Prioritizing Remediation of Enterprise Hosts by Malware Execution Risk

Andrew Chi
Cisco Systems
San Jose, CA, USA
andrchi@cisco.com

Blake Anderson
Cisco Systems
San Jose, CA, USA
blaander@cisco.com

Michael K. Reiter
Duke University
Durham, NC, USA
michael.reiter@duke.edu

ABSTRACT

Defending an enterprise network requires making prioritization decisions daily; one is deciding which compromised hosts to remediate (reimage). We study the utility of endpoint monitoring data to perform this prioritization, with the driving goal being to minimize "regret" as measured by future (next-week) malware execution on hosts whose remediation was deprioritized. Leveraging data gathered by the vendor of a major endpoint protection product, we show that it is possible to prioritize remediation by training a classifier that predicts imminent malware execution. Perhaps surprisingly, while it might seem essential to maximize the amount of training data by collecting across an array of enterprises to which endpoint protection is deployed, at least in the case of the endpoint protection vendor (itself a major, worldwide company), predictive performance for a single enterprise can remain excellent when training is restricted to the enterprise itself. One advantage of single-enterprise training is the ease of combining different views of the hosts, such as via file-based and network-based monitoring. In the cases studied, although an exact comparison was impossible due to a time gap, the single-enterprise dataset with richer features resulted in superior prediction of malware execution compared to the multi-enterprise dataset.

CCS CONCEPTS

• **Security and privacy** → **Malware and its mitigation**; *Vulnerability management*.

KEYWORDS

malware, remediation

ACM Reference Format:
Andrew Chi, Blake Anderson, and Michael K. Reiter. 2023. Prioritizing Remediation of Enterprise Hosts by Malware Execution Risk. In *Annual Computer Security Applications Conference (ACSAC '23), December 04–08, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3627106.3627180

1 INTRODUCTION

The practice of defending an enterprise network involves prioritization at many levels. Each day, an enterprise's security team must choose which compromised hosts to remediate (reformat/reimage), which outdated software to patch, and which network connections to monitor with extra vigilance. This prioritization must be done with stringent constraints on the human and compute resources available that day. Complicating the situation, not all malware encounters warrant a disruptive remediation such as reimaging an employee laptop, which incurs substantial labor cost to both the security team and the employee. Only when malware can circumvent the automated defenses is it appropriate to apply labor-intensive remediation. Intuitively, the security team's goal is to minimize "regret," as measured by a combination of two factors held in tension: first, the failure to remediate a compromised host today that subsequently worsens, and second, causing unjustified business disruption. Typically, this means ignoring malware *encounters* on disk that are quarantined automatically, and reserving human intervention for hosts with malware *execution*, particularly when it is long-lived and persistently circumvents automated defenses.[1]

This study focuses on prioritizing defenses for the subset of the network directly visible from employee laptops (Fig. 1). Employee machines are often the primary client in enterprise networks and are a common entry point for attackers, who use their initial foothold on a compromised laptop as a "pivot" for attacking more critical servers. In environments with limited capacity for monitoring and cleaning infected hosts, it is inevitable that on any given day, mitigations can only be partially implemented. This work endeavors to inform one of those inevitable triage decisions in a data-driven manner by using telemetry from two existing host-based

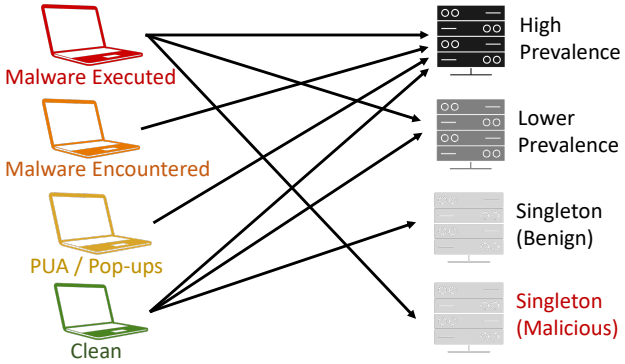


Figure 1: Overview. By assessing malware execution patterns and low prevalence network activity, defenders can prioritize hosts requiring remediation.

[1]We use the term "regret" intuitively. In some scenarios, regret can be quantified as a single number. In our context of remediation, a precision-recall curve more fully captures the cost tradeoffs.

ACSAC '23, December 04–08, 2023, Austin, TX, USA

ACM ISBN 979-8-4007-0886-2/23/12.
https://doi.org/10.1145/3627106.3627180

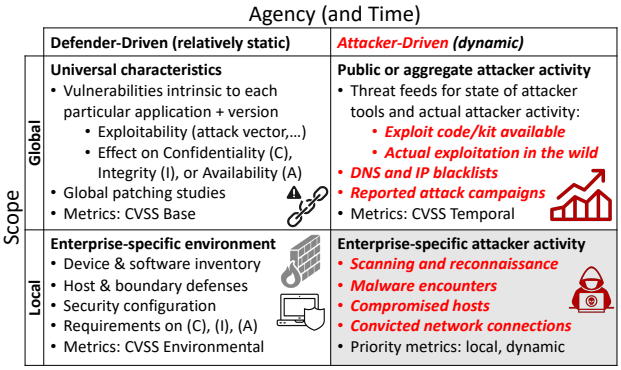


Figure 2: Scope-vs-agency: white boxes inform preventative hygiene; gray box responds to active infection (our focus).

monitoring systems in production environments. We investigate the question, "Which laptops are likely to be compromised?" The answer can be used not only to prioritize remediation, but potentially to enable mitigation strategies such as carefully monitoring network traffic to/from risky machines by redirecting their traffic through enhanced monitoring systems [7, 17, 44].

1.1 Background: Patching vs. Remediation

Prioritization is nontrivial. Security teams juggle many responsibilities, two of which are patching and remediation. Though often conflated, these distinct tasks are analogous to fireproofing and firefighting (prevention vs. treatment), which occur at different time scales and require different training. Our study focuses on remediation; nevertheless, we review both patching and remediation for completeness. Ultimately, security teams must address both.

Consider a problem that initially appears simple: deciding which software vulnerabilities to patch (i.e., which outdated software to upgrade). Large-scale patching can require significant effort and time [21], making it wise to patch important vulnerabilities first. To help with this process, most vulnerabilities are described by a Common Vulnerabilities and Exposures (CVE) entry, with an associated Common Vulnerability Scoring System (CVSS) Base Score that ranges from least to most severe (0.0 to 10.0). It may therefore seem reasonable to simply patch the CVEs with the highest score first. However, this ranking has been shown to be unrepresentative of which vulnerabilities were actually exploited in the wild [42] and in some cases little better than random [2], suggesting that a different ranking would provide better protection (e.g., EPSS [15]).

Why? Consider the taxonomy in Fig. 2, where the white boxes inform prevention, and the gray box represents treatment. A security vulnerability has multiple facets: the intrinsic mechanism of a vulnerability (universal unchanging facts), the existence and frequency of actual exploits in the wild, influenced by availability of easy-to-use exploit code (aggregate infection dynamics), and the specific host defenses and firewalls in an enterprise (local environment). Furthermore, if a host in the enterprise is actually compromised (local infection dynamics), then the security team must both remediate the host and identify the entry point to limit further damage.

Using the CVSS Base Score [9] as a standalone ranking is misguided; it covers only the upper-left box of Fig. 2. Proper use of CVSS considers temporal (upper-right) and environmental (lower-left) factors when informing preventative hygiene. But this work goes further: once an outbreak or infection occurs locally (gray box in Fig. 2), the required course of action is very different from preventative hygiene. Infected hosts must be quarantined or wiped. But importantly, the risk calculation changes significantly, reflecting the difference between the prior probability and posterior probability of infection once a confirmed case exists—others are more likely to be infected by transmission or by a similar vector. This is our focus.

1.2 Contributions

We analyze two real-world datasets, with a focus on remediation. (1) A multi-enterprise dataset of event logs from Cisco Advanced Malware Protection (AMP), an endpoint protection system for Windows and Mac hosts, which includes an antivirus engine. This data consisted of one year (2018-2019) of events generated by the AMP software on several million hosts in thousands of enterprise networks. As our primary indicator of host compromise, we used the execution of known malware, since it implies the attacker was in a position to cause harm. (2) A recent case study (2022-2023) of a single enterprise, including both detailed AMP records and host-based network flow records. The network records were collected by AnyConnect Network Visibility Monitor (NVM), a host-based network monitoring system deployed primarily on laptops. Our analysis showed the following:

(1) In the multi-enterprise dataset, it was possible during a typical week to predict the set of hosts that would execute malware in the next week with 92% precision and 27% recall (avgPrec=0.40, auROC=0.75). While this level of performance can be of some use, individual enterprises may struggle to obtain the data for training the classifier (thousands of enterprises).
(2) In the single-enterprise study, which had orders of magnitude fewer hosts but richer available features, it was possible to predict the set of hosts that would begin or continue to execute malware in the next week with 92% precision and 49% recall (avgPrec=0.68, auROC=0.89).
(3) In the single-enterprise study, it was possible to combine data feeds from file-based and network-based monitoring. By doing so, we found that network connections to low prevalence destinations and new public suffixes could serve as important supporting features when predicting malware execution, consistent with statistics on abused TLDs [32].

In summary, we use the history of malware execution to inform remediation in an enterprise setting. We show that in the case of one large, multi-national enterprise, it is possible to predict imminent malware execution with high precision (Sec. 6), enabling a security team to prioritize human interventions such as reimaging machines. While exact comparison between studies was impossible due to the time gap, our results suggest that it may not be essential to amass data across thousands of enterprises to anticipate compromises.

2 RELATED WORK

We conceptually organize related work using the taxonomy in Fig. 2. The vertical dimension represents global vs local scope, while the horizontal dimension roughly represents defender vs attacker

control (which also corresponds to rate of change: static vs dynamic). Our work focuses on using global and local malware execution data (right quadrants) to inform an enterprise's remediation actions (lower right).

