CAMP: Content-Agnostic Malware Protection

Moheeb Abu Rajab, Lucas Ballard, Noé Lutz, Panayiotis Mavrommatis, Niels Provos
Google Inc.
1600 Amphitheatre Parkway, Mountain View, CA 94043

***Abstract*---In spite of recent advances, the world wide web remains an important vector for malware installation. Approaches to evaluating potentially malicious code before execution in a browser, such as blacklisting or content-based detection are hindered by an attacker's ability to easily change hosting domains or mutate malware binaries. On the other hand, whitelist-based approaches are challenged by the large, dynamic, and heterogeneous space of benign binaries that they must track. In practice these approaches continue to provide value for popular binaries at either extreme of maliciousness (e.g., the current large outbreak of malware, the benign binaries shipped with an OS), but bridging the gap between whitelist and blacklist detection for web malware remains a significant challenge.**

This paper presents CAMP, a content-agnostic malware protection system based on binary reputation that is designed to address these shortcomings. CAMP is built into the browser and determines the reputation of most downloads locally, relying on server-side reputation data only when a local decision cannot be made. This paper gives a detailed overview of CAMP and its architecture and provides an evaluation of the system through a six-month deployment in which 200 million users of Google Chrome requested between eight to ten million reputation requests a day. Our evaluation shows that CAMP exhibits accuracy close to 99% relative to proprietary VM-based dynamic analysis, is able to process requests in less than 130 ms on average, and was able to detect approximately five million intentional malware downloads per month that were not detected by existing solutions.

I. Introduction

Malware is a popular venue for monetization in the underground economy [7]. Adversaries frequently compromise vulnerabilities in users' machines to automatically install malware via so-called drive-by downloads. The action of visiting a malicious web page in a vulnerable browser is sufficient for an adversary to gain complete control over the vulnerable machine allowing her to install arbitrary software. Frequently, installed malware can cause the computer to become part of a botnet or can exfiltrate sensitive data such as credit card numbers to be sold in the underground economy [20].

As browsers and plugins are becoming more secure, the vulnerable user base is shrinking and drive-by downloads are becoming less effective as a malware distribution vector. While drive-by downloads are still a popular distribution vector, adversaries are increasingly turning to social engineering for malware distribution. Fake Anti-Virus products are one such example in which the user is led to believe that their computer has been infected and a free product to remedy the situation is offered. Consequently, the user voluntarily downloads and installs malware under the guise of a free security solution. The benefit to the adversary is that the malware is now in control of the computer and no vulnerabilities or exploits were required for the installation [28], [12].

Traditional defense mechanisms such as Anti-Virus (AV) products are often ineffective against current malware downloads as AV engines tend to be content based and provide an oracle to malware authors allowing them to repackage their software until it is no longer detected [24]. Similarly, URL blacklists such as Google's Safe Browsing API [16] work well for identifying compromised web pages which tend to be relatively static, but cannot be completely up to date in identifying all the domains from which malware is being distributed. The abundance of dynamic DNS providers allow for frequent rotation of domains from which malware is being distributed. If the domains rotate faster than the update interval of the malware lists, domains distributing malware are more likely to remain undetected [3], [27].

The malware arms race has shown that neither signature based AV engines nor malware lists are sufficient to protect users. A whitelist based approach in which only trusted software can be installed is more promising in that it can completely block the installation of socially engineered malware [6], [11]. Since such malware is not on the whitelist, installation of it will not be allowed. However, whitelists suffer from drawbacks, too. The sheer variety of available software and the perpetual creation of new software makes the task of creating a whitelist that covers all benign software nearly impossible. As a result, neither whitelist nor blacklist based approaches provide sufficient coverage to protect users from malicious software in practice.

This paper offers a different approach in which the gap between known benign and known malicious software is bridged by a content-agnostic reputation-based detection approach. CAMP, our detection system, consists of a client component built into Google Chrome and a server component responsible for maintaining a reputation system that predicts the likelihood that a downloaded binary is malicious. CAMP makes use of Google's Safe Browsing API to detect downloads known to be malicious. For all binaries not detected, we extract additional features from information available to the web browser about the download and use these features to build a server-based reputation system. The reputation system predicts if an unknown binary is malicious without prior knowledge of its content. We built CAMP to protect users of Google Chrome from downloading malware on Windows. Our system protects over 200 million users, makes millions of reputation-based decisions every day, and identifies about 5 million malware downloads every month beyond the malware warnings that

Google's Safe Browsing API shows for dangerous web sites. CAMP achieves an accuracy close to 99% relative to proprietary, VM-based dynamic analysis.

In this paper, we provide a detailed overview of the design and architecture that comprises our reputation-based detection system and evaluate its effectiveness based on data collected from February to July 2012. We discuss how the system minimizes the impact on the privacy of browser users and evaluate its detection performance focusing especially on false positives.

This paper makes the following contributions:

- We present CAMP, a content-agnostic malware protection system, which utilizes reputation-based detection to protect users from web-based malware.
- We perform an extensive, six month evaluation of CAMP consisting of over 200 million unique users and millions of daily reputation evaluations. We show that our content-agnostic detection approach is both accurate, with an accuracy close to 99%, and well performing, processing requests in less than 130 ms on average.
- We compare CAMP with the current state of practice and demonstrate that CAMP outperforms Anti-Virus, as well as various web services, e.g. McAfee's Site Advisor, Symantec's SafeWeb, etc. Further, CAMP identifies large numbers of malware undetected by these services, including 5 million malware downloads per month not identified by Google's Safe Browsing API.

The remainder of the paper is organized as follows. We discuss related work in Section II. In Section III, we present a detailed description of the system architecture and all components responsible for creating the reputation system. Section IV discusses how to leverage the reputation system for malware detection. We evaluate the performance of the system in Section V, and present a small case study of a highly dynamic malware campaign. We discuss CAMP's unique properties in Section VI. Finally, we conclude in Section VII.

II. Related Work

Protecting users from malware continues to be a challenging problem. In the following, we provide an overview of different approaches to prevent malware from infecting users and explain how CAMP differs from prior work.

a) Content-based detection: Anti-Virus software is one of the most common content-based malware detection techniques. Usually, Anti-Virus products run on end-user systems and employ signature-based detection to identify variants of known malware.

While Anti-Virus engines do help in protecting users from malware infections, several challenges limit their effectiveness. Malware authors have developed increasingly sophisticated evasion techniques, such as packing and polymorphism, aimed at circumventing detection by AV engines [4], [30]. Additionally, the signature generation and update cycle causes an inherent delay in protecting users against new variants of malware.

Oberheide et al. [24] proposed CloudAV as a new model to increase the effectiveness of Anti-Virus products. Under this model, Anti-Virus protection is moved into the cloud and multiple AV engines, working in parallel, are employed to improve detection rates. While combining the decision of multiple AV engines leads to better detection, this approach is still subject to the limitations mentioned above as CloudAV also relies on timely signature updates. Furthermore, Oberheide et al. show that major Anti-Virus engines detect only 35% to 70% of recent malware which means that many malware binaries remain undetected. CloudAV also requires that all binaries are uploaded to the cloud which exposes these downloads to a third-party and may constitute an unacceptable loss in privacy for some users. While CAMP also moves detection of malware into the cloud, it reduces the privacy impact by employing whitelists so that most download URLs stay within the browser and do not need to be sent to a third party. Binary payloads never leave the browser. Additionally, its content-agnostic approach does not suffer from the same limitations as AV engines, e.g. delays in signature updates.

b) Blacklist-based protection: Blacklist-based approaches provide protection from malware by identifying the sites from which it is being served. Services such as Google's Safe Browsing API [16], McAfee's Site Advisor [22] or Symantec's Safe Web [23] detect malicious or compromised web sites that may infect users with malware. Browsers integrate with these services directly or via tool bars and prevent users from visiting known malicious sites. While effective in identifying compromised web-sites which tend to be long lived, blacklists face challenges when trying to protect against adversaries that employ highly agile malware distribution servers [27]. By frequently changing the domains that serve malware, blacklists become less effective as domains rotate faster than the time it takes to detect them.

