changes in behaviors, and how such effects are driven by both individual self-confidence and cultural norms [13, 32–34, 37, 43–45].

3 METHODOLOGY

Our measurement methodology uses passive network traffic monitoring to infer the security and behavioral practices of devices within a university residential network. This approach has numerous advantages, including scalability (we are able to collect data from tens of thousands of devices) and granular analysis (we can frequently infer when a device updates a particular application and to what version). However, it also introduces liabilities (a focus on a particular population) and risks (in particular to privacy). In this section we first focus on the technical aspects of our data collection methodology and then discuss some of its attendant challenges and limitations.

3.1 Network Traffic Processing

The first stage of our system takes as input 4–6 Gbps of raw bi-directional network traffic from the campus residential network, and outputs logs of processed network events at the rate of millions of records per second. As part of this stage, campus IP addresses are anonymized and, to track the contemporaneous mapping of IP

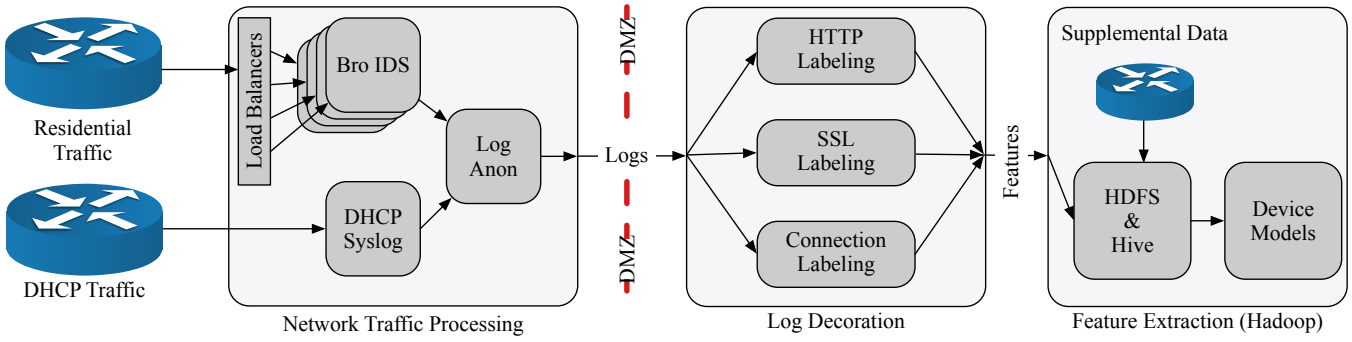


Figure 1: System architecture overview. Network traffic is first processed into logs and its addresses anonymized. The next stage replays the network traffic logs to extract further information and label each connection with (also anonymized) MAC address information. The decorated logs are then stored in Hive where they are labeled with security incidents, security practice features, and behavioral features. Lastly, device models are created for analysis.

addresses to device MAC addresses, this stage also collects and compatibly anonymizes contemporaneous Dynamic Host Configuration Protocol (DHCP) syslog traffic.

3.1.1 Residential Network Traffic. As shown in the Network Traffic Processing stage of Figure 1, our server receives network traffic mirrored from a campus Arista switch using two 10G fiber optic links. In addition to load balancing, the switch filters out high-volume traffic from popular content distribution networks (CDNs) (e.g., Netflix, YouTube, Akamai, etc.), resulting in a load of 4–6 Gbps of traffic on our server.

To minimize loss while processing traffic, we experimented with a number of network processing configurations before settling on the following. We use the PF_RING ZC (Zero Copy) framework [27] to move traffic from the network card directly into user-level ring buffers, bypassing the kernel. We then use the `zbalance_ipc` application from PF_RING ZC to locally perform 4-tuple load balancing across many virtual network interfaces. Instances of the Bro (now Zeek) Intrusion Detection System (IDS) [28] then read from each virtual network interface, consuming and processing the network traffic into a custom log format. This configuration results in an average daily loss of 0.5% of received packets throughout our six-month measurement period.

While IDSes are typically used for detecting threats and anomalous network behavior, we use Bro to convert network traffic into logs since it is extensible, discards raw network traffic as soon as a connection is closed (or after a timeout), and is able to parse numerous network protocols [50]. We also customize the Bro output logs to record only information needed to identify security practice and behavioral features.

In particular, we use the `HTTP`, `SSL`, `DNS`, and `Connection` protocol analyzers. The `HTTP` analyzer provides a summary of HTTP traffic on the network, including components such as the HOST and URI fields. The `SSL` analyzer extracts the Server Name Indication (SNI) field from TLS connections. SNI is an extension of the TLS protocol enabled by most modern browsers, and allows a client to indicate the hostname it is contacting at the start of an encrypted connection. The SNI field is particularly useful for inferring the destination of connections that otherwise are encrypted. The `DNS` analyzer provides a summary of Domain Name System (DNS) requests and responses. Lastly, the `Connection` analyzer summarizes information about TCP, UDP, and ICMP connections.

Every thirty minutes Bro rotates the previous logs through an address anonymization filter that encrypts campus IP addresses. At this stage of processing, the logs contain IP addresses and not MAC addresses since DHCP traffic is not propagated to our network vantage point. After being so anonymized, the logs are rotated across the DMZ to another server for further processing (Section 3.2).

3.1.2 DHCP Traffic. The server also runs a syslog collector that receives forwarded DHCP traffic from the residential network's DHCP servers. DHCP dynamically provides an IP address to a device joining the network. The IP address is leased to the device (by MAC address) for a specified duration, typically 15 minutes. Since we need to track a device's security and behavioral practices for long time periods, we utilize this IP-to-MAC mapping in later processing.

Similar to the Bro IDS logs, every thirty minutes we process the previous DHCP traffic into a (MAC address, IP address, starting time, lease duration) tuple. Then, the entire IP address and identifying lower 24-bits of the MAC address are encrypted using a similar address anonymization filter. The anonymized DHCP logs are then rotated across the DMZ to the Log Decoration server.

3.2 Log Decoration

The second stage takes as input these intermediate network event and DHCP logs, and processes them further to produce a single stream of network events associated with (anonymized) device MAC addresses and domain names.

Associating Flows to Devices. Our goal is to model device behavior based upon network activity over long time spans. While we identify unique devices based upon their MAC address, the network events that we collect have dynamically assigned IP addresses. As a result, we must also track dynamic IP address assignments to map IP-based network events to specific device MAC addresses.

We use a Redis key-value store [31] to build a DHCP cache by replaying campus DHCP logs. We use the DHCP cache to assign a

MAC address to the inbound and outbound IP of each connection. We consider an IP-to-MAC mapping valid if a connection takes place during the time when the IP address was allocated and the lease is still valid. In the event that there is not a valid mapping (e.g., the IP address is a non-university IP, or a the device uses a static IP), we do not assign a MAC address to the IP.

Associating Flows to Domains. When using network activity to model device behavior, it useful to know the domain name associated with the end points devices are communicating with (e.g., categorizing the type of web site being visited). We also extract the registered domain and top-level domain (TLD) from each fully qualified domain name using the Public Suffix List [25]. Again, since the network events we observe use IP addresses, we must map IP addresses to domain names. And since the mapping of DNS names to IP addresses also changes over time, we also dynamically track DNS resolutions as observed in the network so that we can map network events to the domain names involved.

Due to our network vantage point (at the campus edge), the DNS traffic our collection server observes generally has the source IP address of our local DNS resolver, and *not* the IP address of the host which will subsequently make a connection to the resolved IP.[1] This constraint limits our ability to use the DNS mapping alone to infer a connection's domain name. Therefore, one of the steps in this stage is to build a local DNS cache by replaying the logs in chronological order and labeling the domain name of observed connections where it is not already provided (i.e., excluding HTTP and SNI-labeled connections).

We use another Redis key-value store to build a DNS cache by replaying DNS traffic. The cache tracks the mappings of each IP address to domain name at the time the IP address was observed. We consider a mapping to be valid as long as it has not expired — the log time falls between the time at which the DNS request was observed plus the response time to live (TTL) — and there is one registered domain name mapped to the IP address.