The upper left quadrant (global, vulnerability-driven) could in principle include all basic research in identifying security vulnerabilities; we do not presume to survey the entire field. From a practitioner's perspective, the key centralized resource is the National Vulnerability Database (NVD) [22], which catalogues CVEs and CVSS Base Scores [9]. In addition, several works studied global patching behavior. Nappa et al. [21] found that it could take months for patching to reach 90% of hosts worldwide, while Sarabi et al. [27] showed that even updating quickly left a significant window of risk when a product exhibited new vulnerabilities frequently. Vaniea and Rashidi [39] studied user experiences when patching. Our work assumes enterprises are aware of vulnerabilities, but that patch deployment can take time.

The upper right quadrant (global, exploit-driven) includes aggregate compromises and publicly reported attack campaigns, as reported by threat and incident feeds such as SANS @RISK [26], Talos [38], the Internet Storm Center [25], VERIS Community Database (VCDB) [40], Symantec SecurityFocus [37] (contemporaneous with the historical dataset, but which has since been superseded), as well as DNS and IP reputation blocklists [33, 34]. Xiao et al. [42] provided early detection of which CVEs were actually exploited somewhere in the wild, using community detection to correlate the use of vulnerable applications (aggregated by ISP) against the rate of blocklisting of addresses in those ISPs. Jacobs et al. [14, 15] built a different model of which CVEs will be exploited and proposed it as a metric for prioritizing patching. Shen et al. [30] used aggregated Intrusion Prevention System (IPS) data to discover groups of CVEs that were often exploited together, possibly reflecting attack campaigns using a common exploit kit. Soska and Christin [31] predicted compromises of public websites that used a content management system such as WordPress. Liu et al. [19] correlated publicly declared breaches in VCDB against externally measurable symptoms such as DNS or BGP misconfiguration.

The lower left quadrant (local, vulnerability-driven) addresses the specific security posture of a single enterprise. Enterprises tend not to publicize their software inventory and host/network defenses, except when breaches occur [24]. But security controls defined by the Center for Internet Security (CIS) [6] and ISO/IEC 27002:2013 [13] have generally been accepted as best practice, and have been used by cyber insurance firms to assess their clients [41]. Focusing on the operations of security teams, Bollinger et al. [5] described practical heuristics ("playbook") to routinely scan for adversary behavior. Sundaramurthy et al. [35, 36] and Kokulu et al. [16] studied the human factors of several university and industry Security Operations Centers (SOCs).

We now survey the work in the lower right quadrant of Fig. 2, which is the quadrant most relevant to our work. DeKoven et al. [8] assessed the link between the lower left and lower right quadrants by monitoring traffic from a university residential network (15k hosts). Some correlations were intuitive (P2P and Tor correlated with compromise) but others were counterintuitive (password managers and antivirus also correlated with compromise). The surprising results reinforced the need for customized, enterprise-specific strategies. Both Yen et al. [43] and Bilge et al. [4] studied malware encounters (via file appearances) in enterprises, and the latter predicted which machines would be infected by malware sometime in the next 6 months. Our work differs in important ways: we use actual malware execution (not just file appearances) as a stronger, direct indicator of host compromise. We also require prediction in an actionable time frame (1 week), and we require high precision (high auROC is not sufficient), since remediation involves costly human intervention. Sharif et al. [28] studied HTTP traffic from a mobile ISP to predict whether a user was about to visit a malicious URL in the next 30 seconds. Their work showed that it might be possible to anticipate risky behavior on short time scales. Shen et al. [29] developed the Tiresias system, which used recurrent neural networks (RNNs) to predict future security events. Their work resembles ours, but an important difference is that Tiresias's training data and predicted events were mostly exploit *attempts* rather than successful exploitation. In our use case, a CSIRT would need to filter out nearly all of Tiresias's predictions to focus their limited (human) resources on actual compromises. The authors of Tiresias acknowledged that training for rare events is not a strength of RNNs; in our datasets, successful compromises are rare. Our work helps security teams prioritize remediation efforts by anticipating *successful* compromise, specifically future (next-week) malware execution, and does so in a human-actionable time frame.

3 DATASETS

We studied two host-based datasets. The first was a historical (2018-2019), large-scale anonymized dataset covering an array of enterprises, consisting of events collected from the Cisco Advanced Malware Protection, a file-oriented endpoint protection system (antivirus) for Mac and Windows hosts. The second represented a recent (2022-2023) case study of a single enterprise (Cisco itself), and was comprised of data from two host-based systems: events from the endpoint protection system, and network flow records from a host-based network monitor installed on the same hosts.

3.1 Multi-Enterprise Dataset (2018-2019)

The first dataset was a large, multi-enterprise dataset, which consisted of events generated by the Cisco Advanced Malware Protection (AMP) agent software on 12.8 million hosts in 9206 businesses from September 1, 2018 through August 31, 2019.[2] For this dataset, we defined a "business" to be an AMP administrative domain that could represent either an entire enterprise or a subset of an enterprise; for both simplicity and anonymization, we did not distinguish these cases. Although the businesses were anonymized, we assessed the similarities and differences among the enterprises through the lens of their AMP agents—in particular, their statistics on vulnerabilities and compromises (see Appendix A.1). The AMP events comprised 96 event types ranging from administrative logging to high fidelity indicators of compromise: examples include the start/completion of virus scans, maintenance notifications, policy updates, observations of vulnerable software being run, and detection of malicious activities such as malware execution and

[2]All analysis for the multi-enterprise dataset was performed shortly after the data was collected. Due to organizational changes, the authors no longer had access to this data during the single-enterprise study.

connections to blocklisted IP addresses. A few event types in the AMP dataset are shown in Table 1; see Table 10 for a more complete version. Not all event types were present for all businesses, partly because each administrative domain could decide which events were activated and reported.

Table 1: Sample AMP Event Types

Hosts (%)	Events (%)	Event Type
81.593	4.423	Policy Update
66.886	10.011	Computer Metadata Changed
44.441	4.305	Scan Started
41.938	4.253	Scan Completed No Detections
...	...	...
30.349	0.983	Vulnerable Application Detected
2.165	0.028	**Executed Malware**
0.503	0.735	Network Conviction

Our work incorporated statistics on all event types (see Sec. 6). However, we highlight the most important one here: Executed Malware. We used the Executed Malware (ExecMalw) event as our primary indicator that the host was compromised. That is, even if AMP blocked the reported execution, the fact that it was even attempted means that the attacker was already in a position to cause harm, e.g., by taking a less noticeable action such as exfiltrating a file or by executing newer malware not yet recognized by AMP. Moreover, whether the attacker obtained access via social engineering or technical means, it is likely the attacker could have done so again on the same system or similar ones. We assume ExecMalw events are ground truth.

Table 2: Top AMP Vulnerable Applications

Hosts (%)	Program
9.85	Google Chrome
8.61	Oracle Java(TM) Platform SE
6.38	Adobe Acrobat Reader
4.32	Mozilla Firefox
3.37	Microsoft Office
3.13	Adobe Flash Player
2.81	Microsoft Internet Explorer
0.61	Microsoft Silverlight
0.27	Adobe Acrobat
0.02	OpenSSH

In addition, we note one other event type that plays an important role as a supporting feature: Vulnerable Application Detected (VulnApp), which triggers when the host has executed a vulnerable application, such as an out-of-date web browser. Each program executed was queried (via hash) in a vulnerable application database, and an alert was generated if there was an associated CVE.[3] The VulnApp alerts did not cover all possible applications, but rather focused on several common ones (Table 2). In addition, VulnApp was configured by default with a suppression interval of 7 days; if the same vulnerable application was run more than once within the same week, only a single event was emitted. Note that VulnApp is *not* an indicator of compromise, but simply declares a potential weakness, which may or may not be exploitable in the local environment. Nevertheless, as will be seen in our results, VulnApp events can serve as important supporting features in predicting future compromise. We found it useful to split these events into multiple features: one feature for each vulnerable application in Table 2.

[3]The vulnerability database itself is generated using a system by Grieco and O'Donnell [10] that installs known versions of applications on a test machine, which then records hashes of any installed files. The known versions are cross-referenced with the NVD [22] to establish vulnerability status.

3.2 Single-Enterprise Dataset (2022-2023)

The second dataset was a case study of a single enterprise, Cisco itself, from January 17, 2022 through January 29, 2023. Working closely with the security team, we obtained detailed AMP records from about 41,000 hosts, including an extra detail not available in the historical dataset: the malware name, for each event that represented a malware encounter or execution.

This single-enterprise case study also included a second data source: Cisco AnyConnect Network Visibility Monitor (NVM), which was comprised of records of network connections made by Windows and Mac employee laptops, with the notable feature that each flow was associated with the endpoint software that made the connection. Specifically, the NVM telemetry is collected by a host-based network monitor that records statistics of connection "flows" as defined by the standard 5-tuple: TCP/UDP, source IP and port, destination IP and port. Each connection is associated with a set of features including the number of incoming and outgoing bytes, the process making the connection, the parent process, and metadata about the host. For this work, we use the restricted and anonymized subset of NVM fields shown in Table 3.

Table 3: Sample NVM Anonymized Flow Record

Field Name	Sample	Explanation
uuid	[anonymized]	unique per endpoint
dst_ip	2001:DB8:2::a	destination IPv4/v6
dst_name	www.cisco.com	forward DNS
dst_port	443	TCP or UDP port
L4_proto	6	6 = TCP, 17 = UDP
event_start	1576563894	Unix timestamp
event_end	1576563901	Unix timestamp
bytes_client	15230	downloaded to local
bytes_server	5921	uploaded to remote
process	firefox	filename of executable
sha256	3559E2F1...	hash of executable
p_process	launchd	parent filename
p_sha256	066D2A82...	parent hash

Note that in contrast to AMP, which focuses on potentially harmful events such as indicators of compromise or vulnerabilities, the NVM dataset captures a summary of network connections, the vast majority of which are presumably not malicious. Moreover, NVM was only available on a subset of the AMP hosts (22k out of 41k), and only for half of the AMP duration: July 12, 2022 through January 25, 2023. Nevertheless, the NVM data was sufficient to study

the effect of combining network features with the existing AMP features on a limited subset of hosts.