CAMP offers a different approach. Instead of exporting a blacklist to clients, CAMP protects users by augmenting blacklists and whitelists with a content-agnostic reputation system that protects users from malicious binaries without requiring a-priori knowledge of the binary or its serving domains. Specifically, the browser performs the following to check the safety of a downloaded file: (1) the browser tries to locally determine whether the download is malicious by checking the download URL against a list of URLs known to serve malware exported to the browser using Google's SafeBrowsing API, (2) the browser checks locally against a dynamically updated list of trusted domains and trusted binary signers to determine whether the downloaded content is likely benign and, (3) for downloads that do not match any of the local lists, the browser extracts content-agnostic features from the download and sends a request to CAMP's reputation service. As a result, CAMP protects users against malicious downloads that would likely have been missed by a blacklist-based approach. We compare CAMP's performance to popular blacklist services in Section V.

CAMP and Google's SafeBrowsing API complement one another. The API exports blacklists of compromised sites that

include content from malicious sites that automatically exploit a browser with no interaction from the user. This works well for protecting browsers from infected landing pages as they are relatively static, and compromise can be observed by an automated system to build the blacklist [25]. CAMP, on the other hand, targets binaries that were downloaded by misled users from highly dynamic infrastructures.

c) Whitelist-based schemes: In contrast to blacklists, whitelists ensure that only known benign software can be installed and installation of everything else is disallowed. For example, Bit9 [6] and CoreTrace [11] maintain a list of binaries verified to be benign. While whitelisting can be effective in enterprise settings, it remains challenging to maintain an up-to-date whitelist that covers the plethora of applications developed globally. Therefore, protecting downloads in the browser through whitelists alone is not currently practical. Nonetheless, to protect user privacy, CAMP derives a whitelist from its reputation data that is used by the web browser to locally decide if a binary download should be trusted. Downloads not on the whitelist require a reputation-based decision.

d) Reputation-based detection: A lot of research has been conducted on reputation systems [19] and how to use them for detecting malicious activities. Hao et al. [18] proposed SNARE, a reputation system for detecting spam email. Qian et al. [26] proposed using network-based clustering to increase the accuracy of spam-oriented blacklists.

Notos [2] and EXPOSURE [5] offer a dynamic reputation engine for DNS. Both systems use features based on passive DNS resolution to predict the likelihood that a domain name is malicious. CAMP is complimentary to Notos and EXPOSURE but is specifically targeted to predict whether an unknown binary download is malicious. Furthermore, the use of DNS limits a reputation system to making decisions on the granularity of domain names whereas CAMP operates on all of the features that are associated with a downloaded binary.

Finally, closely related to our work, is Microsoft's SmartScreen described briefly in a blog post [10]. SmartScreen utilizes a reputation-based approach to protect users from socially-engineered malware. It does so by computing reputation for the download URL, the file identifier and the publisher if available as a digital signature. Unlike CAMP, it requires that all download URLs are sent to a remote server. In contrast to SmartScreen, CAMP computes its reputation based on many other features available to the browser, such as referrer URLs, IP addresses, etc. In addition, this paper provides in-depth technical details of CAMP's reputation system and evaluates its performance in an operational setting.

III. System Architecture

In the following, we provide an overview of CAMP's design and architecture. Our goal is to create a system that scales to hundreds of millions of users and protects them from downloading malware while at the same time limiting the impact on their privacy. Our focus is specifically on binaries downloaded by users rather than drive-by downloads in which malware is installed in the background without knowledge by the user. We believe that focusing on user downloads is justified for several reasons: the security of browsers has increased significantly over the last few years [17], [29], [32] and proposed exploit detection systems such as Blade [21] or Zozzle [13] offer efficient mechanisms to prevent current exploits from succeeding. As a result, automated exploitation of browsers has become more difficult and adversaries are also incorporating social engineering techniques in which users download malware themselves.

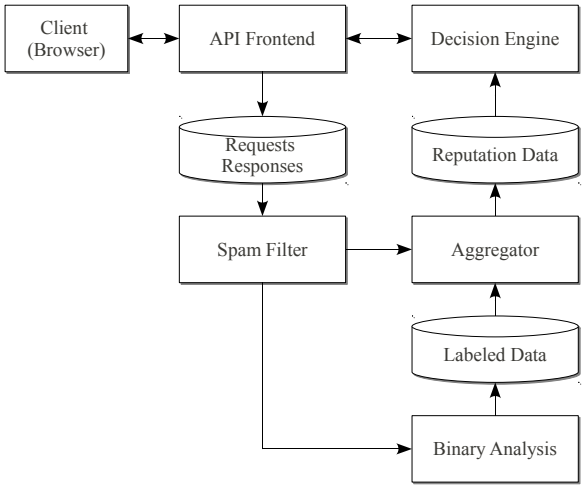

Fig. 1. The diagram presents a high-level overview of the detection system showing the communication between the client and server as well as the server-side processing infrastructure.

To efficiently determine if an unknown binary download is malicious, CAMP is split into a client component and a server component. The client component is integrated into a web browser. As a first step, the client checks the download against both a local blacklist and whitelist. If no match is found, the client extracts features from the binary download, sends these features to the server, and displays a warning to users if the server response instructs it to. The server processes the features sent from the browser and computes a reputation decision informed by the client request and a reputation metric constructed from previous client requests. Figure 1 shows an overview of the complete detection system. We will discuss it in detail in the following sections.

A. Binary Analysis

Many machine learning algorithms require labeled ground truth for training purposes. Our situation is similar in that we need to label our reputation data according to the nature of the binary, e.g. whether it is malicious or benign. We support multiple dimensions for the labels so other classifications such as *spyware* are possible, too. Similarly, the type of binary is its own dimension, e.g. Windows PEbins are treated differently from Mac OS X DMG files. However, for any binary type, a corresponding classification system needs to be available to provide accurate labels.

In this paper, we make use of a binary analysis system that we developed independently to classify binaries based on static features as well as dynamic execution traces. The main goal of this system is to limit the number of false positives and we consciously sacrifice detection in favor of fewer false positives.

The labels produced by the binary analysis system form the ground truth that governs the training of the reputation system. The labels also allow us to compute detection performance of the reputation system post facto. We measure this as part of our evaluation in Section V.

It is important to note that our reputation system does not require a particular binary classification solution and other detection approaches [9], [33] or labeling solely based on the output of AV engines should also be possible. Of course, the overall accuracy of the system would be dependent on the accuracy of the underlying labeling mechanism.

Binary analysis systems are not perfect and are susceptible to evasion [15], [31]. Since CAMP is a reputation system that requires a reasonable estimate of ground truth, any errors in labeling might propagate to CAMP's decisions. However, since CAMP is independent of the classification mechanism, any advances to dynamic analysis or Anti-Virus products will seamlessly improve CAMP's performance. Nonetheless, we show in Section V-B that our binary analysis system exhibits reasonable accuracy.

B. Client

When a user initiates a binary download in her web browser, the web browser applies several checks before asking the server for a reputation decision. If any of these checks fail, the client makes a local decision on whether the download is malicious or not.

1) The web browser determines if the binary is already known to be malicious based on its URL, for example, via a list of known malware sites. In our case, we use Google's Safe Browsing API to make that determination. If the binary is known to be malicious, the browser can display a warning directly without requiring a reputation decision.
2) The browser determines if the binary download could potentially be harmful to the computer, e.g. it might correspond to an executable which may carry malicious code, or a DMG which is how Mac OS X software is installed.
3) The binary download is checked against a dynamically updated whitelist of trusted domains and trusted binary signers. The list of trusted domains or trusted signing certificates consists of reputable software publishers known for not distributing malware. If the binary download matches the whitelist, the browser assumes that the download is benign and no server-side reputation decision is required.

If all the above checks do not result in a local decision, the browser extracts features from the downloaded binary. Since the reputation decision is made on the server, the client can provide as many features as are available to it. The following is a list of features usually available to web browsers:

- The final download URL and IP address corresponding to the server hosting the download.
- Any referrer URL and corresponding IP address encountered when initiating the download, e.g. the results of multiple HTTP redirects.
- The size of the download as well as content hashes, e.g. SHA-256.
- The signature attached to the download which includes the signer as well any certificate chain leading to it. The client also informs the server if it could successfully verify the signature and if it trusted the signature, e.g. it was rooted in a trusted certificate authority.