When sites use virtual hosting, it is possible that an IP address has multiple domain names associated with it. In this case, we first check if the registered domain names match (e.g., bar.bar.com and car.bar.com share a registered domain of bar.com). If the registered domains match, we label the connection using the longest suffix substring match (e.g., ar.bar.com) and set a flag indicating that the fully qualified domain name has been truncated. In the case where there is more than one registered domain with a valid mapping to the IP address, we do not use the mapping to label connections until enough of the conflicting mappings expire such that they share a registered domain, or there is only one mapping.

User Agent. We parse HTTP user agent strings using the open-source `ua-parser` library. From the user agent string we extract browser, operating system (OS), and device information when present.

3.3 Feature Extraction

In the final stage of our system we store the log events in a Hive database [1] and process them to extract a wide variety of software and network activity features associated with the devices and their activity as seen on our network. The last critical feature is device outcomes: knowing when a device has become compromised. We derive device outcomes from a log of alerts from a campus IDS appliance, and also store that information in our database.

3.3.1 Software Features. To identify features describing application use on devices, we crafted custom network traffic signatures to identify application use (e.g., a particular peer-to-peer client) as well as various kinds of application behavior (e.g., a software update). To create our network signatures we use virtual machines instrumented with Wireshark [48]. We then manually exercise various applications and monitor the machine's network behavior to derive a unique signature for each application. Fortunately most applications associated with security risk frequently reveal their presence when checking for updates. In total, we develop network signatures for 68 different applications, including OSes. For a subset of applications, we are also able to detect the application's version. Knowing application versions allows us to compare how fine-grained recommended security practices (i.e., updating regularly) correlates with device compromise.

Antivirus Software. Using antivirus software is virtually always recommended. We created network signatures for 12 popular antivirus products, seven of which were recognized as offering the "Best Protection" for 2019 [26].

Operating System. We created six signatures to identify the OSes running on devices. Since regular OS updating is a popular recommended security practice, we also created signatures to detect OS updates. While Windows and Mac OS operating system updates are downloaded over a Content Delivery Network (CDN) that is removed from the network traffic before reaching our system (Section 3.1), we can use OS version information from the host header and User-Agent string provided in HTTP traffic to infer that updates have taken place.

Applications. Through a combination of network and User-Agent string signatures we detect 41 applications, including those commonly perceived as risky such as Adobe Flash Player, Adobe Reader, Java, Tor, P2P applications, and more. We also detect other popular applications, including browsers, Spotify, iTunes, Outlook, Adobe AIR, etc.

Password Managers. As password managers are frequently recommended to avoid collateral damage of leaked passwords, we also crafted network signatures for nine popular password managers [7].

3.3.2 Network Activity. We track a wide variety of network activity features to quantitatively measure the protocols used (e.g., HTTP, and HTTPS), the categories of sites visited (e.g., file sharing services), when devices are most active, etc.. In doing so, we implement a set of features similar to those used by Canali et al. [5] and Sharif et al. [38] that focused on web browsing activity. As our data set also includes traffic beyond HTTP, we can measure additional behaviors (e.g., remote DNS resolver usage, HTTPS traffic usage, etc.).

Content Categorization. We use the IAB Tech Lab Content Taxonomy to categorize every registered domain in our data set [19]. The domain categorization was generously provided by Webshrinker [9, 47]. The IAB taxonomy includes 404 distinct domain categories [46]. We use the domain categorization to measure the fraction of unique domains each device accesses in a specific category. We also built a list of file hosting sites, and URL shortening services that we use to identify when a device accesses these types of services.

[1] The primary exceptions are devices configured to use remote DNS resolvers.

Usage Patterns. We also develop a number of behavioral features that describe the quantities of HTTP and HTTPS traffic in each TLDs, and the number of network requests made. Additionally, we develop features that quantify customized or non-standard behaviors such the use of remote DNS resolvers, and the proportions of HTTP requests made directly to IP addresses (instead of a domain name).

3.3.3 Detecting Security Incidents. While previous work has relied on the use of blacklists or Google Safe Browsing to identify devices that expose users to potential risk, we are able to identify compromised devices with high confidence as a result of post-infection behavior, typically in the form of command and control (CNC) communication [5, 38]. To identify compromised devices (i.e., ones with a security incident) we use alerts generated by a campus network appliance running the Suricata IDS [39]. The campus security system uses deep packet inspection with an industry-standard malware rule set to flag devices exhibiting post-compromise behavior [30].

The IDS rules also detect network activity that might occur before a device becomes compromised (e.g., possible phishing attempts, exploit kit landing pages, etc.). Since we focus on compromised devices, we reduce the rules we consider to ones that explicitly detect post-infection behavior. False positives are likely with any real-world signature-based intrusion detection system. To minimize the frequency of false positives, we manually remove rules that are frequently triggered, but do not indicate that a device has been compromised.

3.4 Ethical Considerations and Limitations

Having described our measurement methodology in considerable detail, we now consider the risks it presents — both to the privacy of network users and to the validity of conclusions drawn from these measurements.

Protecting user privacy. Foremost among the risks associated with the passive measurement approach is privacy. Even with the prevalence of encrypted connections (e.g., via TLS), processing raw network data is highly sensitive. From an ethical standpoint, the potential benefits of our research must be weighed against potential harms from any privacy violations. In engaging with this question — and developing controls for privacy risk — we involved a broad range of independent campus entities including our institutional review board (IRB), the campus-wide cybersecurity governance committee and our network operations and cybersecurity staff. Together, these organizations provided necessary approvals, direction and guidance in how to best structure our experiment, and strong support for the goals of our research. The campus security group has been particularly interested in using our measurements to gain insight into the security risks of devices operating on their network.[2]

Operationally, we address privacy issues through minimization, anonymization and careful control over data. First, as soon as each connection has been processed, we discard the raw content and log only metadata from the connection (e.g., a feature indicating that device X is updating antivirus product Y). Thus, the vast majority of data is never stored. Next, for those features we do collect, we anonymize the campus IP and the last 24-bits of each MAC address, using a keyed format-preserving encryption scheme [3].[3] Thus, we cannot easily determine the identity of which machine generated a given feature and, as a matter of policy, we do not engage in any queries to attempt to make such determinations via re-identification. Finally, we use a combination of physical and network security controls to restrict access to both monitoring capabilities and feature data to help ensure that outside parties, not bound by our policies, are unable to access the data or our collection infrastructure. Thus, the server processing raw network streams is located in a secure campus machine room with restricted physical access, only accepts communications from a small static set of dedicated campus machines and requires multi-factor authentication for any logins. Moreover, its activity is itself logged and monitored for any anomalous accesses. We use similar mechanisms to protect the processed and anonymized feature data, although these servers are located in our local machine room. The feature data set is only accessible to members of our group, subject to IRB and our agreements with campus, and will not (and cannot) be shared further.

Limitations of our approach. In addition to privacy risk, it is important to document the implicit limitations of our study arising from its focus on a residential campus population — primarily undergraduates — as well as the use of a particular IDS and rule set to detect security incidents [30, 39].

It is entirely possible that the behavioral modes of this population, particularly with respect to security, are distinct from older, less affluent or more professional cohorts. This population bias is also likely to impact time-of-day effects, as well as the kinds of hardware and software used. Additionally, the security incidents we consider rely on the Suricata IDS, commercial network traffic signatures, and security-related network usage requirements of our university environment (e.g., residential students are nominally required to have antivirus software installed on their devices before connecting). It is entirely possible that these incident detection biases also influence the behaviors and software applications that correlate with device compromise. Thus, were our same methodology employed in other kinds of networks, serving other populations, or using different security incident detection techniques, it is possible that the results may differ. For this reason, we hope to see our measurements replicated in other environments.

4 DATA SET

We analyze six months of data from our passive network traffic processing system from June 2018 to December 2018. In this section we describe our approach for identifying the laptop and desktop devices for use in analyzing security risk factors, and determining the dominant OS of devices used in our analysis. In the end, our data set consists of 15,291 devices. Table 1, characterizes our data set in terms of connections processed, and inbound and outbound bytes.