3.3 Ethical Considerations

All endpoint and network data was collected according to the privacy and data protection policies of the authors' institutions.[4] The datasets were analyzed within a platform designed to restrict access to a subset of Cisco employees with proper oversight and valid justification to analyze the data.

Since the datasets contained potentially sensitive corporate and personal information, further steps (beyond access control) were taken to anonymize the multi-enterprise dataset. All telemetry data in the multi-enterprise dataset was aggregated or de-identified such that it could not reasonably be used to identify an individual or entity such as a business. As a preprocessing step, the AMP data was scrubbed in three steps: (1) anonymize all fields that identify a user, (2) anonymize all fields that identify an organization, and (3) remove all fields not critical to the analysis. The anonymized or removed fields included MAC addresses, source IP addresses, usernames, hostnames, full directory paths, and other host-specific identifiers that were used internally by the capture mechanism. The anonymization itself was performed using a keyed hash (HMAC-SHA256), prior to any data processing.

4 LIMITATIONS

We discuss limitations of this study. First, due to limited data retention and access policies, the multi-enterprise dataset was analyzed shortly after data collection, and was no longer accessible to the researchers during the time of the single-enterprise case study. Thus, Sec. 6.1 reflects the analysis as completed at the time of original data collection (2018-2019), which unfortunately could not be revisited for further investigation. This means that the remediation study's comparison between the multi-enterprise and single-enterprise datasets uses similar but not identical experimental setups. Second, we assume that malware executions and retrospective detections reported by AMP are ground truth (though our results do not depend on other AMP detections to be correct), and that manual intervention by the security team is required if and only if these events persist over several days. A given enterprise security team may differ from this admittedly simplified intervention policy. Third, the NVM sensor does no inspection of network packet payloads; it simply associates network flows (5-tuples) with the program on the host that made those connections. This study ignores any host compromises that would have been detected via network traffic signatures alone. Moreover, we do not use IDS/IPS information that would normally be available to a security team; our analysis is entirely host-based, and is therefore blind to activity on hosts (including most servers) that lack the AMP and NVM agents.

Finally, an important caveat is that both malware and malware detection are moving targets. Between the time frame of the multi-enterprise dataset and the single-enterprise case study, the AMP detections shifted from almost entirely traditional signatures toward an increased number of sandbox-based detections, presumably due to new malware (e.g., polymorphic) designed to evade traditional signatures. In order to accommodate this shift, we adjusted the the definition of compromise to include both classic detections of malware execution and "retrospective detection." A retrospective detection occurs when a file that was observed at time t_0 is determined at time $t_1 > t_0$ to have been malicious, usually after submission to a malware sandbox. The connection between malware execution and retrospective detection is that both are cases where the malware likely evaded the automated defenses, at least for some time. Retrospective detection is, with high probability, implied malware execution. We believe our adaptation is a good approximation of "typical malware detections" during both the multi-enterprise and single-enterprise time frames. Nevertheless, it is important to acknowledge that what constitutes a typical malware detection (even by the same antivirus vendor) is a definition that changes over time.

5 EXPERIMENTAL DESIGN

This section provides a high level, intuitive overview of the experiments in this paper. Our objective was to create a classifier that would predict which hosts would execute malware in an actionable time frame (e.g., next week). This enables a security team to allocate (human) resources to the task of investigating and remediating hosts, with priority given to the highest risk hosts. Note that in contrast to a patching study that might focus on the presence or absence of vulnerabilities as their main features, for remediation we used all features available to us from local machines (put another way, vulnerabilities only partially explain malware execution). We created classifiers for both multi-enterprise and single-enterprise datasets, and measured the performance of the classifiers using precision/recall (PR) curves.

Why is prediction needed? Why not simply prioritize remediation of hosts that have recently executed malware? There are two reasons. First, the strategy is reactive: malware often acts in multiple stages, and malicious network activity that finds an entry point for installing malware could precede any file-based malware detection by AMP. Predicting ahead at 1-week granularity provides an opportunity to be proactive—it provides time to configure advanced network monitoring and other extra defenses that cannot be applied to all hosts. Second, there will always be some acute and some chronic infections: typically, in the multi-enterprise dataset, about 40% of hosts that executed malware in week N do not execute malware in week $N+1$. We do not blindly advise security teams to drop their existing priorities in order to manually clean any host that exhibits at least one malware execution. Simply remediating all of last week's infected hosts (even assuming zero false alarms) would waste 40% of the effort and cause business disruption in those cases, as the malware would have been resolved by automated defenses. For these reasons, we aim to predict infected hosts in week $N+1$, and do so whether or not they were infected in week N.

6 PREDICTION EXPERIMENTS AND RESULTS

Sec. 6.1 and Sec. 6.2 detail the experiments and results for predicting which hosts will require remediation using the multi-enterprise and single-enterprise datasets, respectively. We then discuss the implications of those results in Sec. 7.

[4]Our case involved two institutions. The second institution's IRB ruled that in this particular scenario, the primary institution had sole responsibility of ensuring the appropriate approvals to conduct this research.

6.1 Multi-Enterprise Dataset

We now describe the experimental setup for the multi-enterprise dataset. We cast the prediction as a standard binary classification problem. Given as input the AMP data for a single host during week n, we would like to predict a binary indicator: will this host execute malware during week $n + 1$? As an example, if n is 2019wk24 (the week beginning 2019-06-10), our training and test data would be as follows.

- Training: For each host, the AMP event counts from 2019wk23 (begins 2019-06-03), plus a binary indicator of whether it emits an ExecMalw event during 2019wk24 (begins 2019-06-10).
- Testing: For each host, the AMP event counts from 2019wk24 (begins 2019-06-10). The classifier attempts to predict the binary indicator of whether each host emits an ExecMalw event during 2019wk25 (begins 2019-06-17).

We use ExecMalw events as our definition of compromise. For predictors, we used 104 features: 22 vulnerable application event types (see Fig. 2), plus 82 other AMP event types available to us (see Table 10), including previous indicators of compromise, antivirus file quarantines, and typically benign events such as computer metadata changes. We performed minimal feature engineering, and no feature selection; the results therefore reflect "out-of-the-box" performance of our methodology on typical AMP data. The features were simply the weekly event counts, e.g., the total number of malware execution events, ignoring details like malware names.

Unsurprisingly but importantly, the class imbalance for this task is extreme. Clean hosts vastly outnumber compromised hosts; in the example above, the distribution was 2,513,153 (99.78%) negatives to 5,650 (0.22%) positives. For such imbalanced classes, our choice of performance evaluation metric can significantly impact operational utility [12]. In this domain, receiver operating characteristic (ROC) curves are popularly used to evaluate classifier performance, but many classifiers with impressive ROC curves actually produce too many false alarms to be useful, due to the base-rate fallacy [3]. To illustrate this, consider a classifier with a near-perfect ROC curve that, at its chosen operating point, perfectly detects all compromises (true positive rate = 100%) while admitting only a 1% false alarm rate. In our particular situation, this classifier would return 5,650 true detections and over 25,000 false alarms. According to Bollinger et al. [5], such a classifier would be of little practical use to an incident response team, since the cost of investigative reports is dominated by the time spent by analysts trying to find actionable, true positive events. Security teams therefore prefer "high-fidelity reports" which have essentially zero false positives, or "investigative reports" where true positives significantly outnumber false positives, and usually retire a report when this no longer holds.

With these practical considerations in mind, instead of ROC, we choose the precision-recall (PR) curve as our primary evaluation metric, with a particular focus on maximizing precision at the left-most (low recall) portion of the curve. Intuitively, we are preferring a search engine that returns high quality results on its first page, even if this results in a tradeoff where results further down the list are of inferior quality. (The "Precision at K" metric is used extensively in information retrieval [1].) Such a PR curve is desirable because it enables the choice of a high-fidelity operating point: e.g., the classifier reports only the top 100 hits, 95 of which are true positives. Even if such a classifier has an inferior ROC curve and can only predict 30% of the compromised hosts when evaluated on the entire dataset, there exists a threshold that a security team might find useful in practice.

We trained a few standard classification models on the AMP prediction task. We had three requirements in our choice of model. First, the model must be straightforward for a security team to implement out-of-the-box. Second, the result must be interpretable, as security teams must produce incident reports explaining the reason any detection(s) triggered on a particular host [5]. Third, as discussed above, the model needs to be evaluated on its PR curve, which means that the detections must be ordered. One way to do this is to require the classifier to output a real number such as a probability. This enables a PR curve to be produced by sliding the detection threshold between 0 and 1.

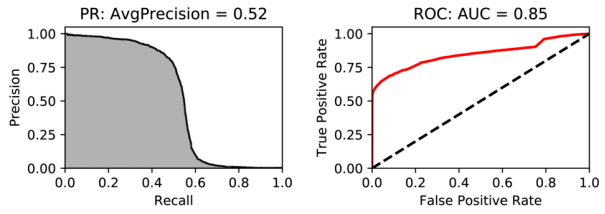
Based on these requirements, we chose to build the following classification models using scikit-learn [23] and Spark ML [20]: logistic regression, k-nearest neighbors (kNN) with $k = 3 \ldots 11$, and random forest[5] with 10, 50, and 100 estimators, adjusting hyperparameters until performance stopped improving. (Linear SVMs in scikit-learn failed to converge on our dataset; we suspect the decision boundary is highly non-linear.) For dataset balancing, we compared the results when using all the data against the results when downsampling the training set to achieve a negative to positive class ratio of 10:1. Overall, we found that random forests achieved the best performance on our dataset.[6] We also observed that in this dataset, using all available data for training was superior to downsampling, despite the extreme class imbalance.