The client then sends these features to the server and awaits its response. The server may reply with several different verdicts based on the reputation data available to it. It can inform the web browser that the download is predicted to be *benign* in which case the web browser completes the download without displaying a warning, or it can tell the web browser that the download is either deemed *malicious* (Figure 2) or *unknown* (Figure 3). In both of the latter cases, the web browser warns the user about the download and offers to delete the downloaded file. The *unknown* verdict indicates that the server did not have sufficient reputation information to label the download as either *benign* or *malicious*. Our evaluation shows that in the majority of all cases *unknown* by itself is good a predictor for malicious downloads. The different reputation verdicts are explained in more detail in Section IV.

User privacy is an important goal for CAMP. Verifying the content type of the file and that it neither matches blacklists nor whitelists drastically limits the number of downloads for which a remote server is contacted. As shown in Section III-D, the web browser contacts the CAMP service for only about 30% of binary downloads. Furthermore, the web browser sends only features computed from the binary, not the binary itself, to the server.

C. Server

The server pipeline has two different roles when processing requests. First, the server receives the client request and renders a reputation verdict based on its reputation system which encompasses aggregate information from all downloads observed by the server during its measurement intervals, including e.g. 1 day, 7 day and 90 day intervals. Second, the server uses the information provided by the client to update its reputation data.

The reputation verdict is computed by a reputation metric calculated by a binary circuit that has access to all features from the client request and any reputation data that is referenced by those features, e.g. how many known benign or malicious binaries are hosted on a given IP address, etc.

To incorporate information provided by the clients into the reputation system, client requests are first despammed to prevent misbehaving clients from unduly influencing the data.

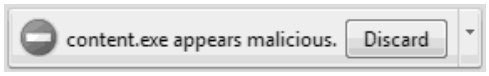

Fig. 2. The browser warns the user that the download is malicious. The intentionally discrete arrow presents an option to keep the file.

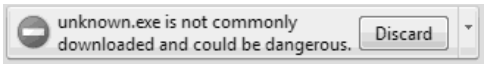

Fig. 3. The browser warns the user that the download is not commonly downloaded.

Despammed requests are then processed within a few minutes to generate up-to-date features.

To create a classification system that has both high performance and high reliability, we employ BigTable [8] and MapReduce [14] for distributed data storage and parallel processing. In the following, we provide a detailed overview of each component involved in making reputation decisions.

1) Reputation System: The heart of the decision process is the reputation system. To better understand its properties, we place it within the analysis framework proposed by Hoffman et al. [19]. According to Hoffman a reputation system can be characterized across the following three dimensions:

- Formulation, which represents the mathematical underpinnings of the reputation metric as well as its information sources.
- Calculation, which is the concrete implementation of the formulation.
- Dissemination, which characterizes how the results from the reputation metric are propagated to participants.

In our case, both the calculation and dissemination are centralized and deterministic. The storage of reputation data is transient as each item expires after 90 of days.

In the following, we explain the formulation of the reputation system in more details.

Our reputation data is based solely on direct, automatic sources. The output of the binary analysis pipeline is a trusted automatic source. Data collected and sent by web browsers is also a direct, automatic source but may be untrusted. The reputation data consists of features across a variety of dimensions that each describe some aspect of the binary or its hosting infrastructure. As mentioned in Section III-B, for each binary download, we receive not only a content-hash but also related information such as corresponding URLs and IP addresses of the servers hosting them, etc. The server may derive further features from the client request.

The reputation data maps each feature to an *aggregate* that contains measurements over data observed during a given time frame. Aggregates are continuous features and consist of two counts: the number of interesting observations and the total number of observations. For example, assume CAMP observed 10 downloads, 6 of which were malicious, on IP address IP_A. The aggregate corresponding to the feature IP:IP_A would then be $\{6, 10\}$. Each aggregate also contains the first and last time the particular feature was seen.

The aggregates include both positive as well as negative events, i.e. they can be both trust building and trust diminishing. For example, the number of users downloading a binary from a site may represent an increase in trust. On the other hand, the number of malicious binaries hosted on a site may diminish its trust.

As CAMP is deployed to a large number of users, many of the design decisions in building the system are in favor of reducing false positives. However, the performance of the reputation system can be adjusted gradually to favor recall over precision.

We provide a detailed discussion of CAMP's reputation-based detection in the next section but give an overview here of the reputation system itself. The reputation system is responsible for receiving a browser reputation request and replying to it with a verdict.

For each client request, the reputation system can make a decision based on a-priori information if either the URL or the content hash is known to be malicious. Similarly, to respond to major false positives, the reputation system consults a server-side whitelist to override any reputation decision.

The reputation metric is calculated by a binary circuit that references reputation data in form of previously computed aggregates. The features from the client request and the reputation formulation determine which aggregates are looked up from the data store. The reputation system then computes a verdict which can be either: benign, malicious or unknown; see Section IV for a discussion of the different meanings. The data store lookups happen in parallel and are non-blocking to reduce overall latency. The time spent computing the decision from the aggregates is insignificant compared to the time it takes to look up data from the data store.

2) Frontend and Data Storage: The frontend is responsible for receiving requests from web browsers and answering them without incurring any significant latency. To achieve low latency, we split the reputation metric computation and the integration of new data into the reputation system into separate components. Upon receiving a request, the frontend issues a Remote Procedure Call (RPC) to the reputation system, which determines whether the binary download is malicious. After receiving an answer, the frontend writes the request and the verdict to a data store that other components of the pipeline can process, and then returns the verdict to the client.

As CAMP needs to handle a large number of web browser requests, the temporary storage of request data requires a carefully chosen storage layout. We use BigTable [8], a non-relational distributed database that provides key-value stores and allows the association of a timestamp with a given key. While Bigtable scales well, it is limited to approximately 2GB of data per key. For subsequent data processing it is helpful to index requests by the URL of the binary. However, as we store each request for two weeks and some URLs are requested frequently, on the order of hundreds of thousands times a day, we chose not to index solely by URL. Instead, we append the Reverse-Ordered hexadecimal string representation

of the timestamp of the request to the URL. This causes the data to be placed in different rows while maintaining identical ordering compared to indexing by URL. This design decision was crucial in scaling CAMP to handle popular URLs.

3) Spam Filtering: The spam filter processes the data written by the frontends in real time and discards requests that do not originate from regular users. We do not require the spam filter to be highly accurate and false positives are acceptable. The spam filter may make use of any information provided in the client request and has visibility into all client requests made within the last 24 hours. As a result, the spam filter can apply velocity controls on the user IP address of the request, the Autonomous System Number (ASN) corresponding to the IP address, etc. The spam filter also ensures that requests are properly formed and contain all required features, e.g. properly formatted URLs, etc.

Requests not discarded by the spam filter are forwarded to an aggregator that computes aggregate features used by the reputation system. The output of the spam filter is also employed to fetch binaries from the web that have not been analyzed by the binary classifier. Since binary downloads may carry sensitive information, we apply further filters so that only binaries that exhibit sufficient diversity of context are considered for analysis. The binary analysis does not gate any reputation decision and may complete a long time after a reputation decision was sent back to the web browser.

As stated previously, our goal is not only to make highly accurate decisions but also to reduce the impact on the privacy of web users as much as possible. The requests received from web browsers reveal not only the binary URL visited by a user but by their very nature also a corresponding source IP address. The data processing architecture of the reputation system is designed so that the source IP address is visible only to the spam filter and completely deleted after two weeks. The binary URL is visible to rest of the pipeline, but is stored in such a way that it also is automatically deleted after two weeks. The only information that is stored for up to 90 days are the aggregates that make up the reputation data.

4) Aggregator: The basis of the reputation system is formed by the reputation data which consists of statistical aggregates indexed by keys derived from request features. Aggregates are computed and written by the aggregator, which processes the output of the despammer and organizes aggregates according to a three-dimensional index.