4.1 Device Filtering

The university allows heterogeneous devices on its network, including personal computers, mobile phones, printers, Internet of Things (IoT) devices, and more. Recommended security practices,

[2] Indeed, during the course of our work we have been able to report a variety of unexpected and suspicious activity to campus for further action.

[3] Thus, the IP address `192.168.0.1` may be replaced with `205.4.32.501` and the MAC address `00:26:18:a5:38:24` may become `00:26:18:b5:fe:ba`. We do not anonymize the organizationally unique identifier (OUI) to allow us to derive the network device manufacturer.

Name	Value
Date Range	June 2018 – December 2018
Total Filtered Devices	15,291
DNS Connections	17.1 B
Non-DNS Connections	1.92 B
Total Connections	19 B
Outbound Bytes	38.4 TB
Inbound Bytes	720 TB
Total Bytes	758 TB

Table 1: Data set characterization. Note that our network vantage point provides DNS requests from the local resolver, which includes DNS traffic from devices in this paper as well as other devices using the university's networks.

however, are commonly offered for laptop and desktop computers, and therefore we focus our analysis solely on such devices. As a result, we develop techniques to identify laptop and desktop computers among the many other devices on the network. We remove devices that are easily identifiable, and then develop heuristics to filter remaining devices.

We first remove devices that are not active for a minimum of 14 days, and ones that never provide a major web browser's User-Agent string (removing 13.1% of all devices). For studying security practices, devices need to have a modicum degree of network activity to be able to model behavior, and devices without any web browser activity are a strong indication that they are not laptops or desktops.

Next, we use User-Agent strings to identify a device's OS [14]. Since applications are not required to provide accurate User-Agent string information, to identify a device's OS we consider User-Agent strings from major browsers, and require that a device's OS is consistent on 95% of all requests. We identify 40.8% of the total devices as having a mobile or IoT OS and remove them from our data set. For the fraction of devices that fall below the 95% requirement, we remove ones that frequently contact domains which are not regularly accessed by laptop or desktop devices[4] (4.1% of all devices).

We also compile a list of network hardware vendors used within devices other than laptops and desktops (e.g., Vizio, etc.), and remove devices with a matching organizationally unique identifier (OUI) vendor (2.2% of devices).

Lastly, we filter some of the remaining IoT devices using network traffic-based heuristics. Our intuition is that most of these devices[5] will either make close to the same number of connections each day, a small number of daily connections, or connections within a limited number of /24 network subnets. We pick each threshold by manually inspecting the three network traffic distributions, and select the value corresponding to the first peak of the distribution. We remove devices that make the same number of connections each day ±7, on-average 40 daily connections, or contact on-median 31 distinct /24 networks each week (4.2% of all devices).

[4]We manually label eight domains that are contacted by TVs, printers, game consoles, and iPhones, such as "hpeprint.com," "vizio.com," "nintendowifi.net," or "iphone-ld.apple.com." If any of these domains are in the device's ten most-frequently-accessed domains, we exclude it. We also exclude devices that never make a single connection to any university web site.

[5]With the exception of user-directed IoT devices (e.g., Chromecasts, etc.).

To validate our device filtering heuristics in practice, we manually label a sample of 100 devices (50 laptop and desktop, and 50 that are removed). We find our filtering methodology to be sufficiently accurate: one laptop is incorrectly removed, and four mobile phones are incorrectly included. The excluded laptop was removed because it did not have a consistent OS in 95% or more of its User-Agent strings. The four mobile phones reported desktop operating systems as well as "Android" in 95% of their User-Agent strings (for example, a User-Agent containing the string "Linux; U; Android;" was common for several devices). To allow for the legitimate cases where a desktop or laptop could be running multiple operating systems, we do not exclude devices like these from our data set.

4.2 Identifying Dominant OSes

Since different OSes have different risk profiles, identifying the OS used by a device is an important step. Being able to observe device network traffic makes OS identification an interesting task. The majority of devices are straightforward: using signatures of OS update events, we can immediately identify a single unambiguous OS for 79.1% of devices.

The remaining devices either have no OS update signatures, or have more than one.[6] For these devices, we use a combination of OS update signatures, OS User-Agent strings, and Organizational Unique Identifier (OUI) vendor name information to identify the dominant OS of a device (e.g., the host OS with virtual machines, Windows if tethering an iPhone, etc.). We assume that devices with an Apple OUI vendor name will be using Mac OS (7.2%). We then use the dominant OS extracted from User-Agent strings to assign an OS (11.5%). The remaining 340 devices (2.1%) have both Windows and Mac OS updates. We choose to assign Windows as the dominant OS in these cases because of strong evidence of tethering, in which iTunes allows users to update their Apple devices (e.g., iPhone, iPad, etc.) using the network connection of their computer [2].[7] For each of these heuristics, we confirmed the labeling by manually checking the traffic profile of a random sample of devices.

5 RECOMMENDED PRACTICES

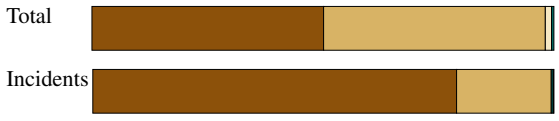
There are a variety of security practices widely recommended by experts to help users become safer online. Prior work has explored some of these practices in terms of users being exposed to risky web sites [5, 38]. Since our data includes actual security outcomes, we start our evaluation by exploring the correlation of various security practices to actual device compromises in our user population: operating system choice, keeping software up to date, web sites visited, using HTTPS, using antivirus software, and software used.

5.1 Operating System

Different operating systems have different security reputations, so it is not surprising that experts have recommendations of the form "Use an uncommon OS" [35]. Part of the underlying reasoning is that attackers will spend their efforts targeting devices with most

[6]There are a number of legitimate reasons why a device can have more than one OS detected, including dual-booting between different OSes, using virtual machines, device tethering, etc.

[7]We measure the baseline of iTunes installs across devices with only Windows to be 11.9%, whereas the install rate for these 340 devices is 67%.



Operating System	*Incidents*	*Total Devices*
Windows	538 (7.0%)	7,668
Mac OS	140 (1.9%)	7,339
ChromeOS	1 (0.5%)	205
Linux Variant	3 (3.8%)	79

Figure 2: Device OS classification after removing IoT and mobile devices: the total number of devices with each OS and the number with a security incident.

common systems, so using an uncommon operating system makes that device less of a target.

In terms of device compromise, as with previous work and experience, such advice holds for our user population as well. Using the OS classification method described in Section 4.2, Figure 2 shows the number of devices using major operating systems and the number of each that were compromised during our measurement period. Most devices use Windows and Mac OS, split nearly equally between the two. The baseline compromise rate among devices is 4.5%, but Windows devices are 3.9× more likely to be compromised than Mac OS devices. The Chrome OS population is small, but only one such device was compromised.

Of course, modulo dual-booting or using virtual machines, this kind of advice is only actionable to users when choosing a device to use, and is no help once a user is already using a system.

5.2 Update Software

Among hundreds of security experts surveyed, by far the most popular advice is to "Keep systems and software up to date" [35]. In this part we explore the operating system, browser, and Flash update characteristics of the devices in our population, and how they correlate with device compromise.

5.2.1 Operating System. **Mac OS.** We start by analyzing the update behavior of devices running Mac OS. Our system labels each HTTP connection of a device with the type of operating system and its current version number, both extracted from the User-Agent string. However, if a device leaves the network and returns with an updated version number in the UA string, then we cannot accurately tell when the device was updated. Thus, to bound the error on update times we only include devices that are never absent from the network for more than three days (in practice few devices are absent for long).

We see 7,268 (47.5%) devices that identify as Mac according to the User-Agent string and are not absent from the network for a long period. Of these devices, we see at least one update for 2,113 of them (29.1% of all Mac OS devices). Figure 3 shows the update pattern of these Mac OS devices over time, anchored around the three OS updates released by Apple during our measurement period. In general, Mac OS users are relatively slow to update, anecdotally because of the interruptions and risks Mac OS updates entail.