Fig. 3 shows the performance of the best classifier of our set, random forest with 100 estimators, over five consecutive weeks of the AMP telemetry. These five weeks were chosen in order to exemplify both the typical classifier performance (2019wk7, 2019wk10, and 2019wk11), as well as two of the worst performing weeks of the year (2019wk8, 2019wk9). Examining one of the typical weeks, 2019wk10, the classifier achieved an average precision of 0.40, with precision reaching 1.0 at the leftmost (low recall) edge of the curve. A usable operating point for this classifier might be a threshold of 1000 detections (prec=0.98 and recall=0.10). At this operating point, only 978 of the 10054 compromised hosts are reported by the classifier, but the precision is high enough to be of immediate practical use. Depending on the application, a threshold of 3000 detections (prec=0.92, recall=0.27) may also be acceptable. If we permit 50% precision, perhaps as an investigative report performed by an analyst with extra time, the classifier covers around 40% of the compromised hosts.

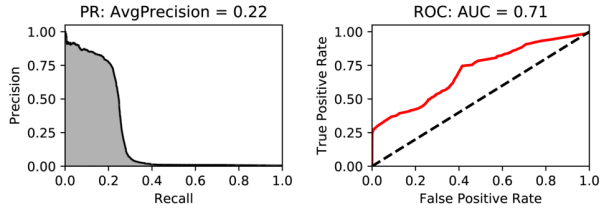
The two poorly performing weeks (2019wk8, 2019wk9) in Fig. 3 occurred at the time of a sudden spike in the number of compromised hosts in the AMP population. The compromise count doubled one week and then returned to normal the next week. Spikes can occur for several reasons: an attack campaign that both spreads quickly and is resolved quickly, a false-positive AV signature that is subsequently corrected, or a new true-positive AV signature that detects malware that has been on a large number of hosts for some

[5] A random forest works by voting among the individual decision tree estimators. One way to output a score between 0 and 1 is to report the fraction of estimators that gave an output of 1 (compromised host).

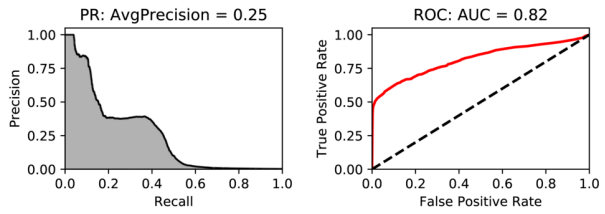
[6] Spark random forests did not support cost-sensitive training.



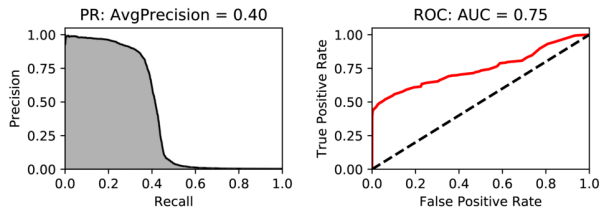
(a) 2019wk7, begins 2019-02-11



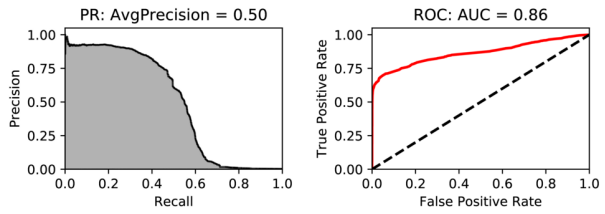
(b) 2019wk8, begins 2019-02-18



(c) 2019wk9, begins 2019-02-25



(d) 2019wk10, begins 2019-03-04



(e) 2019wk11, begins 2019-03-11

Figure 3: Classifier performance for 1-week-ahead prediction of malware execution, shown over 5 consecutive weeks. Most weeks resemble 2019wk7, 2019wk10, and 2019wk11. A sudden spike in the number of compromised hosts (which is resolved by the next week) resulted in what appears to be under-prediction for 2019wk8 and over-prediction for 2019wk9.

time. After examining the distribution of raw prediction probabilities from the classifier as compared to typical weeks, we believe the classifier performance reflects an under-prediction for 2019wk8 and over-prediction for 2019wk9. This type of event occurred a few other times, as shown in Fig. 4, which summarizes the performance trends over the course of the year for the following metrics: average precision, area under ROC, precision at 1000, and precision at 3000. But we observed that even during weeks with poor performance, precision at 1000 tended to remain at 0.8 or higher, a potentially useful operating point.

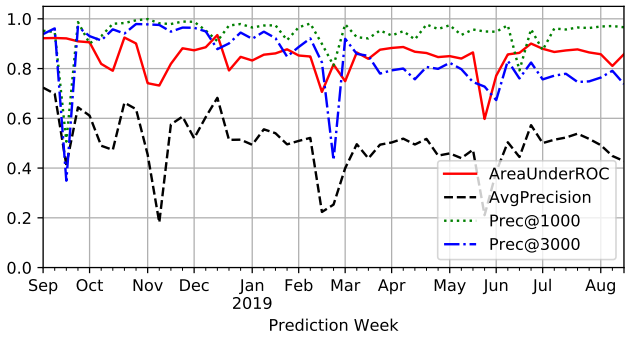


Figure 4: Performance trends over the course of a year, for 1-week-ahead prediction of malware execution using a random forest classifier.

We now examine the classification model produced by the random forest classifier for one of the more typical weeks (2019wk24, avgPrec=0.5, auROC=0.86). Table 4 shows the top feature importances of the random forest classifier. Feature importances are computed by the mean decrease in Gini impurity provided by that feature over all estimators (trees). They are normalized such that the sum of all feature importances equals 1, and in Table 4, the rightmost column is simply the cumulative sum of the "importance" column. Note that feature importance can be misleading when features are highly correlated, as it can spread the importance across the similar features. Nevertheless, we see some notable patterns that we list here. One of the top features for predicting future malware execution is a past history of malware execution. (We discuss implications of this in Sec. 7.) The second strongest feature is "Threat Detected in Exclusion." These are files originally marked malicious by AMP, but which the security team has explicitly allowed. The presence of these files is apparently still a predictor of imminent malware execution. Next on the list is "File Detection," which indicates that a known-malicious file has appeared on the system, but without an attempt (yet) to execute it. The next indicator, "Computer Metadata Changed," is triggered whenever there is a detectable change in system metadata such as hostname or network interface card, for example when the IP address changes. "Threat Quarantined" is the event that occurs whenever a routine virus scan encounters known malware on disk and places it in a file quarantine. Only lower in the list do the vulnerable applications begin to appear, beginning with vulnerabilities in Adobe Acrobat Reader.

Table 4: Feature Importances for Multi-Enterprise

Feature (Event Type)	Importance	$\sum_{i=1}^{k}$
Executed Malware	0.2649	0.2649
Threat Detected in Exclusion	0.0805	0.3454
File Detection	0.0662	0.4116
Policy Update	0.0638	0.4753
Computer Metadata Changed	0.0543	0.5297
Failed to Delete from Quarantine	0.0529	0.5826
Attempting Quarantine Delete	0.0524	0.6350
Low Prevalence Execution	0.0352	0.6702
Threat Quarantined	0.0323	0.7025
Generic IOC	0.0301	0.7326
Quarantine Failure	0.0251	0.7577
Repeated Malware Download	0.0226	0.7803
Scan Started	0.0186	0.7989
Scan Completed No Detections	0.0178	0.8167
Vulnerable Application Detected	0.0177	0.8344
Vulnerable Adobe Acrobat Reader	0.0154	0.8499
Quarantined Item Deleted	0.0149	0.8647
Network Conviction (NFM Det.)	0.0106	0.8754
Execution Blocked	0.0092	0.8845
Retro Quarantine of False Negative	0.0090	0.8935

We note three side experiments. First, to investigate the effects of business size, we examined the precision-recall curves of groups of businesses: small (under 1000), medium (1000 to 10000), large (over 10000). While the large group had the greatest average precision (median 0.61), surprisingly, the small group (median 0.45) slightly outperformed the medium group (median 0.41). It appears that large businesses benefit the most from an extensive dataset, but small and medium businesses may, hypothetically, compensate somewhat by learning from each other. Second, since the goal was to predict compromise in week $n+1$, it seems reasonable to train not just on week n, but on weeks $n-1, n-2, \ldots$. In our experiments, we noticed a slight benefit when adding a few weeks of historical data, but performance *decreased* when training over several months. We hypothesize that this is due to the non-stationarity of the features that predict compromise over time. Third, there appears to be an inherent upper bound on recall for the multi-enterprise dataset. A sizable fraction of hosts that executed malware each week had classifier prediction probability ≈ 0, meaning that based on the previous week's AMP features, they were indistinguishable from the large majority of clean hosts. This may be a limitation of the model, or of the AMP features available in the multi-enterprise dataset.

6.2 Single-Enterprise Dataset

In the previous subsection, we predicted host compromise (malware execution) across a large array of enterprises. While the results showed promise, the approach is unfortunately not useful to a typical enterprise for at least two reasons: most enterprises do not have access to data across thousands of other businesses, and the classifier is optimized for overall performance, not performance in a particular enterprise of interest. Thus, in this section we describe our second dataset: a single enterprise case study (Cisco itself). Note that the large historical dataset occurred in 2018-2019, while the single-enterprise case study covered 2022-2023, which means that there was some evolution in both the malware landscape and the endpoint protection software. Therefore, in addition to examining multi-enterprise vs. single-enterprise scenarios, this also tested whether the predictive approach remained effective after a several year gap.

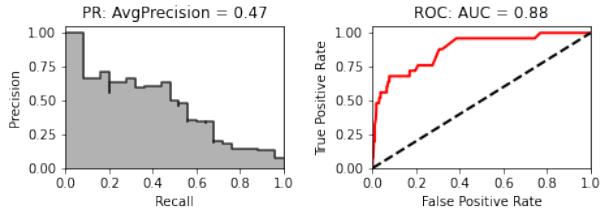
Our experimental setup for the single-enterprise dataset was similar to the large historical dataset in Sec. 6.1, with slight improvements for operational relevance. For each host, we aggregated AMP/NVM data across two-week sliding windows (6 days in the past, today, 7 days in the future), where the classification task was to use the past week (today and 6 days past) to predict whether a compromise would occur any time in the next 7 days. We narrowed the scope to hosts with AMP detections but not necessarily compromises: for example, a JavaScript pop-under that occurs when visiting a website is a detection but not a compromise, as it does not by itself warrant human intervention. We omitted hosts with absolutely zero AMP activity, since a remediation action could not be justified without any evidence of potentially suspicious activity. In order to accommodate the shifting landscape of malware and malware detectors from traditional signatures toward more sandbox-based detections, we adjusted the the definition of compromise to include detections of both malware execution and sandbox-based "retrospective detection" (see Sec. 4).