- The first index dimension is defined by whether the aggregate is computed from client requests or based on aggregates from the binary analysis system. Aggregates computed from client requests are considered untrusted whereas aggregates from the binary analysis system are inherently trusted.
- The second index dimension consists of features from client requests and additional features derived from the request on the server side.
- The third index dimension contains broad categories over which aggregates are computed. For client side aggregates, this contains the number of requests that received a malicious reputation decision as well as the number of requests for binaries known a priori to be malicious, either based on their URL or corresponding content hash. For the aggregates from the binary analysis system, this contains the number of URLs hosting malicious downloads as well as the number of malicious content hashes.

For example, the aggregate for

```
client|site:foo.com|reputation
```

represents the total number of client requests for the site *foo.com* as well as the number of client requests for the same site that received a malicious reputation decision. Another example is

```
analysis|site:foo.com|urls
```

which contains the total number of URLs found under *foo.com* as well as the number of such URLs that were labeled malicious by the binary analysis system.

To construct these aggregates, the despamming component writes client requests to a temporary data store which is indexed by the second aggregator index dimension. Then a series of MapReduces [14] periodically processes all entries in the data store. This process merges new data with older aggregates to generate aggregates over different time intervals, currently 1 day, 7 days, 14 days, 28 days and 98 days. All aggregates computed this way are available to the reputation system.

D. Client-side Whitelist

CAMP reduces the privacy impact on its users in several different ways. Since the detection is content-agnostic, web browsers do not need to send binary payloads to CAMP's servers. As users download binaries that potentially contain sensitive information, any solution that requires transmission of payload data is likely to face privacy challenges.

Another way in which CAMP reduces the privacy impact is by client-side whitelists that determine which binaries are trusted in advance. If a binary is trusted, CAMP does not need to send the binary URL to a remote server. The whitelists contain trusted domains such as *microsoft.com* as well as trusted software publishers. A software publisher is identified by the public key of the signing certificate and the CA certifying it. The goal of the whitelists is to resolve the majority of all downloads locally without requiring a reputation-based decision. At the time of this writing, approximately 70% of all downloads are considered benign due to policy or matching client-side whitelists.

For a domain or signer to be added to the whitelist, it needs to fulfill the following criterion. CAMP needs to have seen the binaries from the domain or signer for at least 90 days without encountering any signs of maliciousness. We add new domains or signers to the whitelist in the order in which they contribute to the number of requests received by CAMP over that time period.

Since adding a domain or signer to the whitelist implies that the CAMP server no longer receives any downloads for it, we employ a web crawl to regularly analyze binaries hosted

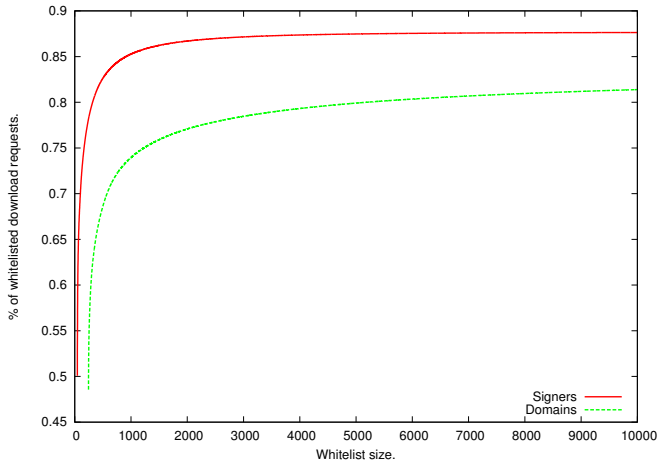

Fig. 4. The graph shows the percentage of downloads for which a web browser would make a local reputation decision depending on the size of the domain and signer whitelist.

on the trusted domains or signed by trusted signers. If the binary analysis reveals evidence of malicious behavior, the corresponding entries are removed from the whitelist and are again subject to reputation-based detection.

Figure 4 shows the number of downloads that a web browser would handle locally when adding more trusted sites to the domain or the signer whitelist. We estimated these values by considering all reputation requests that we received, then iteratively adding the most popular trusted signers and domains to our whitelist and tracking which requests would have been handled locally by the browser. The graph does not contain measurements for whitelists smaller than our initial whitelists as CAMP does not have any knowledge about requests corresponding to them. Our initial domain whitelist contained 243 entries and matched approximately 47% of all download requests remaining after policy checks. The graph shows that by increasing the domain whitelist to 1000 entries the local percentage of downloads not requiring a request to CAMP would increase to 73%. The signer whitelist is more effective as increasing it to 1000 entries would cover about 85% of downloads locally.

IV. Reputation System

The goal of our reputation system is to determine a-priori if a binary is likely going to be malicious or not without having access to the content of the binary itself. Instead, metadata about the binary such, as its hosting infrastructure and how the user reached it, form the basis for detection.

One of the challenges with detecting malware is the ease with which malware authors can change their binaries, e.g. by repacking, and how quickly they can change the locations from which binaries are being served. On the other hand, there is no need to frequently change the hosting location of benign binaries or the binaries themselves. Differences such as these may be leveraged to create a reputation metric that can determine with a high degree of confidence whether a binary is benign or not.

Binaries hosted on the web fall within a wide spectrum ranging from known-benign to known-malicious. Known-benign binaries can be added to a whitelist and known-malicious binaries to a blacklist. In an ideal situation, any binary is either known to be good or known to be bad. The reputation system described here is concerned with the gray area between whitelists and blacklists.

For CAMP, we attempt to create a whitelist that covers the majority of benign binaries and subject anything not covered by it to a reputation decision. In the following, we explain how to classify a binary using reputation data.

A. Feature Extraction

For each feature in the client request, the reputation system derives features that form the indices for aggregate lookups. The aggregates contain statistics about the total number of times a feature occurred and how many times it occurred in a malicious context. Each feature type has a different set of derived features.

For IP addresses, the features are the IP address itself, the corresponding $/24$ netblock, as well as the corresponding $/16$ netblock. The rationale for using netblocks is that serving IP address frequently change, e.g. due to load-balancing or due to adversaries switching IP addresses to avoid IP-based blocking. For netblocks that are used predominantly for serving malicious content, the likelihood that a yet unknown IP address in that netblock is also serving malicious content is often higher.

CAMP also uses URL-based features. The reputation request from the client contains the URL pointing directly to the binary download as well as any URL encountered by the web browser when following redirects to the binary. For each URL, the reputation system extracts the host, the domain, and the site. The domain and site are usually identical, but differ for dynamic DNS providers where the site is the same as the host name. The goal of using the site and domain as a feature is to track potential ownership of resources hosted on them.

For the signature-based features, we extract the signer key and the corresponding CA certifying the key for each certificate chain encountered in the binary signature. We also keep track if the signature was trusted on the client, e.g. the client had a trusted CA that was the root of one of the certificate chains.

Some of the client features such as the content hash are used directly in subsequent aggregate lookups.

B. Aggregate Computation

The aggregation step discussed in Section III-C4 computes aggregates for all possible dimensions based on client features and features derived on the server. However, not all of these aggregates are consulted when making a reputation decision. Which aggregates are consulted for rendering a reputation verdict depends on the configuration of the reputation system described below.

Request		Aggregate Index	1 day	7 day	28 day
URL	http://a.foo.com/	analysis—host:a.foo.com—urls	0/1	1/10	3/100
		analysis—domain:foo.com—urls	0/2	1/20	7/200
		analysis—site:foo.com—digests	3/22	4/25	11/153
		client—site:foo.com—requests	10/20	20/50	123/2200
IP	10.0.0.1	analysis—ip:10.0.0.1—urls	0/1	0/1	0/1
		analysis—ip:10.0.0.1—digests	0/1	0/1	0/1
		client—ip:10.0.0.1—requests	1183/1208	9801/11327	33555/37455
		analysis—ip24:10.0.0.1/24—urls	2/112	5/237	6/238
		analysis—ip16:10.0.0.1/16—digests	0/3	0/5	0/5
		client—ip24:10.0.0.1/24—requests	1313/2571	9912/32227	34127/43119

TABLE I
THIS TABLE SHOWS HOW REQUEST FEATURES RELATE TO DIFFERENT INDICES AND CORRESPONDING REPUTATION AGGREGATES. FOR BREVITY, NOT ALL CLIENT FEATURES NOR ALL DERIVED FEATURES ARE SHOWN HERE.