Of these devices, 57 (2.7%) of them were compromised. Compromised devices have a mean and median update rate of 16.2 and 14.0 days, respectively, while their clean counterparts have a mean and

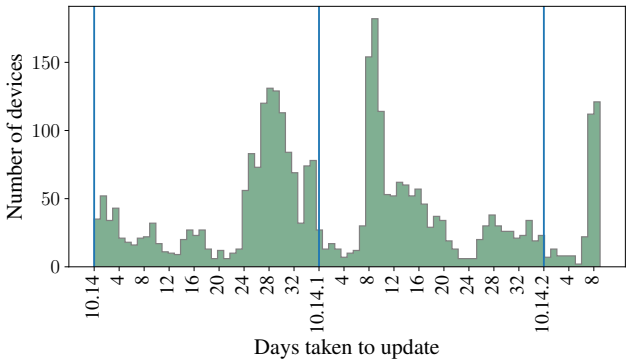


Figure 3: Number of days a Mac OS device takes to update to a specific version. The version number on the x-axis denotes the day that the specified version update was published.

Incident?	# Devices	μ	Median	P90	P95	P99	σ^2
No	5,976	2.5	0	6	15	42	59
Yes	483	2.6	0	6	14	49	62

Table 2: Windows device updates deltas. We compute the average, median, P90, P95, P99, and variance of the number of days between when the update was released, and when we observe each device download the update. The devices are partitioned by those with and without a security incident.

median update rate of 18.0 and 16.0 days. However, this difference is not statistically significant according to the Mann-Whitney U test ($p = 0.13$).[8]

Windows. For Windows we developed a signature to extract the knowledge base (KB) number of "Other Software" updates (e.g., Adobe Flash Player, etc.).[9] Our signature detects when a device downloads the update. While we cannot verify that the update was applied, it does indicate whether the device is using the default Windows Update settings. Since it is possible to miss an update (e.g., a device may download the update while connected to a different network), we only compare devices that we see updating. We also restrict the updates considered to ones released during our measurement period since there is nothing preventing an unpatched device from joining the network.[10] We identify the update's release day using Microsoft's Update Catalog service [24].

Across devices running Windows, we see at least one update for 6,459 of them (84% of all Windows devices). Table 2 shows the average, median, P90, P95, P99, and variance of the number of days between when an update is downloaded and when it is released. Based upon the averages and medians, devices update with similar deltas (2.5 days and 0 days, respectively) regardless of whether they have a security incident. We confirm our hypothesis using the

[8] The Mann-Whitney U test is a non-parametric statistical test that can be used to determine if two independent samples are selected from populations with the same distribution. The null hypothesis for a Mann-Whitney U test is that the populations are selected from the same distribution.

[9] An example update is https://support.microsoft.com/en-us/help/4462930

[10] We exclude updates released multiple times with the same KB number.

Browser	Mean, Median, # (Cmp)	Mean, Median, # (Cln)
Chrome	14.4, 15.0 (421)	15.4, 15.0 (7883)
Firefox	5.64, 3.00 (24)	9.65, 5.00 (424)

Table 3: Number of days between when an update is published and when devices update. Compromised devices update faster than their clean counterparts across their lifetimes.

Mann-Whitney U test ($p = 0.052$). We also find the fraction of compromised devices that update (7.5%) to be similar in magnitude to the baseline fraction of incidents across all Windows devices (7.0%). In short, the update behavior of compromised Windows devices is little different than that of clean devices.

5.2.2 Web Browser. Updating the browser may be as important as updating the operating system. Browsers are also large, complex pieces of software used on a daily basis and, as with most software, these large programs have vulnerabilities. Updating the browser is viewed as such an important process that Chrome and Firefox employ auto-updating by default [12, 41], with UI features to encourage timely updating.

As such, we explore the relationship between compromised and clean devices and browser updating behaviors. Similar to the Mac OS devices, we are able to detect the current browser version number from the User-Agent string of a device. Since browser vendors publish the dates when they make updates available,[11] we can check whether the browser on a device is out of date each time we see the device on the network. Across the measurement period, we then calculate how quickly devices update. Also, similar to the Mac OS analysis, we exclude devices that are absent from the network for more than consecutive three days.

Moreover, we only analyze the dominant browser for each device. Many devices have User-Agent strings naming different browsers. While users may use different browsers for different use cases, we identify a dominant browser to remove the noise from user applications that spoof a browser in their User-Agent string. Thus, we determine which browser connects to the largest number of distinct registered domains from a device and label the device with that dominant browser. We choose unique registered domains as our metric over number of HTTP connections because there are web sites and applications that "spam" the network, making the device appear to use one browser dominantly when the natural user behavior is actually coming from a different browser.

We analyzed updates for devices that dominantly use Chrome, Edge, Firefox, and Safari. Of the total devices, 10,831 (70.8%) devices use Chrome, 719 (4.7%) devices use Edge, 561 (3.7%) devices use Firefox, and 2993 (19.6%) devices use Safari. However, only 8,304 (76.7%) of the Chrome devices, 132 (18.4%) of the Edge devices, 448 (80.0%) of the Firefox devices, and 1592 (53.2%) of the Safari devices are on the network continuously (absent for less than three days). Table 3 shows the browsers with statistically significant differences in update time between clean and compromised devices (Mann Whitney U: Chrome $p = 4.2 \times 10^{-4}$ and Firefox $p = 0.03$).

[11] During our measurement period each popular browser had at least three major updates.

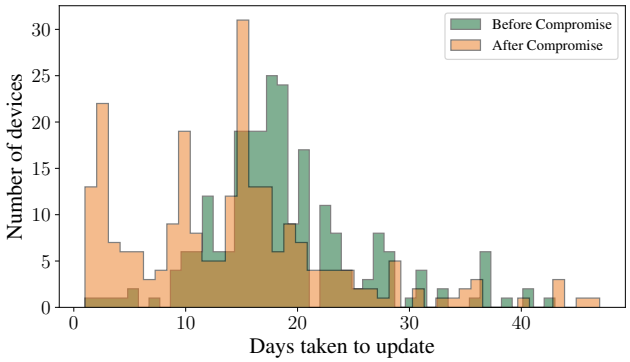


Figure 4: Distribution of days a device takes to update Chrome before compromise and after compromise.

Incident?	# Devices	μ	Median	P90	P95	P99	σ^2
No	1,702	4.2	1	16	20	30	53
Yes	149	3.7	1	16	21	26	47

Table 4: Flash Player updates on Windows devices.

Clean devices appear to spend more time out of date than their compromised counterparts. Examining this phenomenon in more detail, we compare the update behavior of compromised devices before and after their compromise date. We focus on devices using Chrome that have two updates spanning the compromise event (other browsers do not have a sufficiently large sample size). Figure 4 shows the distribution of times devices were out of date with respect to when a browser update was released for updates before and after the device was compromised. The shift in distributions illustrates that devices update faster after compromise. In more detail, devices that use Chrome have a before-compromise mean update rate of 18.9 days (18.0 median days) and an after-compromise mean update rate of 14.2 days (15.0 days median). This difference is significant, with $p = 4.8 \times 10^{-12}$ using the Wilcoxon signed-rank test.[12]

5.2.3 Flash Player. The Adobe Flash player has long been associated with security risk and device compromise. The typical recommendation is to not use Flash at all, but if you do, to keep it up to date. We created a signature to detect Adobe Flash Player on Windows devices.[13] We focus on the desktop version of Flash as major browser vendors issue Flash plugin updates directly. Adobe released six updates within our measurement period, and we use Adobe's web site to identify the version and release date for each.

Somewhat surprisingly, desktop Flash is still quite prevalent on devices. Fortunately, though, update patterns and compromise rates do not indicate that the use of Flash puts devices at greater risk of compromise. A total of 2,167 devices (28% of Windows devices) check for a Flash Player update, of which 1,851 are seen downloading an update. Table 4 shows the average, median, P90, P95, P99, and

[12] The Wilcoxon signed-rank test is a non-parametric paired difference test which indicates if the means of two dependent samples differ. The null hypothesis of the Wilxocon signed-rank test is that the means do not differ.