Where possible, we also added host-based network telemetry data from the NVM system. Based on an actual example of compromise (malware that performed DLL injection) that was observed by the security team during the study, we hypothesized that malware was more likely than other software to connect to low prevalence network destinations and may serve as a supporting feature. On the other hand, there was a long tail of benign low prevalence activity: on any given day, almost 1/3 of all destination domains visited were singletons (i.e., only one host connected to that domain), most of which were benign. We therefore restricted the set of singletons to those that were not already present the previous day, resulting in the following "new singleton" features: file hash, process name, destination domain (coarsened to the first private suffix, e.g., mail.google.com is truncated to google.com), and destination IP subnet (/24 for IPv4 and /48 for IPv6). Finally, we added the per-host count of new public suffixes that were not present the previous day.

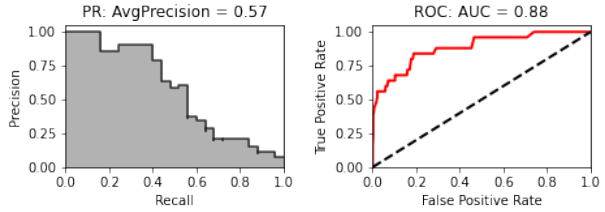
Combining AMP and NVM data for each host and time point, we created a feature vector containing the following:

- AMP: Histograms of events and detected malware names for the past 6 days and the current day (histograms were encoded as a count per feature using scikit-learn DictVectorizer):
- NVM: per-host counts of new singletons of the following types: hash, process name, IP subnet, domain
- NVM: per-host counts of new public suffixes (e.g., .com, .co.uk)

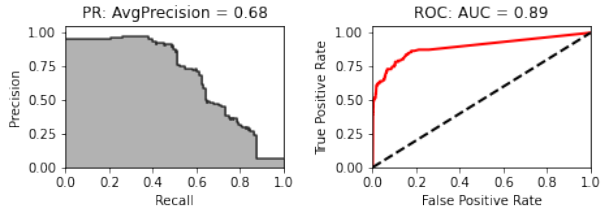
We set up several experiments. Due to heterogeneous deployment of AMP and NVM across managed hosts and limitations on data availability, only 22k out of 41k hosts with AMP detections had corresponding NVM data, and that data was available only for half the duration AMP. Nevertheless, the available data was sufficient to measure the effect of training data length, number of hosts, and the presence or absence of NVM features. We ran the following



(a) 6-month AMP (22k hosts), test begins 2023-01-02



(b) 6-month AMP + NVM (22k hosts), test begins 2023-01-02



(c) 12-month AMP (41k hosts), test begins 2023-01-02

Figure 5: Classifier performance for 1-week-ahead prediction of malware execution in a single enterprise, with varying features and training time. All test sets were the same time period, beginning 2023-01-02. Note that while ROC curves must be monotonic, PR curves can fluctuate, e.g., when a few false positives are followed by many true positives.

3 experiments, training a binary classifier to predict whether a compromise (malware execution or retrospective detection) would occur in the next 7 days.

- **6-month AMP**: 22k hosts with both AMP and NVM, using AMP features only
- **6-month AMP + NVM**: 22k hosts with both AMP and NVM, using both AMP and NVM features
- **12-month AMP**: 41k hosts, using AMP features only

In all three cases, the training and the testing data intervals were separated by a 1 week gap. In each case, we trained a balanced random forest classifier [18] with 100 estimators, where we randomly under-sampled the majority class for each bootstrap sample in order to balance it. All models trained in less than an hour of compute time.

We show the results of all three experiments, including the precision-recall (PR) and receiver operating characteristic (ROC) plots in Fig. 5. In addition, we list the top features for each of the three experiments in Table 5, Table 6, and Table 7.

Table 5: Feature Importances for 6-Month AMP

Feature (Past/Current Event or Detection)	Importance
current_detect__JS:Adware.Popunder.G	0.059004
current_event__Threat Detected	0.050519
past_compromise	0.035114
past_detect__JS:Adware.Popunder.G	0.034255
current_compromise	0.033429
current_event__Cloud IOC_Executed Malware	0.031082
current_detect__JS:Adware.Popunder.D	0.030757
past_event__Threat Detected	0.022609
past_event__Retrospective Detection	0.021819
past_event__Cloud IOC_Executed Malware	0.019762
past_detect__JS:Adware.Lnkr.L	0.012575
current_detect__JS:Adware.Lnkr.L	0.011596
past_detect__JS:Adware.Popunder.D	0.011395
current_event__Retrospective Detection	0.010339
current_detect__W32.File.MalParent	0.008382

Table 6: Feature Importances for 6-Month AMP + NVM

Feature (Past/Current Event or Detection)	Importance
current_detect__JS:Adware.Popunder.G	0.048932
past_compromise	0.045179
current_event__Threat Detected	0.044980
current_compromise	0.038727
new_public_suffixes_count	0.036172
past_detect__JS:Adware.Popunder.G	0.031208
current_event__Cloud IOC_Executed Malware	0.030402
new_singleton_subnets_count	0.028367
new_singleton_domains_count	0.028246
current_detect__JS:Adware.Popunder.D	0.024364
past_event__Retrospective Detection	0.024230
past_event__Threat Detected	0.023441
past_event__Cloud IOC_Executed Malware	0.016360
new_singleton_hashes_count	0.013341
past_detect__JS:Adware.Popunder.D	0.012955

To illustrate the contribution of low prevalence ("singleton") network features in Table 6, we describe two examples that actually occurred. Low prevalence features are a type of anomaly detection, but in our use case, are not sufficient by themselves. For example, one of the authors visited `farmsidekitchen.com` (a local restaurant), and was the only employee in the company that did so during a given time period. This singleton event (prevalence=1) is the most extreme possible anomaly, yet it was benign. In fact, on any given day, nearly half of all domains visited are benign singletons. On the other hand, one machine compromise incident involved the Sality malware, which injected itself into what appeared to be a normal program: `duo device health.exe`. The hash of the device health executable on disk remained identical to the hash observed on thousands of other hosts. However, the device health process, which normally connects only to a few well-known servers, connected to several malicious singleton domains: `suewyllie[.]com`, `724hizmetgrup[.]com`, and `pelcpawel.fm.interia[.]pl`, as well as more than 100 IPv4 addresses in singleton /24 prefixes. The

Table 7: Feature Importances for 12-Month AMP

Feature (Past/Current Event or Detection)	Importance
past_evt_Cloud IOC_Executed Malware	0.058180
past_compromise	0.051350
current_evt_Cloud IOC_Executed Malware	0.038131
current_evt_Threat Detected	0.030108
current_compromise	0.028917
past_evt_Threat Detected	0.022447
past_det_JS:Adware.Popunder.G	0.015835
past_det_W32.DFC.MalParent	0.015001
current_det_JS:Adware.Popunder.G	0.013212
current_det_Auto.7DF7E9D268.Adware	0.011176
current_det_W32.DFC.MalParent	0.010958
current_det_JS:Adware.Popunder.D	0.010012
past_evt_Retrospective Detection	0.009230
past_det_JS:Adware.Popunder.D	0.008447
past_det_PUA.Win.Dropper.Generic	0.008225
current_det_PUA.Win.Dropper.Generic	0.007561
current_evt_Retrospective Detection	0.006965
past_det_Auto.7DF7E9D268.Adware	0.006389
current_det_W32.File.MalParent	0.004822
current_det_W32.20509F5C91-95.SBX.TG	0.004217
past_det_W32.6165CE1CFD-95.SBX.TG	0.003764
past_det_W32.File.MalParent	0.003748
current_det_JS:Trojan.JS.Agent.UJY	0.003597
current_det_W32.B65FE764AC-95.SBX.TG	0.003187
past_det_W32.20509F5C91-95.SBX.TG	0.003145
current_det_W32.6165CE1CFD-95.SBX.TG	0.003041
past_det_W32.B65FE764AC-95.SBX.TG	0.002995
past_det_W32.Auto:3d62c45b09	0.002945
current_det_W32.Auto:e52a185db3	0.002911
past_det_W32.D5D86903BB-95.SBX.TG	0.002604

magnitude of singleton behavior was a strong supporting feature indicating a compromised host.

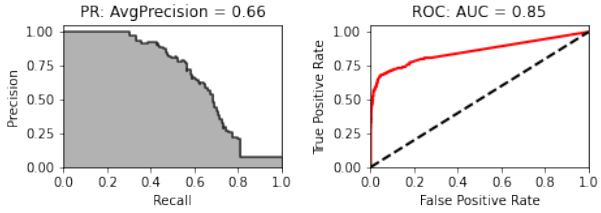
Perhaps surprisingly, despite orders of magnitude fewer training examples, the single-enterprise classifier for predicting next-week malware execution performs as well as or better than the classifier for the multi-enterprise dataset. Moreover, precision significantly improves when we add features from NVM, and when we train on a longer period of data with more example hosts. Table 8 shows example operating points for the classifiers, illustrating the progressive improvement. We discuss potential reasons for this in Sec. 7.