Table I provides an example of how client features such as an IP address or URL are turned into indices for looking up aggregates. The counts associated with each index measure how often the index was encountered either in client requests or by the binary analysis pipeline. The first value counts the number of malicious verdicts and the second value counts the total number of occurrences. For example, the index

`analysis|domain:"foo.com"|urls`

measures how many different URLs under the domain *foo.com* were analyzed by the binary analysis system and how often they were labeled malicious, i.e. for the 7-day time period, the analyzer checked 20 URLs from *foo.com* and found one of them malicious.

An example for a client aggregate is the index

`client|ip24:10.0.0.1/24|requests`

which measures how many client requests for the netblock *10.0.0.1/24* were received by CAMP during the specified time period. For the last 24 hours, CAMP received 2571 requests total and used reputation to label 1313 of them as malicious.

As the binary analysis system provides us with ground truth for client requests for which the payload is known to us, we make decisions based on the direct and derived features as well as their corresponding aggregates. We explain how the decision is calculated in the next section.

C. Reputation Decision

The aggregates computed for each reputation request provide a large number of continuously valued features suitable for many types of machine learning. One drawback of employing machine learning is that the trained models are often large and difficult to understand by humans. As any detection mechanism is going to produce false positives, we favor a classifier that allows human inspection and reasoning about false positives. To achieve this, we employ a depth-2 boolean circuit with a small number of boolean inputs. Figure 5 shows the boolean circuit and its inputs as used in CAMP. Its structure is simple as it contains only `AND` gates at depth 1 and one `OR` gate at depth 2. Each individual `AND` gate represents a single detection rule. In the following, we use `AND` gate and *rule* synonymously.

Aggregates are mapped to boolean inputs via simple threshold functions. The reputation metric is calculated from a boolean circuit and the thresholds for each aggregate. These thresholds are determined via a training phase discussed below. Logically, the boolean circuit can be separated into three different parts: the site aggregates, the IP aggregates and the unknown aggregates.

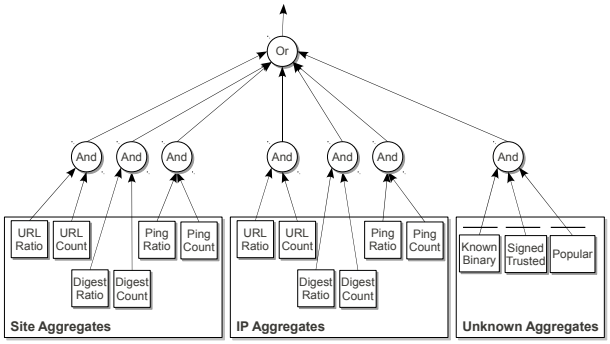

Fig. 5. This figure shows a depth-2 boolean circuit for classifying a binary based on boolean inputs computed from the aggregates of a reputation request (''Ping'').

For the IP and site aggregates, we use two different functions to map an aggregate to a boolean value. The threshold function is defined as $f(p, n, t) := \frac{p}{n} \geq t$ and the count function is defined as $f(p, n, t) := n \geq t$ where p is the number of positive samples in the aggregate, n is the total number of samples and t is the threshold provided by the model.

The booleans derived from the unknown aggregates are more complex. The goal of the unknown aggregates is to flag binaries for which we do not have sufficient reputation to label them benign. The input is formed by the following three negated booleans:

1) The boolean is true if the binary analysis pipeline has already analyzed the binary based on the content hash.
2) The boolean is true if the binary was signed and the client could trace the certificate chain to a trusted CA root.
3) The boolean is true if the binary is popular because a large number of users downloaded either the digest itself or other binaries from the same hosting site.

As a result, a binary is labeled unknown by the binary circuit

if CAMP has never seen it before and the binary is not signed by a trusted signer and not from a popular download site. The parameters are thresholds on the number of downloads required before the binary or the site is considered popular.

The reputation system is configured by choosing thresholds according to the precision and recall for each AND gate. Precision and recall are determined from a labeled training set. The training set is created by matching the content hash included in reputation requests from web browsers with the content hash from the binary analysis system. The binary analysis system provides the label, e.g. benign or malicious, and the reputation request provides the features. The training set consists of $4,000$ benign requests and $1,000$ malicious requests.

Figure 6 shows the precision and recall for site aggregates that result from changing the thresholds for the aggregate mapping functions. We see that achieving a precision larger than 0.95 results in recall of less than 0.2.

Figure 7 shows the precision and recall for IP aggregates. We see that achieving a precision larger than 0.95 results in recall of less than 0.15.

We omit the precision and recall for the unknown rule as they are very similar to the previous two graphs. As can be seen from these graphs, a high precision leads to low recall when considering each AND gate individually. Our evaluation in Section V shows that the combination of all AND gates, that is the complete boolean circuit, leads to acceptable detection results with very low false positive rates.

V. Evaluation

For CAMP's initial deployment, we used Google Chrome and targeted Windows executables, e.g. PEbins, MSI-files, etc.. The following evaluation measures the performance of CAMP in that context. We evaluate resource requirements, accuracy of the baseline dynamic analysis framework, and the precision of the reputation system. We also provide a comparison to other common malware prevention solutions and a case study of a campaign that CAMP identified.

A. Server-Side Resource Requirements

Here we briefly overview CAMP's operational performance and resource requirements. To service 200 million Google Chrome users, CAMP frontends and reputation RPC servers (Section III-C2) use approximately 4 cores and 9 GB RAM globally to serve on average 190 QPS. Most of these resources are reserved for geographic distribution and load balancing, and we have load-tested CAMP frontends at 400 QPS using 1 core and 3 GB RAM. The median latency is 127 ms and the 90th percentile is 313 ms. As many binary downloads take much longer time, on the order of seconds, requesting a CAMP verdict does not significantly add to the download latency observed by users.

Download requests can be processed with minimal overhead since CAMP leverages preprocessing to build models that are amenable to fast lookup. This is a multi-stage process, encompassing despamming (Section III-C3) and aggregation (Section III-C4). The despammer runs continuously, requiring 1.25 cores and 25 GB RAM. The valid requests that the despammer outputs are processed by an indexer in the aggregation pipeline, which writes data to a temporary BigTable [8] for aggregation by a series of MapReduces [14]. The indexer uses 2 cores and 8 GB RAM. The aggregation MapReduces run periodically throughout the day. Each run finishes in approximately one hour and uses 90 cores and 120 GB RAM. The MapReduces produce a model that is comprised of approximately 4.75 billion aggregates requiring approximately 1.4 TB of BigTable storage. This model is replicated to locations near each of CAMP's frontends.

The above metrics do not include resources for running our binary analysis pipeline, or the CPU and memory resources required by BigTable.

B. Accuracy of Binary Analysis

Before we evaluate CAMP's accuracy relative to our baseline proprietary VM-based dynamic analysis framework, it is important to understand the accuracy of that framework. To do so, we compared the framework against the AV scanners provided by VirusTotal [1].

We selected a sample of 2200 binaries that our dynamic analysis framework processed on a single day, but were not known to VirusTotal. Of these, 1100 were labeled clean by our dynamic analysis framework, and 1100 were labeled malicious. We submitted each of the binaries to VirusTotal, and waited 10 days, to allow AV engines to catch up. We then consulted VirusTotal for each of the 2200 binaries to see how many AV engines agreed with our initial analysis.

After 10 days, 99% of the binaries that were flagged as malicious by our framework were flagged by 20% or more of AV engines on VirusTotal. Only 12% of the binaries that we flagged as clean were also flagged by 20% or more of the AV engines. Of these false negatives, a large percentage are classified as AdWare, which our system intentionally does not detect.

These results indicate that, by design, our dynamic analysis framework has a very high true positive rate, but does suffer from some false negatives. As we improve this framework, CAMP's detection rates should improve as well.

C. Accuracy of CAMP

The evaluation is based on data collected between February 2012 to July 2012 from our production deployment of CAMP. During that time frame, CAMP was used by approximately 200 million users. Each day, it received between 8 million to 10 million reputation requests and labeled approximately 200 to 300 thousand download requests as malicious; During our evaluation, the reputation database kept track of approximately 3.2 billion aggregates.