[13] Flash Player updates on Mac OS are downloaded over HTTPS, preventing us from crafting an effective signature.

variance of the number of days between when an update is downloaded and when it is released. Curiously, compromised devices updated Flash slightly faster than clean devices (Mann-Whitney U test $p = 0.025$). However, the rate of compromise across devices that update Flash is 8.1%, only slightly higher than the rate across of Windows devices (7.9%) (Chi-Square $p = 0.057$).[14] Among the 316 devices that we detect Flash Player on, but do not see updates, only 15 are compromised (4.8%). We interpret these results as a community success story. A combination of widespread awareness, aggressive updates, and focused attention have mitigated desktop Flash as a significant risk factor.

We next explore why compromised devices update Flash Player more quickly. We hypothesize that a compromised device's update behavior will change after being compromised. To evaluate this claim, we compare the update patterns for compromised devices before and after becoming compromised. Out of the 149 compromised devices that update Flash, there are 60 devices (40.3%) with updates before and after their first incident. The median and average days compromised devices take to update before an incident are 6.5 and 9.9 respectively, and 0 and 1 days after becoming compromised (Wilcoxon signed-rank test $p = 1.73 \times 10^{-7}$). As with Chrome browser update behavior, these results suggest that shortly after a security incident devices exhibit better Flash update hygiene.

5.3 Visit Reputable Web Sites

Experts recommend users to be careful in the web sites that they visit ("Visit reputable web sites" [35]), and indeed prior work has found that the category of web site users visit can be indicative of exposure to risky sites [5, 38]. We perform a similar analysis for devices that are actually compromised, and for the most part confirm that the types of sites that lead to exposure to risky sites also correlate with actual compromise.

To categorize the content devices access we use the IAB domain taxonomy (Section 3.3.2). We use the Kolmogorov-Smirnov (KS) test with Bonferroni correction to compare the ECDFs of the fraction of distinct registered domains in each category that clean and compromised devices access, and confirm that they are statistically significant (i.e., $p < 0.001$).[15]

Table 5 shows the most substantial differences between the types of content accessed, e.g., with clean devices accessing more business, advertising, and marketing content, while compromised devices accessed more gaming, hobby, uncategorized, and illegal. We note that, while previous work found that exposed devices visit more advertising domains [38], our finding of the opposite behavior can be explained by differences in methodology. The previous finding used solely HTTP requests generated by static content, while our network traces include all HTTP requests (including those generated by JavaScript) as well as HTTPS traffic.

[14]The Chi-Square statistical test is a non-parametric test that indicates whether the observed differences between categorical datasets are statistically significant. The null hypothesis of the Chi-Square test is that the differences between the datasets are not significant.

[15]The Kolmogorov-Smirnov statistical test is a non-parametric test that indicates whether the difference between the empirical distribution functions (ECDF) of two samples are statistically significant. The null hypothesis of the KS test is that the differences are not significant.

Clean Devices Access More			
Feature	**Cln. Median**	**Cmp. Median**	**Delta**
Business	22.36	20.14	2.22
Advertising	22.65	20.88	1.77
Marketing	12.96	11.66	1.3
Education	3.98	3.53	0.45
Content Server	6.96	6.58	0.38
Television & Video	2.18	1.89	0.29
Arts & Entertainment	2.54	2.27	0.27
Business Software	2.69	2.49	0.2
Web Design/HTML	1.39	1.24	0.15
Compromised Devices Access More			
Feature	**Cln. Median**	**Cmp. Median**	**Delta**
Computer Games	1.3	2.84	-1.54
Hobbies & Interests	2.61	3.78	-1.17
Uncategorized	26.25	26.97	-0.72
Technology	17.65	18.08	-0.43
Under Construction	5.33	5.65	-0.32
Network Security	1.43	1.65	-0.22
File Sharing	2.28	2.51	-0.23
News/Weather	2.44	2.64	-0.2
Illegal Content	0.15	0.33	-0.18

Table 5: Types of content accessed more by clean or compromised devices. We show the median fraction of registered domains accessed in the category for clean (Cln.) and compromised (Cmp.) devices, and delta in median.

5.4 Use HTTPS

Another recommended browsing behavior is to use HTTPS when available. Of course, it is the web site itself that ultimately determines whether HTTPS can be used: if a site does not support it, users have to use HTTP. However, since prior studies on device security behavior were not able to trace HTTPS traffic, we next examine HTTPS use and network activity more generally, and then examine how it correlates with device compromise.

For each device, we count the total number of distinct fully qualified domains it contacted using HTTPS and HTTP (approximating distinct sites visited). We then consider the number of distinct FQDNs contacted just using HTTPS divided by the total (HTTPS + HTTP) as the ratio of its HTTPS use. Since a recent study of HTTPS adoption on Chrome and Firefox showed that it depends on both browser and operating system [11], we similarly categorize first by dominant browser on the device (Section 5.2.2) and then OS. Table 6 shows the mean and median HTTPS use across all devices, browsers, and operating systems. As a point of comparison, HTTPS use among the devices in our population is roughly consistent with the results from [11]: devices contact sites via HTTPS 78% of the time on average, and HTTPS use is lower on Windows (74–76%) compared to Mac OS (79–80%). In terms of browsers, though, in our device population Chrome does not have a distinctly higher use of HTTPS for our metric.

Browser	OS	Mean (Median)
Chrome	Mac OS	78.6% (79.2%)
	Linux	78.5% (79.0%)
	ChromeOS	78.1% (78.3%)
	Windows	76.2% (76.2%)
Firefox	Linux	80.8% (80.3%)
	Mac OS	80.5% (80.7%)
	Windows	78.2% (79.0%)
Safari	Mac OS	80.5% (80.7%)
Edge	Windows	73.6% (74.0%)
All Devices		77.6% (78.5%)

Table 6: HTTPS use among devices.

Feature	P-value	Cln. Median	Cmp. Median
Unique HTTP FQDNs	< 0.001	705	1137
Unique HTTP RDs	< 0.001	375	522
Unique HTTP TLDs	< 0.001	27	36
Unique HTTP IP URLs	< 0.001	4	57
Unique HTTPS FQDNs	< 0.001	2.5k	3.1k
Unique HTTPS RDs	< 0.001	1k	1.2k
Unique HTTPS TLDs	0.001	49.0	57.0

Table 7: Differences in network usage for clean (Cln.) and compromised (Cmp.) devices. We use the KS test with Bonferroni correction to compare the ECDF of usage for each device type, and show the p-value and median values for each population.

Turning to security outcomes, we separate the activity of devices between HTTP and HTTPS traffic and calculate their distributions for compromised and clean devices at various aggregations: number of connections to all and unique URLs (for HTTP), unique fully-qualified domain names (FQDNs), unique registered domains (RDs), and unique top-level domains (TLDs). To identify significant differences in device behavior we use the KS test of statistical significance with Bonferroni correction. For each aggregation, Table 7 shows the p-value and the median values of the distributions for clean and compromised devices.

Overall, the ratio of HTTPS use is not strongly correlated with security outcomes.The connections made by compromised have similar usage of HTTPS and HTTP compared to clean devices that make similar number of connections. However, these results do show that devices that make more connections use HTTPS more than HTTP.