Table 8: Example Classifier Operating Points

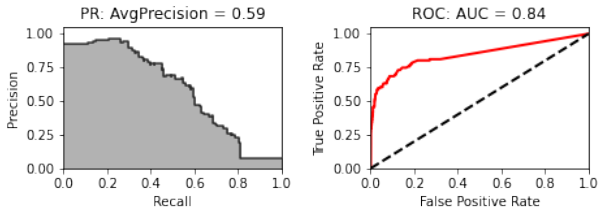
Experiment	Precision	Recall
6-month AMP	0.71 (5/7)	0.20 (5/25)
6-month AMP + NVM	0.83 (10/12)	0.40 (10/25)
12-month AMP	0.92 (54/59)	0.49 (54/111)

6.3 Classifier Degradation Over Time

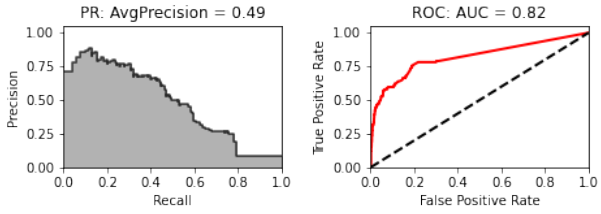
One question not addressed in the previous experiments is how quickly the classifier trained for predicting malware execution would degrade over time. Answering this could shed light on how quickly malware behavior changes. In the experiments above, the classifier was trained on data from weeks $1 \ldots n$ and applied to week $n+2$. (The 1 week gap avoids any possible overlap between training and test data.) In this section, we take the same classifier (trained on weeks $1 \ldots n$) and apply it to weeks $n+3$, $n+4$, and further. Specifically, we used exact the classifier from Fig. 5c, which was trained on 1 year of data, and applied it to test data starting with a 1 week gap and increasing up to a 19 week gap. Representative performance curves are shown in Fig. 6. Note that a 1 week gap is equivalent to Fig. 5c, so we omit it.



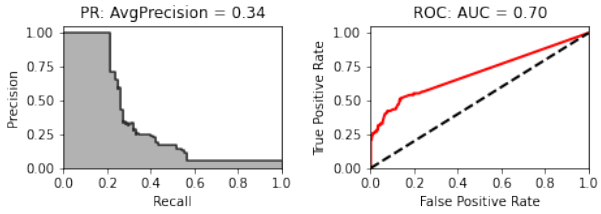
(a) 2023wk2 (2 week gap), begins 2023-01-09



(b) 2023wk3 (3 week gap), begins 2023-01-16



(c) 2023wk4 (4 week gap), begins 2023-01-23



(d) 2023wk14 (14 week gap), begins 2023-04-03

Figure 6: Classifier degradation over time. The random forest classifier decreases in performance as the gap between training and test data is increased from 1 week to 14 weeks.

The degradation in classifier performance can be tracked through the average precision numbers in Fig. 6 (roughly, the area under

the PR curve), which decrease from 0.68 (1 week) and 0.66 (2 weeks) to 0.34 (14 weeks). While there still remains a high precision region on the left side of the precision-recall graph, there is a noticeable decrease in performance. This suggests that some malicious activity is transient ("malware-of-the-week"), while other malware is perennial and recurs over long periods of time. The data in Fig. 6 therefore suggests that while the classifier retained some utility for months, the "best used by" period of the classifier appears to be 1-2 weeks after training.

7 DISCUSSION

We discuss the results of the prediction experiments in Sec. 6. While an exact comparison is impossible due to the time gap, it is remarkable that the prediction of next-week malware execution appears to have *benefited* from focusing on a single enterprise with orders of magnitude fewer training examples than the multi-enterprise dataset. We have several hypotheses for why this may be the case. The first is that there may exist local idiosyncrasies that do not generalize across enterprises. To revisit the firefighting analogy, suppose that in some neighborhood, the houses with metal roofs (usually beneficial) were built near a row of pine trees; in this neighborhood, the houses with metal roofs may be at *greater* risk, since dry pine needles in gutters can become kindling for fires. The single-enterprise case study is a unique environment with multiple layers of defense, including network blocking via managed DNS and IP null routes, which already eliminates connections to many low-reputation domains. This may mean that the detection of a JavaScript pop-under ad (browser pop-ups that do not block the active window but instead remain underneath until the main browser window is closed) is a slightly stronger indicator of compromise than it would be in the average enterprise, even if in both cases the detection itself does not warrant remediation.

A second factor may be that focusing on one enterprise enabled us to use more detailed features such as malware names, increasing the number of AMP features used by the random forest classifier from around 100 to tens of thousands, even if most appeared infrequently. A third factor in favor of the single-enterprise is the ability to effectively leverage a longer time span of training data. The multi-enterprise dataset did not appear to benefit from extending the training period beyond two weeks, and in some cases it worsened performance. In the single-enterprise case, the increase in training data from the 6-month to the 12-month experiment yielded substantial improvements (and using balanced random forests helped counteract over-fitting a small training set). We suspect that both the network environment and the malware landscape are more stable in a single enterprise than the multi-enterprise dataset. A final factor in favor of the single-enterprise case is the relative ease of joining data from multiple sources and vendors. While not a fundamental limitation of the multi-enterprise case (i.e., vendor-aggregated data), typical vendor product teams for antivirus and network monitoring software are scattered across corporate boundaries and are prevented by policy from sharing sensitive customer data. In contrast, practically every security team already joins multiple data sources from different vendors in its day-to-day operations. The 6-month AMP + NVM experiment showed that predicting imminent malware execution can benefit from adding a network perspective—in particular, by aggregating rare events by host, counting the number of new "singleton" domains, IP subnets, hashes, and processes, as well as the number of new public suffixes (for a given host). This suggests that well-crafted network features can be useful predictors.

There were operational lessons learned while working closely with an enterprise security team. As discussed in Sec. 6.1, when human analysts must review the outputs of a detector, high precision is critical, and ROC curves can be misleading. For example, Fig. 5a, Fig. 5b, and Fig. 5c all have nearly identical auROC ≈ 0.88, which is generally considered a good result. However, an analyst handed a list of results from the detector in Fig. 5a would waste considerable time on false positives. We therefore recommend focusing on high precision when human analysts are involved. A second lesson was that simple ML models are more suitable for several reasons. They tend to be more explainable, which is often a legal requirement—in unfortunate cases where an employee is fired based on evidence acquired from their company laptop, any ML-based detections must be translated into justifications acceptable in a court of law. Simple models also require minimal tuning as data and malware evolve over time. A security team for a large enterprise often deploys thousands of detectors in their "playbook" [5]; plays that require hand tuning quickly become irrelevant. Moreover, it is not yet clear that deep learning models are always superior on tabular, heterogenous data types; Grinsztajn et al. show that tree based models are still superior in many cases [11].

Finally, we discuss the implications of the feature importance rankings in both the multi-enterprise and single-enterprise case studies (Table 4 and Table 7), as they shed light on malware behaviors. The top feature was always a variant of "past history of malware execution," which reflects several common incident patterns. First, some hosts did not or could not undergo remediation when malware was initially detected (and automatic quarantine failed). Second are cases where root cause malware went undetected by AMP; that malware repeatedly launched a secondary payload that *was* recognized by AMP and quarantined. Third, some malware such as "droppers" are intrinsically designed to download other malware, and so are reliable indicators of further infection ("malware begets malware"). Not surprisingly, these three cases tend to correspond to infections that require human intervention. Moving down the list in Table 4 were two features that reflect human behavior: "Policy Update" and "Computer Metadata Changed." The former is consistent with an IT organization deploying a new AV policy to better detect threats, while the latter represents user behavior such as moving from a secure network (office) to an unsecure network (coffee shop). Lower in the feature importance table were vulnerable applications—vulnerabilities did correlate with compromises, just not as the primary, strongest predictors. In the single-enterprise study (Table 7), one of the notable indicators was JavaScript pop-unders. Pop-under ads are not themselves harmful, but as discussed above, in certain environments or in large numbers, they may be an indicator of other issues. The presence of malware sandbox (SBX.TG) detections in the top 30 features in Table 7 suggests that AMP was detecting early signs of malware that did not yet have a signature in the virus database. The particular malware families that require human intervention will inevitably change over time. However, our approach can detect many of those

patterns automatically and present them to the incident response team via the feature importance tables.

8 CONCLUSION

Defending an enterprise network involves many prioritization decisions: one is deciding which machines to remediate via human intervention. We show that malware execution risk can be an effective metric for prioritization. Perhaps surprisingly, it does not appear to be essential to collect training data across thousands of enterprises; predictive performance for a single enterprise can remain excellent even when training is restricted to the enterprise itself. Moreover, in the single-enterprise setting, it is relatively easy to combine different views of the hosts, such file-based and network-based telemetry. In our case, the richer features enabled nearly half of imminent (next-week) malware executions to be predicted with high precision.

ACKNOWLEDGMENTS

The authors would like to thank Cynthia Sturton for her help in the initial planning stages of this work. Joey Rosen and Adam Weller of Cisco CSIRT provided important feedback on the needs of incident responders and the quality of the data and results.

REFERENCES

[1] Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving Web Search Ranking by Incorporating User Behavior Information. In *29th ACM SIGIR Conference on Research and Development in Information Retrieval*. 19–26. https://doi.org/10.1145/1148170.1148177

[2] Luca Allodi and Fabio Massacci. 2014. Comparing Vulnerability Severity and Exploits Using Case-Control Studies. *ACM Transactions on Information and Systems Security* 17, 1, Article 1 (Aug 2014). https://doi.org/10.1145/2630069

[3] Stefan Axelsson. 2000. The Base-Rate Fallacy and the Difficulty of Intrusion Detection. *ACM Transactions on Information and System Security* 3, 3 (Aug 2000), 186–205. https://doi.org/10.1145/357830.357849

[4] Leyla Bilge, Yufei Han, and Matteo Dell'Amico. 2017. RiskTeller: Predicting the Risk of Cyber Incidents. In *2017 ACM Conference on Computer and Communications Security*. 1299–1311. https://doi.org/10.1145/3133956.3134022

[5] Jeff Bollinger, Brandon Enright, and Matthew Valites. 2015. *Crafting the InfoSec Playbook: Security Monitoring and Incident Response Master Plan*. O'Reilly Media, Inc.

[6] Center for Internet Security. 2023. CIS Controls Version 8. https://www.cisecurity.org/controls/

[7] Andrew Chi, Robert A. Cochran, Marie Nesfield, Michael K. Reiter, and Cynthia Sturton. 2017. A System to Verify Network Behavior of Known Cryptographic Clients. In *14th USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 177–195. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/chi

[8] 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 *ACM Internet Measurement Conference*. 36–49.