To get a better understanding of how well the reputation metric works in practice, we measured the true positive (tpr), false positive (fpr), true negative (tnr) and false negative rates (fnr) of the reputation requests received by CAMP. The rates

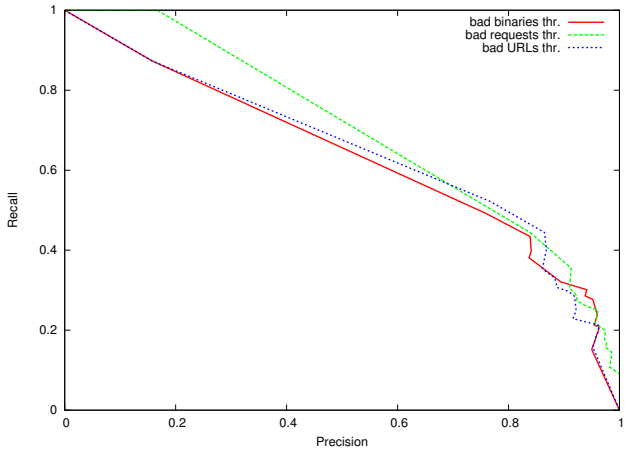

Fig. 6. The figure shows precision and recall for site aggregates for different thresholds of the aggregate mapping functions.

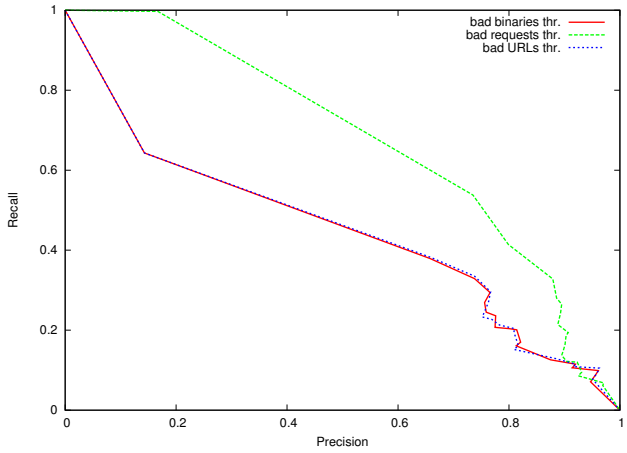

Fig. 7. The figure shows precision and recall for IP aggregates for different thresholds of the aggregate mapping functions.

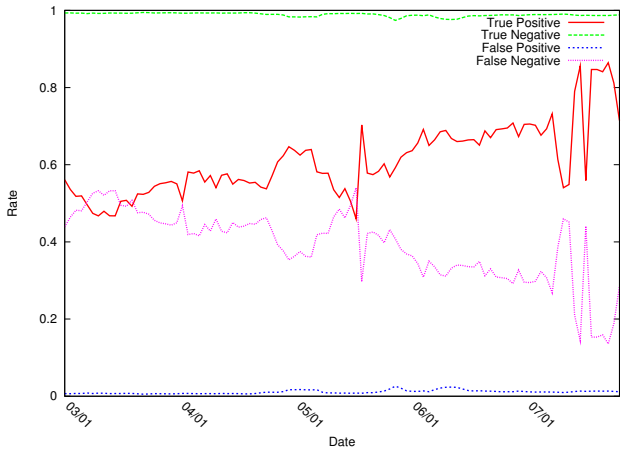

Fig. 8. The figure shows the accuracy of the reputation-based classifier as determined post-facto from the production deployment of CAMP.

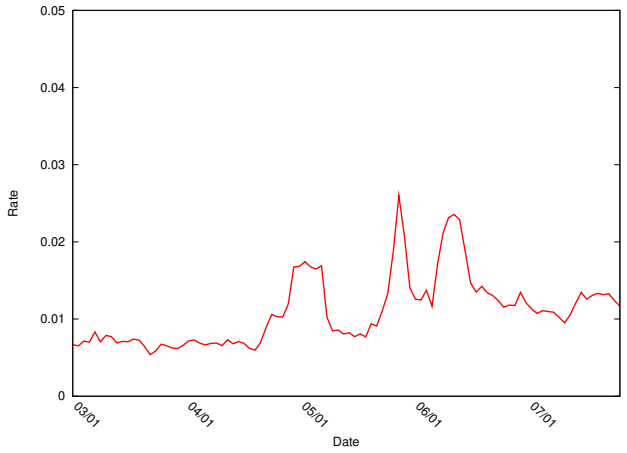

Fig. 9. The figure shows the false positive rate of the reputation-based classifier as determined post-facto from the production deployment of CAMP.

are defined as follows; tpr $:= \frac{\text{tp}}{\text{tp+fn}}$, fpr $:= \frac{\text{fp}}{\text{fp+tn}}$, tnr $:= \frac{\text{tn}}{\text{tn+fp}}$ and fnr $:= \frac{\text{fn}}{\text{tp+fn}}$.

The labels to compute the rates are taken from the binary analysis system for binaries that could be matched to reputation requests. See Section V-B for an evaluation of the accuracy of the labels. The results are shown in Figure 8. During the measurement period, the true negative rate was greater or equal to 98% and the false positive rate was around 1% for most of the measurement period but for three occasions where it increased to around 2%.

A more detailed graph of the false positive rate is shown in Figure 9. We see that the false positive rate was noticeably lower than 2% for most of the measurement period. When taking into account that the web browser makes local reputation decisions for about 70% of downloads, the effective true negative rate increases from 98% to 99.5% and the effective false positive rate decreases from 2% to 0.6%. While false positives are inherent to our content-agnostic approach, our criteria for choosing CAMP's reputation algorithm parameters is optimized to minimize false positives.

At the end of the measurement period, the overall accuracy of CAMP was 98.6%. We compute this as $\frac{\text{tp+tn}}{\text{tp+tn+fp+fn}}$. The dominating factor in overall accuracy comes from our traffic distribution; the majority of requests are for benign binaries and our system exhibits low false positive rates. As with false positive and true negative rates, accuracy increases further when taking the client-side whitelists into account.

In June 2012, the true positive rate was around 70% and the false negative rate around 30%. CAMP's ability to detect 70% of recent malware without access to the content of the binary validates the reputation-based detection approach. We provide a more detailed comparison between CAMP and AV engines in Section V-D.

In designing CAMP, our main consideration was avoiding false positives and as such the binary analysis pipeline frequently prefers false negatives over false positives which has a negative impact on all measurements presented here, e.g. a false negative in the binary analysis pipeline could

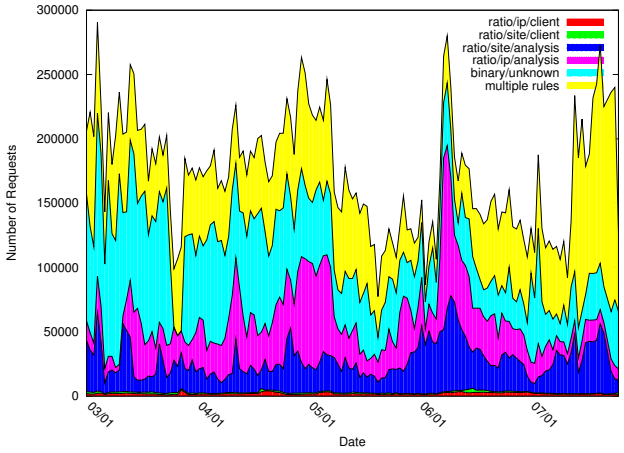

Fig. 10. The graph shows the stacked contribution to detection in cases where a single AND gate or rule is responsible for labeling a download as malicious or unknown. It also shows the number of cases in which multiple rules flagged the binary download.