Across the board both kinds of devices generate more HTTPS traffic than HTTP, but the prominent trend is simply that compromised devices generate more web traffic than clean devices. To illustrate this point in more detail, Figure 5 shows the distributions of average weekly device web activity for clean and compromised devices. For every device, we count the number of fully qualified domains the device visits via HTTP and HTTPS combined per week, and normalize by averaging across all weeks that the device was active. Each bar in the histogram counts the number of devices that visit a given

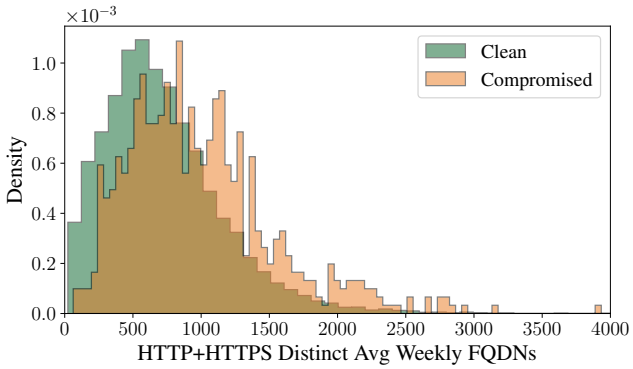


Figure 5: Distributions of average weekly device web activity for clean and compromised devices.

AV Name	*# Devices*	*%*
Avast	6,704	33.2%
Windows	5,752	28.5%
McAfee	3,659	18.1%
Avira	1,837	9.1%
Norton	866	4.3%
Other	1,383	6.8%

Figure 6: Five most prevalent antivirus products observed, with all others aggregated as "Other".

number of FQDNs per week, with 100-domain bins. The distribution for compromised devices is clearly shifted towards visiting more sites per week (and other traffic granularities show similar behavior). We interpret this result as just reflecting that more activity correlates to greater exposure and risk (much like automobile accidents).

5.5 Use Antivirus

Using antivirus software is a nearly universal recommendation. In fact, residential students on our campus are nominally required to have antivirus software installed on their devices to use the network. We crafted signatures to detect network activity (e.g., updates to software or the signature database, callbacks when scanning, etc.) for over a dozen antivirus products, and Figure 6 shows the distribution of popular products among our device population. If a device matched multiple signatures (e.g., Windows Defender and a third-party product), we counted the device in each category (hence the devices in the table sum to more than the unique device count). Avast, Windows Defender, and Avira are free, explaining their popularity among student devices.

Notably, while student devices technically need to have AV installed, regulations are not always followed. We verified that students can still access the residential network without antivirus installed by repeatedly using a mechanism for visitors, or lying about their device type (e.g., claiming a MacBook is an iPad), and 7.5% of our devices fall into this category.

Group	Feature	# Dev	P-value	w/ Feat.	w/o Feat.
All	Adobe AIR	826	< 0.001	10%	4%
All	P2P	2,237	< 0.001	13%	3%
All	Thunderbird	69	< 0.001	33%	4%
All	Uses Tor	321	< 0.001	12%	4%
All	Password Mgr.	434	< 0.001	8%	4%
All	Remote DNS	8,631	< 0.001	6%	2%
Win	Adobe AIR	490	< 0.001	13%	7%
Win	P2P	1,676	< 0.001	15%	5%
Win	Thunderbird	28	< 0.001	43%	7%
Win	Uses Tor	188	< 0.001	15%	7%
Win	Password Mgr.	262	0.001	12%	7%
Win	Remote DNS	5,249	< 0.001	8%	5%
Mac	Adobe AIR	336	< 0.001	6%	2%
Mac	P2P	541	< 0.001	7%	2%
Mac	Thunderbird	29	< 0.001	34%	2%
Mac	Uses Tor	123	< 0.001	7%	2%
Mac	Password Mgr	159	0.755	1%	2%
Mac	Remote DNS	3,212	< 0.001	3%	1%

Table 8: Software features across device populations correlated with compromise. For each feature we show the number of devices with the feature, p-value from the Chi-Square test, fraction of compromised devices with and without the feature. Compromise rates: All devices 4.5%, Windows devices 7.0%, and Mac OS devices 1.9%.

Using AV is strongly recommended to reduce risk. When focusing on Windows devices, interestingly a larger percentage (7%) of devices with antivirus are compromised compared to devices that do not have it (4%). By definition, though, most compromised devices in our population are those that were compromised by malware that antivirus did not catch.

5.6 Software Use

As discussed in Section 3.3, we extract a wide variety of features about the software used on devices observed on the network. We now explore how these software features correlate with a device being compromised. Since compromise depends on the operating system used (Windows devices are compromised more often than Mac OS devices), we also explore software features not only in the context of all devices but also individual operating systems.

For each correlated software feature, Table 8 shows the device population, fraction of compromised devices with the feature, and fraction of compromised devices without the feature. These results provide direct comparisons on compromise rates between devices with a particular software feature and without: e.g., devices using Tor are compromised 2–3.5× more often than devices that do not. To ensure that the comparisons are statistically significant, we use the Chi-Square test with Bonferroni correction since these are binary categorical features, and the very low p-values shown in Table 8 confirm significance.

Devices using some specific applications correlate very strongly with compromise, independent of operating system and network activity. Devices using Adobe AIR, P2P file sharing networks, Thunderbird, and Tor on average are much more likely to be compromised than devices that do not use such applications. Using these applications does indeed put devices at significantly more risk. The Thunderbird email client is particularly ironic since one reason why people use Thunderbird is because of its PGP integration [10]; yet, Thunderbird is rife with reported vulnerabilities (420 code execution vulnerabilities reported in CVE Details [8]).

Some of these software features do not directly lead to compromise, but instead indirectly reflect how attentive users are with respect to security. For instance, devices are not compromised due to using password managers or not, or whether they are kept updated, but the use of password managers does suggest that users are more security aware. We find the use of password managers to be correlated with compromise among the All and Windows device groupings. Similarly, users who explicitly configure their device to use a remote DNS server, instead of the DHCP default, reflect a certain degree of sophistication and confidence — for better or worse, considering that devices using remote DNS servers for resolution have a 1.6–3× higher rate of compromise.

6 RANKING FEATURE IMPORTANCE

Our analyses so far have focused on individual security practices. As a final step, we explore the relative importance of all the features we extract using statistical modeling, as well as the relative importance of features exhibited during the hour before a device is compromised. Our goal is not to train a general security incident classifier. Rather, it is to generate a logistic model that produces interpretable results for ranking the relative importance of our features.

6.1 Experimental Setup

Logistic regression is a statistical technique for predicting a binary response variable using explanatory variables [18]. We set the response variable to be whether or not a device is compromised, and use all of the device features we extract from the network as the explanatory variables. We first split the data into training (50%) and test (50%), and normalize the explanatory variables to have zero mean and unit variance.

To find the important explanatory variables we use a specialized type of logistic regression called least absolute shrinkage and selection operataor (LASSO), or L1 logistic regression, since we have a high number of explanatory variables. L1 logistic regression can be regularized to correct for overfitting, thereby preventing a model from becoming too closely tied to the data that it is built from. Regularization restricts the number of explanatory variables the model will use proportionally to how regularized the model is. The regularization parameter itself is configurable in the Scikit-learn machine learning framework we use [29].

To find the optimal regularization parameter we implement hyperparameter tuning: we build 200 models, each with a different regularization parameter, and identify the model that performs best. To identify the best model while avoiding selection bias, for each model, we perform 10-fold cross validation. We track the average area under curve (AUC) from the receiver operating characteristic (ROC) curves produced when predicting on the ten different validation data sets. We then select the regularization parameter

Group	Feature	Val AUC	Test AUC	Ratio
All	IAB Computer Games	+68.3%	+69.7%	2.2x
All	HTTP Reg Domains	+7.0%	+5.2%	1.6x
All	HTTP in TLD .cn	+2.3%	+3.7%	3.5x
All	Windows Antivirus	+1.9%	+1.1%	1.7x
Win	HTTP FQ Domains	+71.9%	+71.1%	1.6x
Win	IAB Computer Games	+4.2%	+2.9%	1.7x
Win	UA Str Safari	+2.2%	+2.5%	3x
Win	UA Str IE	+1.4%	+1.3%	1.1x
Mac	HTTP in TLD .cn	+76%	+76%	∞
Mac	UA Str IE	+5.3%	+4.3%	6.2x
Mac	HTTP Traffic at 2AM	+3.8%	-1.3%	0.9x
Mac	HTTP in TLD co.kr	+1.5%	+3.7%	1x
HTTP	IAB Shareware	+66.3%	+60%	∞
HTTP	UA Str IE	+7.2%	+7.9%	1.9x
HTTP	UA Str Android	+3.4%	+1.3%	2.2x
HTTP	Uses P2P	+1.0%	+2.7%	1.3x

Table 9: AUC gains from the top four features used to detect devices with security incidents. For each feature we also provide the ratio of median (continuous) or mean (categorical) values. Ratios > 1 (green) indicate that compromised devices exhibit more of the feature.

from the model that provides the maximum average validation AUC. After identifying the optimal regularization parameter we search for multicollinearity by computing the variance inflation factor (VIF) across features used in the model, and do not find features with a VIF greater than ten [20].