[9] FIRST. 2019. Common Vulnerability Scoring System v3.1: Specification Document. https://www.first.org/cvss/v3.1/specification-document

[10] Anthony H. Grieco and Adam J. O'Donnell. 2018. Information Technology Vulnerability Assessment. U.S. Patent No. 9928369-B2, Filed Feb. 9, 2015, Issued Mar. 27, 2018.

[11] Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data?. In *Advances in Neural Information Processing Systems*, Vol. 35. 507–520.

[12] Haibo He and Edwardo A Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* 21, 9 (2009), 1263–1284.

[13] International Organization for Standardization (ISO). 2022. Information security, cybersecurity and privacy protection—Information security controls. ISO/IEC 27002:2022. https://www.iso.org/standard/75652.html.

[14] Jay Jacobs, Sasha Romanosky, Idris Adjerid, and Wade Baker. 2020. Improving Vulnerability Remediation Through Better Exploit Prediction. *Journal of Cybersecurity* 6, 1 (Sep 2020). https://doi.org/10.1093/cybsec/tyaa015

[15] Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Idris Adjerid, and Michael Roytman. 2021. Exploit Prediction Scoring System (EPSS). *Digital Threats* 2, 3, Article 20 (Jul 2021), 17 pages. https://doi.org/10.1145/3436242

[16] Faris Bugra Kokulu, Ananta Soneji, Tiffany Bao, Yan Shoshitaishvili, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. 2019. Matched and Mismatched SOCs: A Qualitative Study on Security Operations Center Issues. In *2019 ACM Conference on Computer and Communications Security*.

[17] Chang Lan, Justine Sherry, Raluca Ada Popa, Sylvia Ratnasamy, and Zhi Liu. 2016. Embark: Securely Outsourcing Middleboxes to the Cloud. In *13th USENIX Symposium on Networked Systems Design and Implementation*. 255–273.

[18] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research* 18, 17 (2017), 1–5. http://jmlr.org/papers/v18/16-365.html

[19] 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 *24th USENIX Security Symposium*. 1009–1024.

[20] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. 2016. MLlib: Machine Learning in Apache Spark. *Journal of Machine Learning Research* 17, 1 (Jan. 2016), 1235–1241.

[21] Antonio Nappa, Richard Johnson, Leyla Bilge, Juan Caballero, and Tudor Dumitras. 2015. The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. In *2015 IEEE Symposium on Security and Privacy*. 692–708. https://doi.org/10.1109/SP.2015.48

[22] NIST. 2023. National Vulnerability Database. https://nvd.nist.gov/

[23] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[24] Miloslava Plachkinova and Chris Maurer. 2018. Teaching Case: Security Breach at Target. *Journal of Information Systems Education* 29, 1 (2018), 11.

[25] SANS. 2023. Internet Storm Center. https://isc.sans.edu/

[26] SANS. 2023. @RISK Newsletters Archive. https://www.sans.org/newsletters/at-risk/

[27] Armin Sarabi, Ziyun Zhu, Chaowei Xiao, Mingyan Liu, and Tudor Dumitraş. 2017. Patch Me If You Can: A Study on the Effects of Individual User Behavior on the End-Host Vulnerability State. In *International Conference on Passive and Active Network Measurement*. Springer, 113–125.

[28] Mahmood Sharif, Jumpei Urakawa, Nicolas Christin, Ayumu Kubota, and Akira Yamada. 2018. Predicting Impending Exposure to Malicious Content from User Behavior. In *2018 ACM Conference on Computer and Communications Security*. 1487–1501. https://doi.org/10.1145/3243734.3243779

[29] Yun Shen, Enrico Mariconti, Pierre-Antoine Vervier, and Gianluca Stringhini. 2018. Tiresias: Predicting Security Events Through Deep Learning. In *2018 ACM Conference on Computer and Communications Security*. 592–605. https://doi.org/10.1145/3243734.3243811

[30] Yun Shen and Gianluca Stringhini. 2019. ATTACK2VEC: Leveraging Temporal Word Embeddings to Understand the Evolution of Cyberattacks. In *28th USENIX Security Symposium*. 905–921.

[31] Kyle Soska and Nicolas Christin. 2014. Automatically Detecting Vulnerable Websites Before They Turn Malicious. In *23rd USENIX Security Symposium*. 625–640.

[32] Spamhaus. 2023. Abused Top Level Domains (TLD). https://www.spamhaus.org/statistics/tlds/

[33] Spamhaus. 2023. Composite Blocking List (CBL). https://www.abuseat.org/

[34] Spamhaus. 2023. Domain Block List (DBL). https://www.spamhaus.org/dbl/

[35] Sathya Chandran Sundaramurthy, John McHugh, Xinming Ou, Michael Wesch, Alexandru G. Bardas, and S. Raj Rajagopalan. 2016. Turning Contradictions into Innovations or: How We Learned to Stop Whining and Improve Security Operations. In *Twelfth Symposium on Usable Privacy and Security*. USENIX Association, 237–251.

[36] Sathya Chandran Sundaramurthy, John McHugh, Xinming Simon Ou, S Raj Rajagopalan, and Michael Wesch. 2014. An Anthropological Approach to Studying CSIRTs. *IEEE Security & Privacy* 12, 5 (2014), 52–60.

[37] Symantec. 2019. SecurityFocus Community. Retrieved 2019-10-24 from https://www.securityfocus.com/

[38] Cisco Talos. 2023. Threat Source Newsletter. https://blog.talosintelligence.com/

[39] Kami Vaniea and Yasmeen Rashidi. 2016. Tales of Software Updates: The Process of Updating Software. In *2016 CHI Conference on Human Factors in Computing Systems*. ACM, 3215–3226. https://doi.org/10.1145/2858036.2858303

[40] VERIS. 2023. The Veris Community Database (VCDB). https://verisframework.org/vcdb.html

[41] Daniel Woods, Ioannis Agrafiotis, Jason RC Nurse, and Sadie Creese. 2017. Mapping the coverage of security controls in cyber insurance proposal forms. *Journal of Internet Services and Applications* 8, 1 (2017), 8.

[42] 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 *27th USENIX Security Symposium*. 903–918.

[43] Ting-Fang Yen, Victor Heorhiadi, Alina Oprea, Michael K. Reiter, and Ari Juels. 2014. An Epidemiological Study of Malware Encounters in a Large Enterprise. In *2014 ACM Conference on Computer and Communications Security*. 1117–1130. https://doi.org/10.1145/2660267.2660330

[44] Xingliang Yuan, Xinyu Wang, Jianxiong Lin, and Cong Wang. 2016. Privacy-preserving Deep Packet Inspection in Outsourced Middleboxes. In *35th Annual IEEE International Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2016.7524526

A APPENDIX

A.1 Multi-Enterprise Dataset Survey: Correlating Malware and Vulnerabilities

This section provides a survey of the multi-enterprise AMP dataset through the lens of aggregate statistics on vulnerabilities and malware executions. While all business entities were anonymized (including business sector), this analysis gives a sense for the similarities and differences among those enterprises, from the vantage point of their AMP agents. An added benefit of this lens is that it enables us to test a classic claim: that running vulnerable software is a major reason for compromised hosts.

To survey the diversity of businesses in the multi-enterprise dataset through the lens of their vulnerabilities and compromises (i.e., malware executions), we began by relating the two factors on a per-business basis over a 12 month time period. In Fig. 7, each disc represents a business of at least 1000 hosts, with y-position indicating the fraction of hosts that executed malware that month, and x-position indicating the fraction of vulnerable hosts (i.e., that executed software with CVSS Base Score $\geq$ 6 at the time). The relative size of the businesses (by number of hosts) is monotonically related to the area of the disc. At such a coarse granularity, the link between vulnerability fraction and compromise fraction was not visually obvious, but Spearman's rank correlation coefficient indicated a weak positive relation of $\rho \approx 0.3$ that was statistically significant in all months (p-value < 1e-6). Moreover, when pooled across all businesses, vulnerable hosts were slightly more likely to be compromised than non-vulnerable hosts; the relative risk (a.k.a. risk ratio) of compromise was greater than 1 in all but a single month ($RR = 0.95$ in 2018-11). Fig. 7 also suggests that compromises tended to follow seasons where many businesses experience coordinated increases or decreases in attacker activity in the same month(s). Note that there were businesses with a large fraction of vulnerable hosts that suffered few compromises, and vice versa. This suggests, not surprisingly, that more factors affect host compromise than the number of unpatched vulnerabilities alone.

Ideally we would like to discover why compromises occurred, i.e., whether a particular CVE is correlated with host compromises. The AMP data sheds light on this, with a couple of important caveats. The first caveat is that this type of analysis can show correlation but not causation. The second caveat is that there exist equivalence classes of vulnerabilities that are statistically indistinguishable from the standpoint of the AMP compromise counts. This can happen, for example, if every employee is issued a standard laptop that includes MS Office affected by CVE-x and Adobe Acrobat Reader affected by CVE-y. More precisely, we say that vulnerabilities x and y "coincide" on a set of hosts H and time interval T when $\forall h \in H, t \in T$: x affects h at time $t \Leftrightarrow y$ affects h at time t. The "coincide" relation is an equivalence relation, and thereby forms equivalence classes of vulnerabilities; we name these "coincidence classes." Any statements we make based on statistical correlation of vulnerabilities to compromises will necessarily apply to coincidence classes as the fundamental unit. Moreover, two different coincidence classes of vulnerabilities may still be highly correlated if the sets of hosts they affect are close but not quite identical.

Caveats granted, we can compare the compromise rate of hosts that are affected by a given vulnerability against the baseline compromise rate of hosts in the AMP ecosystem. Fig. 8a shows an example for a Java vulnerability (CVE-2013-3829), which is a member of a coincidence class of 11 CVEs. The plot includes a 95% confidence interval (translucent band) for the compromise fractions of both the vulnerable and clean hosts. In the case of CVE-2013-3829, the ExecMalw rate does not differ significantly from the baseline rate. Note that we use a conservative definition of significance: we require non-overlap of the 95% confidence bands when comparing the compromise fractions of the vulnerable and non-vulnerable host populations.