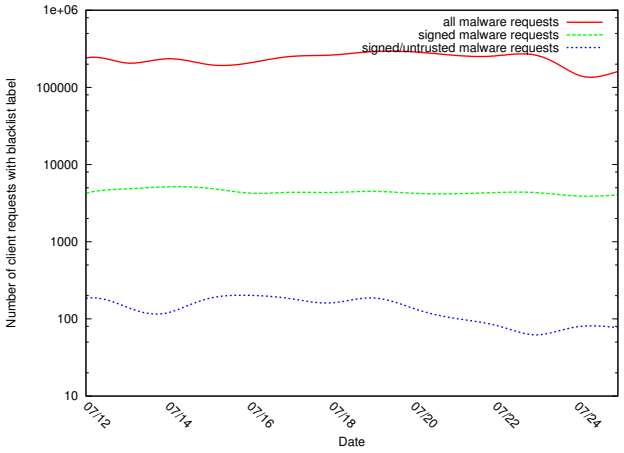

Fig. 11. The graph shows the number of client requests corresponding to binaries labeled as malware by the binary classifier. It also shows requests to malware were the binary was signed.

result in an incorrect false positive for the reputation-based classifier. Specifically, when CAMP detects a malware binary correctly, but the binary analysis system has a false negative, our evaluation will count this as a false positive for CAMP.

As discussed in the previous section, each individual AND gate achieves only low recall. Our hypothesis was that different AND gates would combine to increase the overall performance of the classifier. To measure this assumption, we plot how often an AND gate is responsible for detection by itself and how often multiple AND gates detect bad content simultaneously. The results are shown in Figure 10 as a stacked graph. The graph has two interesting features. The majority of detections are due to a single rule, i.e. each individual rule contributes to the overall detection. The other interesting observation is that the detection of unknown binaries accounts for a significant fraction of malicious verdicts. We show in the case study below how adversaries frequently rotate the domains from which they are serving malware and thus never build up any reputation. The drop in the detection of unknown binaries around March 24th is due to a change in data formats that resulted in dropping the contributions of the unknown rule from the graph.

The rule for labeling binaries as unknown requires that a binary is not signed by a trusted signer. To understand how often malicious binaries with trusted signatures occur in our analysis, we extracted the signature state from binaries we analyzed from client requests post-facto. As shown in Figure 11, the majority of requests to malware are for unsigned binaries. That makes a trusted signature a good predictor for the likelihood that a binary is benign.

We also wanted to understand how often a binary that is labeled as unknown transitions to clean or to malicious after it has been evaluated by our dynamic analysis pipeline. To do so, we analyzed requests from clients post-facto, for a period of 10 days in November 2012. Of all the sites that served binaries that were classified as unknown on the first day, 74% no longer appeared in our requests the next day. This percentage increased to 80% over 10 days. We hypothesize that this is because such sites are malicious and rotate to avoid blacklists. This corroborates previous findings that social engineering domains rotate quickly to avoid blacklists [27]. 0.02% of the sites transitioned from unknown to bad over the 10 day period, 8% transitioned to unknown to clean, and the remaining stayed in the unknown state. We manually analyzed the top sites that transitioned to clean, and found that almost all of them were indeed hosting dangerous downloads, and the clean classification was a false negative of our dynamic analysis framework. Based on these results, we believe that the unknown verdict is actually a very strong indicator that a binary is actually malicious.

D. Comparison to other systems

To determine if CAMP provides benefits beyond that provided by other malware detection systems, we conducted two different measurements in which we compared reputation decisions from CAMP to results from Anti-Virus engines as well as web-based malware services.

Over a period of two weeks we collected a random sample of $10,000$ binaries that were labeled as benign by CAMP as well as approximately $8,400$ binaries that were labeled as malicious by CAMP. To avoid bias in binary selection, for example due to popular download sites which might be over represented in a completely random sample, we sampled by site rather than by individual binaries. We compared the results from CAMP against the results from four different AV engines[1] that scanned the binaries on the same day as CAMP made its reputation decision. We conducted the AV scans on the same day to approximate the performance users might see when AV engines scan their downloads. The AV engines ran in a sandboxed environment, and were not permitted any

[1]Due to licensing restrictions, we cannot disclose the specific AV engines we used in our measurements.

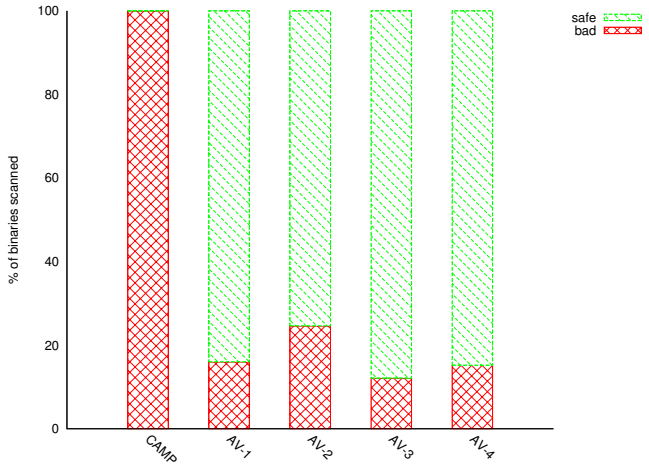

Fig. 12. The graph shows the results from AV engines for binaries flagged as malicious by CAMP.

network access. Thus any cloud-based reputation data that may have been provided by the AV companies was unavailable to the AV engines. However, we proactively updated AV signatures every two hours to ensure freshness.

For the $10,000$ binaries that CAMP labeled as clean, the maximum number of binaries labeled as malicious by a single AV engine was only 83. Only 16 binaries were flagged as malicious by two or more AV engines. This implies that CAMP has a very high True Negative rate relative to commercial Anti-Virus products.

On the other hand, the majority of binaries that CAMP labeled as malicious were classified as benign by the AV engines (see Figure 12). The AV engine that agreed the most with CAMP only flagged 25% of the binaries as malicious. When combining the results from all four AV engines, less than 40% of the binaries were detected. One possible explanation for these results is that CAMP might exhibit a high false positive rate, but as shown in Figure 9 and discussed earlier, CAMP's false positive rate is quite low and thus false positives are not a likely explanation. However, as observed by Oberheide et al. [24] many AV engines exhibit poor detection rates for recent malware and we believe that to be confirmed by our measurements, too.

In addition to comparing CAMP's detection results with AV engines, we also consulted several web services to classify URLs that hosted binaries. For this measurement, we consulted the following services: Malware Domain List, McAfee's Site Advisor [22], Symantec's Safe Web [23], Google's Safe Browsing [16] and TrendMicro's Site Safety Center. We selected $20,000$ URLs from a single day's worth of requests; $10,000$ URLs pointed to binaries that were classified as benign by CAMP and $10,000$ URLs that pointed to binaries that were identified as malicious. We employed the same site-based sampling strategy that was used for evaluating AV engines. For each of the selected URLs, we consulted the web services listed above and compared their verdict with CAMP. The results are shown in Figure 13 and 14.

The URL classification services mostly agreed with CAMP when presented with the set of clean URLs. TrendMicro flagged about 3.5% as malicious, Symantec about 2.5% and Site Advisor about 1.5%. Furthermore, many of the benign URLs were unknown to these three services. For example, TrendMicro did not know over 55% of the URLs. Neither the Malware Domain List nor Safe Browsing flagged any of the URLs are malicious.

The URL classification services mostly disagreed with CAMP when presented with the set of malicious URLs. TrendMicro identified about 11% as malicious, Safe Browsing about 8.5%, Symantec about 8% and Site Advisor about 2.5%. The Malware Domain List did not flag any of them as malicious. However, as with the benign URLs, many of the malicious URLs were not known to the web services. For example, TrendMicro did not know 65% of the URLs that CAMP found to be malicious.

Google Chrome asks for a reputation decision only if a URL is not known by the Safe Browsing API. Therefore, it is not surprising that many of URLs CAMP considers malicious were not in the Safe Browsing list. Moreover, the Safe Browsing list primarily targets drive-by downloads, not intentional user installs. Although, the Malware Domain list did not return any detections, we included it in our measurements as it is frequently used as a base of comparison by other work in this space.