To compare the importance of each feature we implement a greedy deletion algorithm [16]. Our algorithm works in the following way: We start with the N important features used to predict security incidents identified by the best model (previous paragraph). For $N-1$ feature combinations we train regularized models with hyperparameter tuning. From the resulting models, we identify the model that has the maximum AUC (when predicting on validation data), and exclude the unused feature in the next iteration of the algorithm. We exclude the unused feature since it contributes least to the overall AUC compared to the other feature combinations. We repeat this process until we have a model that uses a single feature ($N = 1$); the remaining feature contributes the most to the AUC by itself and in the presence of other features. Finally, we interpret the results in terms of the changes to the test AUC when features are added to the final model.

6.2 All Features

We run the greedy deletion algorithm multiple times with different device groupings: all devices, Windows devices, Mac OS devices, and devices with on-median more HTTP traffic. We consider devices that produce on-median more HTTP traffic based on our observations in Section 5.4. Table 9 shows the top four features for each grouping, the feature's AUC contribution when predicting on validation and test data, and the ratio of the feature's median (continuous) or mean (categorical) value for compromised and clean devices. Since we select the feature combination with the highest validation AUC it is possible that adding in an extra feature will result in a small negative contribution to the test AUC (e.g., the "HTTP Traffic at 2AM" feature for Mac OS devices).

Feature	Val AUC	Test AUC
IAB Computer Games	+71.9%	+74.2%
IAB Web Search	+4.0%	+3.6%
IAB Illegal Content	+2.2%	+3.6%
IAB JavaScript	+1.0%	+0.1%
IAB Computer Networking	+0.7%	+0.1%
IAB Adult Content	+0.7%	+0.7%
IAB Shareware/Freeware	+0.7%	+0.4%
IAB Internet Technology	+0.5%	+1.5%

Table 10: AUC gains for the top eight features used to detect devices with security incidents one hour before compromise.

Our results indicate that behavioral features, regardless of device grouping, are most correlated with device compromise. In all cases, the first feature in each grouping relates to how much a device accesses web content or the type of content being accessed. Having Windows antivirus products (a proxy for using Windows, which has a significantly higher compromise rate), or using P2P applications are the only two software features in the top four of any grouping. Having the IE User Agent feature highly ranked highlights the challenge of cursory feature extraction. Applications can make use of embedded browsers, and examining traffic with an IE User Agent string shows many of the detections are actually from the QQ chat application and Qihoo 360 security product, not the IE browser. We also find that compromised devices, in the majority of cases (except for two features within the Mac OS grouping), exhibit more of each feature compared to clean devices.

6.3 One Hour Before Compromise

Lastly, we use our statistical model to examine the relative importance of security features focusing on the hour leading up to device compromise: Compared to devices that are not compromised, how are compromised devices behaving differently leading up to becoming compromised? For each compromised device, we extract their features from the hour before their first incident. To compare differences in behavior, we construct a synthetic control by taking a pseudorandom sample of clean devices. Specifically, for each compromised device we randomly select up to 300 clean devices that are (1) active in the same hour window, and (2) visit at least 50 distinct registered domains.[16]

Table 10 shows the most important features (relative to one another) for identifying compromised devices an hour before they are compromised. For our devices, the type of web sites visited (Section 5.3) are the most distinguishing features. On-average, compromised devices visit more web sites in each of the eight categories in Table 10 than clean devices. The most popular domains our devices visit in these categories do correspond well to the category domains. For some of the very generic labels, "Computer Games"

[16] On average compromised devices visit 50 distinct registered domains the hour before being compromised.

are gaming sites; "Computer Networking" include ISPs and IP geolocation services; "Internet Technology" include SSL certificate sites and registrars, etc.

7 CONCLUSION

The practice of cybersecurity implicitly relies on the assumptions that users act "securely" and that our security advice to them is well-founded. In this paper, we have sought to ground both assumptions empirically: measuring both the prevalence of key security "best practices" as well as the extent to which these behaviors (and others) relate to eventual security outcomes. We believe that such analysis is critical to making the practice of security a rigorous discipline and not simply an art.

However, achieving the goal of evidence-based security is every bit as formidable as delivering evidence-based healthcare has proven to be. In any complex system, the relationship between behaviors and outcomes can be subtle and ambiguous. For example, our results show that devices using the Tor anonymizing service are significantly more likely to be compromised. This is a factual result in our data. However, there are a number of potential explanations for *why* this relationship appears: Tor users could be more risk-seeking and expose themselves to attack, alternatively they might be more targeted, or there might be vulnerabilities in Tor itself. Indeed, it is even possible that Tor use simply happens to correlate with the use of some other software package that is the true causal agent.

Thus, while some of our results seem likely to not only have explanatory power but also to generalize (e.g., the use of Thunderbird and Adobe AIR, both historically rife with vulnerabilities, have significant correlations with host compromise), others demand more study and in a broader range of populations (e.g., why are gamers more prone to compromise?). Those results that lack simple explanations are a reflection of the complexity of the task at hand. Having started down this path of inquiry, though, we are optimistic about answering these questions because we have shown that the methodological tools for investigating such phenomena are readily available. We look forward to a broader range of such research going forward as our community helps advance security decision making from the "gut instinct" practice it is today, to one informed and improved by the collection of concrete evidence.

ACKNOWLEDGEMENTS

We thank our shepherd Walter Willinger and the anonymous reviewers for their insightful suggestions and feedback. We also thank Cindy Moore, Brian Kantor, Cooper Nelson, Nick Colias, Jim Madden, Michael Corn, Vern Paxson, Seth Hall, and Robin Sommer for their infrastructure support, collaboration, and guidance. This work was supported in part by NSF grants CNS-1629973 and CNS-1705050, DHS grant AFRL-FA8750-18-2-0087, and the Irwin Mark and Joan Klein Jacobs Chair in Information and Computer Science.