In contrast, the Microsoft Office vulnerability (CVE-2015-1641) in Fig. 8b was reported by SecurityFocus [37] to have been exploited in the wild, and indeed the compromise rate for the hosts affected by this CVE is significantly greater than the baseline AMP compromise rate. (Note: the baseline is the same in all three plots, but the vertical scale is different.) Likewise, Fig. 8c shows an Excel vulnerability (CVE-2014-6360) with greater compromise rate than baseline, but at the time of writing, SecurityFocus had no confirmed reports of this vulnerability being exploited in the wild. Nevertheless, it may be worth examining these hosts for common patterns of malicious activity centered around Excel.

While the 3 CVE examples seem consistent with intuition, broadening the scope of analysis yields a surprising result, shown in Table 9. The contingency table summarizes the observed compromises related to 118 CVEs (coincidence classes), for which an average of at least 500 hosts/day were vulnerable to each CVE. Each CVE is counted in one cell of the table, depending on two factors: (1) public exploit status according to SecurityFocus (no reports yet, proof-of-concept published, or exploited in the wild), and (2) the duration of time in the AMP population for which the CVE-affected hosts exhibited significantly more compromise than baseline (never, minority of the time, or majority of the time). Interestingly, the compromise rates do *not* change significantly based on a CVE's public exploit status; Pearson's $\chi^2 = 1.75$ ($p = 0.78$) fails to reject the null hypothesis of independence.

This result could be due to the small fraction of exploits that are publicly reported, or to near-coincident CVEs dampening the observed differences in the rows. Nevertheless, this result suggests that the exploit status available in public databases may be a poor proxy for the actual vectors of compromise in a given set of enterprises. Instead, performing analysis of actual compromise rates such as that in Fig. 8 and Table 9 may be more effective for prioritizing patching.

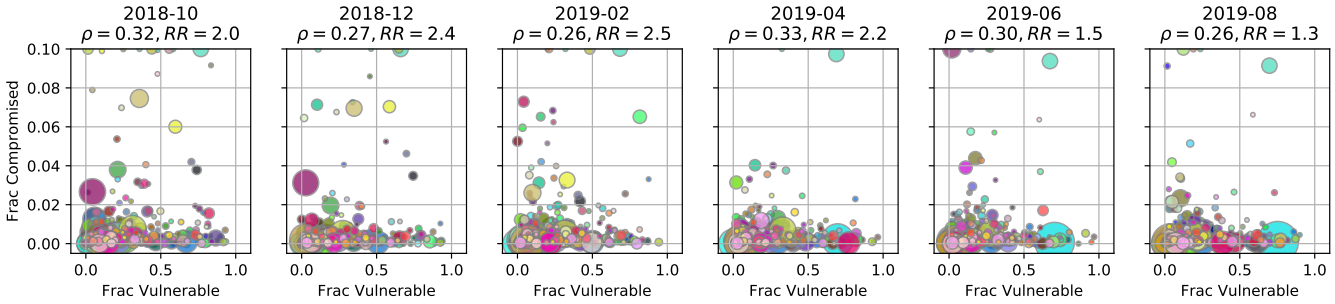


Figure 7: Per-business compromised host vs. vulnerable host fractions (every other month shown). Each business is a circle of random color with area monotonically related to number of active hosts. For all months, the Spearman rank correlation (ρ) between compromised and vulnerable fractions was weak but significant (p-value < 1e-6). The relative risk (RR) of compromise for vulnerable hosts as compared to non-vulnerable hosts, was computed across all hosts regardless of business.

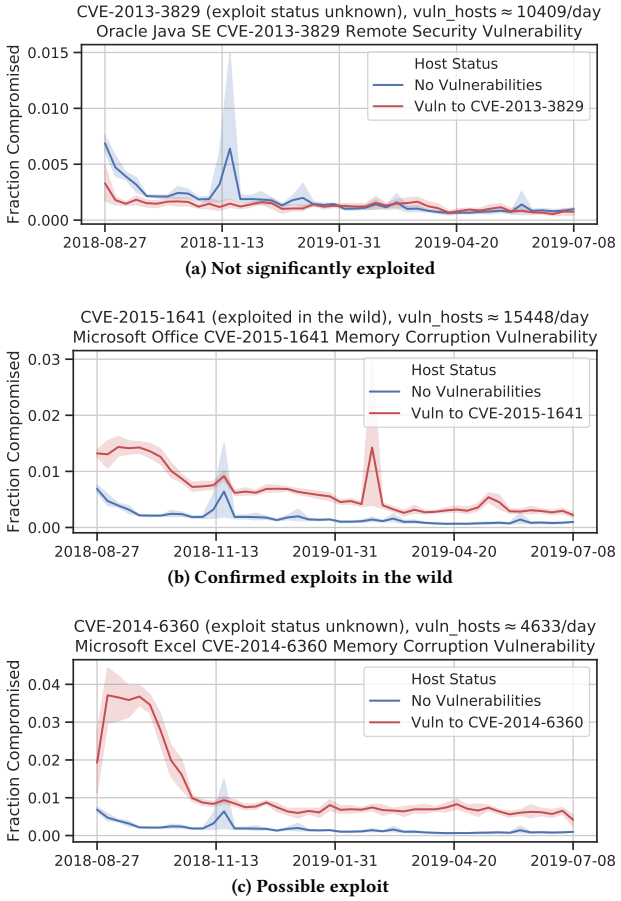


Figure 8: CVEs corresponding to various levels of host compromise over time, with 95% confidence bands. CVE-2015-1641 had public reports of exploits in the wild (SecurityFocus), but CVE-2014-6360 had no reports yet.

Table 9: CVE Counts by Public Exploit Status vs AMP Significant Compromise Duration ($\chi^2 = 1.75, p = 0.78$)

Public Exploit Status	AMP Duration of Signif. Compr.		
	Never	Minority	Majority
No Reports Yet	50	14	36
Proof-of-Concept	4	1	5
Expl. in the Wild	4	2	2

A.2 Supplemental Table: AMP Event Types

The following table (Table 10) provides a more comprehensive listing of AMP event types than the abbreviated sample table provided in the main body of the text (Table 1).

Table 10: Top AMP Event Types by Hosts and Events

Hosts (%)	Events (%)	Event Type	Description (if available)
81.593	4.423	Policy Update	An agent has been told to fetch policy.
66.886	10.011	Computer Metadata Changed	Some of the computer's metadata (e.g. IP address, OS) changed.
44.441	4.305	Scan Started	An agent has started scanning.
41.938	4.253	Scan Completed No Detections	A scan has completed without detecting anything malicious.
33.484	3.628	Product Update Started	A product update has begun.
30.814	0.289	Product Update Completed	A product update has successfully completed.
30.349	0.983	Vulnerable Application Detected	Host has executed a program with known vulnerabilities.
28.322	0.166	Reboot Pending	An agent has started the reboot process.
28.259	0.170	Reboot Completed	An agent has completed its reboot.
24.666	2.064	Low Prevalence Execution	A file with low prevalence was executed.
22.944	0.093	Update Reboot Required	The new connector is installed but not running.
18.220	0.464	Policy Update Failure	A policy update failed, and the policy was not successfully applied.
14.047	0.072	Update Reboot Advised	New connector installed and running; some features need reboot.
9.985	2.300	File Detection	A file-based threat was found on this system (detection).
8.684	3.260	Product Update Failed	A product update has failed.
8.393	1.451	Threat Quarantined	A threat was successfully quarantined.
8.311	24.960	Attempting Quarantine Delete	Attempting to remove an item from quarantine.
6.214	1.729	Quarantined Item Deleted	A quarantined item has been successfully deleted.
5.991	0.206	Retro File Fetch Completed	The request for a remote file was successful.
5.826	24.165	Failed to Delete from Quarantine	A quarantined item was not successfully removed from quarantine.
5.803	0.024	Uninstall	
4.475	0.497	Quarantine Failure	A detected threat was not successfully quarantined.
4.271	0.097	Retro File Fetch Failed	The request for a remote file failed (retrospective).
4.063	0.152	Generic IOC	Suspicious behavior indicates possible compromise (e.g., OpenIOC).
3.863	7.503	System Process Protected	Prevented memory injection attack on Windows system process.
2.721	1.068	Threat Detected in Exclusion	A threat was detected in an excluded path, or otherwise whitelisted.
2.616	0.086	Retro Quarantine Attempt Failed	A retrospective quarantine was attempted and failed. Most likely the original location no longer exists.
2.316	0.105	Retro Quarantine of False Negative	A file (once thought clean) was marked malicious and quarantined.
2.165	0.028	**Executed Malware**	The computer executed known malware.
2.159	0.113	Exploit Prevention Full Summary	Prevented memory injection attack on an application - full summary.
2.159	0.113	Exploit Prevention	Prevented memory injection attack on an application - summary.
1.960	0.016	Scan Completed With Detections	A scan has completed and detected malicious items.
1.647	0.018	Retro Restore of False Positive	A file (once thought malicious) was marked as clean and restored.
1.364	0.007	IOC Configuration Update Success	Endpoint IOC Configuration Update Success.
1.282	0.014	Scan Failed	A scan has been attempted, and failed to run.
1.165	0.006	IOC Definition Update Success	Endpoint IOC Definition Update Success.
1.128	0.013	Retro Restore from Quarantine	A retrospective restore was completed successfully.
1.082	0.014	Retro Quarantine Success	A retrospective quarantine was completed successfully.
0.789	0.007	Fault Cleared	
0.762	0.283	Execution Blocked	Execution of an application was blocked.
0.759	0.004	Retro Restore from Quarantine Failed	A retrospective restore was attemped and failed. Most likely the original location no longer exists.
0.573	0.006	Repeated Malware Download	Same parent process repeatedly attempted to introduce the same piece of malware.
0.567	0.025	Critical Fault	
0.565	0.019	Major Fault	
0.503	0.735	Network Conviction	Connection detected by Network Flow Monitor (NFM), aka DFC.
...	...	...	...