REFERENCES

[1] Apache Software Foundation. 2019. Apache Hive Website. https://hive.apache.org/. (2019).
[2] Apple. 2018. Update your iPhone, iPad, or iPod touch. https://support.apple.com/en-us/HT204204. (2018).
[3] Mihir Bellare and Phillip Rogaway. 2010. The FFX Mode of Operation for Format-Preserving Encryption. *Manuscript (standards proposal) submitted to NIST* (January 2010).
[4] Leyla Bilge, Yufei Han, and Matteo Dell'Amico. 2017. RiskTeller: Predicting the Risk of Cyber Incidents. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Dallas, Texas, USA.
[5] Davide Canali, Leyla Bilge, and Davide Balzarotti. 2014. On the Effectiveness of Risk Prediction Based on Users Browsing Behavior. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (CCS)*. Kyoto, Japan.
[6] Yannick Carlinet, Ludovic Mé, Hervé Debar, and Yvon Gourhant. 2008. Analysis of Computer Infection Risk Factors Based on Customer Network Usage. In *2008 Second International Conference on Emerging Security Information, Systems and Technologies*. Cap Esterel, France.
[7] Carrie Marshall and Cat Ellis. 2018. The best free password manager 2019. https://www.techradar.com/news/software/applications/the-best-password-manager-1325845. (2018).
[8] CVE Details. 2019. Mozilla Thunderbird Vulnerability Statistics. https://www.cvedetails.com/product/3678/?q=Thunderbird. (2019).
[9] DNSFilter. 2019. DNSFilter Website. https://www.dnsfilter.com/. (2019).
[10] The Enigmail Project. 2019. Enigmail — OpenPGP encryption for Thunderbird. https://www.enigmail.net/index.php/en/home. (2019).
[11] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel1, and Parisa Tabriz. 2017. Measuring HTTPS Adoption on the Web. In *Proceedings of the 26th USENIX Security Symposium*. Vancouver, BC, Canada.
[12] Firefox. 2019. How to stop Firefox from making automatic connections. https://support.mozilla.org/en-US/kb/how-stop-firefox-making-automatic-connections. (2019).
[13] Alain Forget, Sarah Pearman, Jeremy Thomas, Alessandro Acquisti, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, Marian Harbach, and Rahul Telang. 2016. Do or Do Not, There Is No Try: User Engagement May Not Improve Security Outcomes. In *Procedings of the 12th Symposium on Usable Privacy and Security (SOUPS)*. Denver, CO, USA.
[14] Aaron Gember, Ashok Anand, and Aditya Akella. 2011. A Comparative Study of Handheld and Non-handheld Traffic in Campus Wi-Fi Networks. In *Proceedings of the 12th International Conference on Passive and Active Measurement*. Berlin, Heidelberg.
[15] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. 2007. BotHunter: Detecting Malware Infection Through IDS-driven Dialog Correlation. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*. Boston, MA, USA.
[16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer New York Inc.
[17] Cormac Herley. 2009. So Long, and No Thanks for the Externalities: The Rational Rejection of Security Advice by Users. In *Proceedings of the 2009 Workshop on New Security Paradigms Workshop*. Oxford, United Kingdom.
[18] David W Hosmer Jr and Stanley Lemeshow. 2000. *Applied Logistic Regression* (2nd ed.). John Wiley & Sons.
[19] IAB. 2019. IAB Tech Lab Content Taxonomy. https://www.iab.com/guidelines/iab-tech-lab-content-taxonomy/. (2019).
[20] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.
[21] Moazzam Khan, Zehui Bi, and John A. Copeland. 2012. Software updates as a security metric: Passive identification of update trends and effect on machine infection. In *Proceedings of IEEE Military Communications Conference (MILCOM)*. Orlando, Florida, USA.
[22] Fanny Lalonde Lévesque, Jude Nsiempba, José M. Fernandez, Sonia Chiasson, and Anil Somayaji. 2013. A Clinical Study of Risk Factors Related to Malware Infections. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Berlin, Germany.
[23] Yang Liu, Armin Sarabi, Jing Zhang, Parinaz Naghizadeh, Manish Karir, Michael Bailey, and Mingyan Liu. 2015. Cloudy with a Chance of Breach: Forecasting Cyber Security Incidents. In *Proceedings of the 24th USENIX Conference on Security Symposium*. Washington, DC, USA.
[24] Microsoft. 2019. Microsoft Update Catalog. https://www.catalog.update.microsoft.com/Home.aspx. (2019).
[25] Mozilla Foundation. 2019. Public Suffix List Website. https://publicsuffix.org/. (2019).
[26] Neil J. Rubenking. 2019. The Best Antivirus Protection for 2019. https://www.pcmag.com/article2/0,2817,2372364,00.asp. (2019).
[27] ntop. 2018. PF_RING ZC (Zero Copy) Website. https://www.ntop.org/products/packet-capture/pf_ring/pf_ring-zc-zero-copy/. (2018).
[28] Vern Paxson. 1999. Bro: a System for Detecting Network Intruders in Real-Time. *Computer Networks* 31, 23-24 (1999), 2435–2463.
[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D.

Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[30] ProofPoint. 2019. ET Pro Ruleset. https://www.proofpoint.com/us/threat-insight/et-pro-ruleset. (2019).

[31] Redislabs. 2019. Redis Website. https://redis.io/. (2019).

[32] Elissa M. Redmiles, Sean Kross, and Michelle L. Mazurek. 2016. How I Learned to Be Secure: A Census-Representative Survey of Security Advice Sources and Behavior. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Vienna, Austria.

[33] Elissa M. Redmiles, Sean Kross, and Michelle L. Mazurek. 2017. Where is the Digital Divide?: A Survey of Security, Privacy, and Socioeconomics. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Denver, Colorado, USA.

[34] Elissa M. Redmiles, Sean Kross, and Michelle L. Mazurek. 2019. How Well Do My Results Generalize? Comparing Security and Privacy Survey Results from MTurk, Web, and Telephone Samples. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy*. San Fransisco, CA, USA.

[35] Robert Reeder, Iulia Ion, and Sunny Consolvo. 2017. 152 Simple Steps to Stay Safe Online: Security Advice for Non-tech-savvy Users. *IEEE Security and Privacy* 15, 5 (June 2017), 55–64.

[36] Armin Sarabi, Ziyun Zhu, Chaowei Xiao, Mingyan Liu, and Tudor Dumitras. 2017. Patch Me If You Can: A Study on the Effects of Individual User Behavior on the End-Host Vulnerability State. In *Proceedings of the 18th Passive and Active Measurement PAM*. Sydney, Australia.

[37] Yukiko Sawaya, Mahmood Sharif, Nicolas Christin, Ayumu Kubota, Akihiro Nakarai, and Akira Yamada. 2017. Self-Confidence Trumps Knowledge: A Cross-Cultural Study of Security Behavior. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Denver, Colorado, USA.

[38] Mahmood Sharif, Jumpei Urakawa, Nicolas Christin, Ayumu Kubota, and Akira Yamada. 2018. Predicting Impending Exposure to Malicious Content from User Behavior. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Toronto, Canada.

[39] Suricata. 2019. Suricata IDS Website. https://suricata-ids.org/. (2019).

[40] Samaneh Tajalizadehkhoob, Tom Van Goethem, Maciej Korczyński, Arman Noroozian, Rainer Böhme, Tyler Moore, Wouter Joosen, and Michel van Eeten. 2017. Herding Vulnerable Cats: A Statistical Approach to Disentangle Joint Responsibility for Web Security in Shared Hosting. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (CCS)*. Dallas, TX, USA.

[41] Update Google Chrome. 2019. Update Google Chrome. https://support.google.com/chrome/answer/95414?co=GENIE.Platform%3DDesktop&hl=en. (2019).

[42] Tom van Goethem, Ping Chen, Nick Nikiforakis, Lieven Desmet, and Wouter Joosen. 2014. Large-Scale Security Analysis of the Web: Challenges and Findings. In *Proceedings of the International Conference on Trust and Trustworth Computing*. Heraklion, Crete, Greece.

[43] Francesco Vitale, Joanna McGrenere, Aurélien Tabard, Michel Beaudouin-Lafon, and Wendy E. Mackay. 2017. High Costs and Small Benefits: A Field Study of How Users Experience Operating System Upgrades. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Denver, Colorado, USA.

[44] Rick Wash. 2010. Folk Models of Home Computer Security. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*. Redmond, Washington, USA.

[45] Rick Wash and Emilee Rader. 2015. Too Much Knowledge? Security Beliefs and Protective Behaviors Among United States Internet Users. In *Proceedings of the Eleventh USENIX Conference on Usable Privacy and Security*. Ottawa, Canada.

[46] Webshrinker. 2018. IAB Categories. https://docs.webshrinker.com/v3/iab-website-categories.html#iab-categories. (2018).

[47] Webshrinker. 2019. Webshrinker Website. https://www.webshrinker.com/. (2019).

[48] The Wireshark Team. 2019. Wireshark Website. https://www.wireshark.org/. (2019).

[49] Chaowei Xiao, Armin Sarabi, Yang Liu, Bo Li, Mingyan Liu, and Tudor Dumitras. 2018. From Patching Delays to Infection Symptoms: Using Risk Profiles for an Early Discovery of Vulnerabilities Exploited in the Wild. In *Procedings of the 27th USENIX Security Symposium (USENIX Security)*. Baltimore, MD, USA.

[50] Zeek. 2019. Zeek Protocol Analyzers Website. https://docs.zeek.org/en/stable/script-reference/proto-analyzers.html. (2019).