The results for the other web services seem to confirm our suspicion that blacklist based approaches are not as effective in the current environment of frequently changing malware distribution domains. The majority of URLs identified as malicious by CAMP are not known to be bad by any web service. On the other hand, CAMP explicitly assigns negative reputation to domains unknown to it. We could interpret the unknown results from the web services in a similar way. In that case, detection rates would increase noticeably. For example, when combining unknown and known malicious results, the detection rate for TrendMicro would be 76%, for Symantec 46% and for Site Advisor 45%. However, in that case, the potential false positive rates as measured by comparing to the benign URLs would increase significantly, too. From that perspective, TrendMicro would flag 59% of the benign URLs, Symantec 24% and Site Advisor 29.5%. As the potential false positive rates are much higher than can be sustained in practice, our original interpretation of the inherent drawbacks in blacklists is a more likely explanation.

E. Case Study

CAMP provides an interesting vantage point into malware distribution across the web. In the following, we explore an example of one of many malware campaigns discovered by CAMP. This campaign distributes Fake Anti-Virus binaries and leverages frequent repacking of binaries as well as fast domain rotation to evade blacklist-based defense mechanisms. We observed the campaign between February 13, 2012 and March 1, 2012.

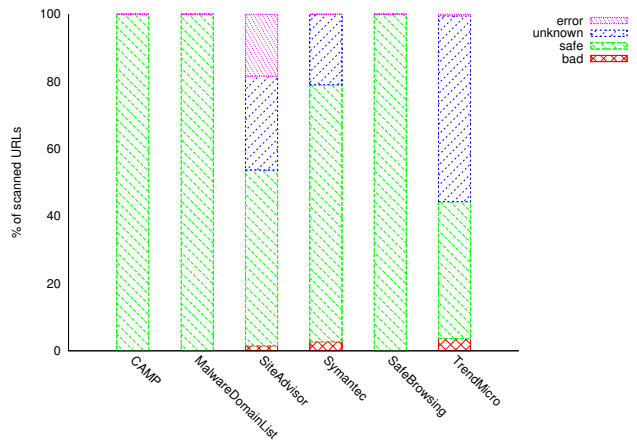

Fig. 13. The graph shows how different web services classify URLs flagged as benign by CAMP.

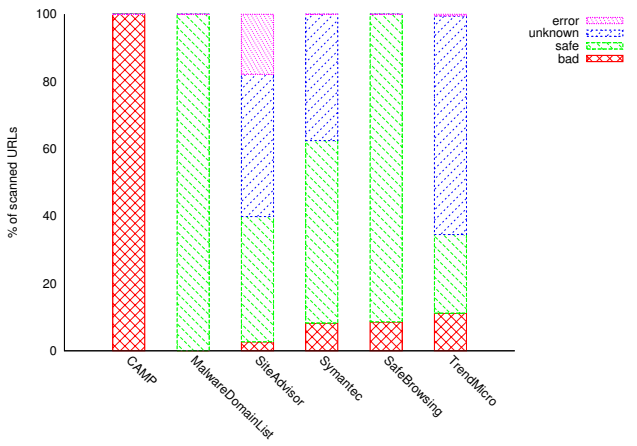

Fig. 14. The graph shows how different web services classify URLs flagged as malicious by CAMP.

The Fake AV binaries were primarily hosted on the free domain provider `uni.me`. Free domain providers allow third parties to register many domain names at low cost, and are frequently abused by adversaries seeking to distribute malware. This particular campaign registered domains matching the following pattern

`[srv|www|server|update]NN.dict.uni.me`

where `dict` corresponds to a random English word. Each domain was active for only a short period of time. To estimate domain lifetime, we take the `client|host` 1 day aggregates from February and compute the lifetime as the difference in time between the last observed download request and the first. The median lifetime for a domain was 406 seconds, and the $90th$ percentile was 1749 seconds. Some of the domains were active simultaneously. Table II provides examples of domains employed in the campaign, along with the time of the first request we observed for that domain and its estimated lifetime. Over the two week period, we observed over $13,000$ unique domains on `uni.me` involved in this campaign.

We observe that the high frequency of domain changes thwarts simple blacklisting-based approaches. The maintainers of the blacklist would need to be able to fetch the content, analyze it, and push out list updates within minutes. We noticed that even fetching the content would be challenging, as each domain stopping resolving properly after a short time period.

The campaign not only switched domains, but also changes the binaries that were served as indicated by their changing content hash. We observed that the binaries served by each host changed approximately every 10 minutes. We fetched and analyzed several samples, and they all offered the same functionality, a Fake Anti-Virus that hijacks the user's system and refuses to release it until a registration fee is paid. We submitted one of the samples to VirusTotal and only one of 40 AV engines identified it as malware.

This particular malware hijacks the user's machine by setting the execution environment for all essential system processes to run in a debugger that points to a dummy process. This is achieved by setting the registry key

`HKLM\software\microsoft\windows nt\currentversion\image file execution options\msa.exe\Debugger:svchost.exe`

This prevents key processes from performing meaningful tasks rendering the machine unusable. The Fake AV binary provides a mechanism to ''fix'' the problem once the software is registered for a fee.

Time of first appearance	Life (sec)	Host(.uni.me)
2012/02/18 15:02:00	632	srv62.specialarmor
2012/02/18 15:02:02	330	srv76.specialarmor
2012/02/18 15:02:05	629	www12.fuelwire
2012/02/18 15:02:06	587	server78.fuelwire
2012/02/18 15:02:15	246	www82.fuelwire
2012/02/18 15:02:26	25	update96.fuelwire
2012/02/18 15:02:38	560	server45.specialarmor
2012/02/18 15:02:50	693	server52.specialarmor
2012/02/18 15:02:53	575	www77.fuelwire
2012/02/18 15:02:57	1258	www92.specialarmor

TABLE II
THIS TABLE LISTS EXAMPLE DOMAINS FOR A FAKE AV CAMPAIGN AND THEIR CORRESPONDING LIFETIME.

As the domains for this campaign rotated quickly, our analysis pipeline fetched only a few samples. We observed over 900 distinct content hashes resulting in binaries with identical behavior. In total, we saw over $41,000$ different binaries exhibiting similar behavior.

When the campaign first started and we had not yet fetched these samples, CAMP replied with an *unknown* verdict, thus still protecting our users from these downloads. In general, when we cannot fetch the binaries that our users download, due to e.g. one-time URLs, the unknown verdict offers protection. In the future, CAMP could be updated to use the lack of ability to fetch content from an URL as a feature by itself. The browser could also be changed to allow users to upload files. This would facilitate analysis of campaigns that host executables on one-time URLs.

While this campaign aggressively switched domains and changes binaries, it did not rotate IP addresses as frequently. We observed it across only five IPs during a two week period: `95.143.37.145`, `46.21.154.155`, `194.28.114.103`, `194.28.114.102`, and `217.116.198.33`. Our hypothesis is that the adversaries behind the campaign focused on domain rotation to avoid widely-deployed blacklists and binary mutation to avoid AV engines, but were not concerned with IP rotation as there are few widely-deployed IP-based blacklists.

VI. Discussion

Blacklist-based approaches in which web browsers block content from known malicious sites offer some protection against malware, but suffer from a knowledge gap when adversaries frequently switch to new domains or repack binaries. Blacklists are still effective in protecting web browsers in situations where it is not possible to quickly rotate domains, e.g., when using a compromised web site to drive traffic. A potentially more resilient approach leverages whitelists so that web browsers download content only from trusted sites. Unfortunately, such a whitelist is never going to be complete either, resulting in legitimately benign content not being available to web users. CAMP bridges the gap between blacklists and whitelists by augmenting both approaches with a reputation system that is applied to unknown content. As our evaluation has shown, CAMP does not suffer from significant false positives, but could benefit from higher detection rates. Utilizing more traditional machine learning approaches in addition to the binary circuit currently employed by CAMP may improve detection. However, the ability for humans to reason about detection verdicts is important to our deployment and additional research is required to better reason about the large models generated by machine learning approaches.

The performance of CAMP depends significantly on the mechanism used for labeling binary samples. Any improvement to detection rates in the binary classifier will directly translate to improved detection in CAMP. Our current binary classifier is conservative and has the explicit goal of not tolerating any false positives. However, it is conceivable that in the context of CAMP, we could tolerate a small number of false positives to improve overall detection. Instead of using a binary analysis platform, we posit that binaries could also be labeled by AV engines, for example, by taking a majority vote to determine if a binary is malicious or not.

One of CAMP's important properties is to minimize the impact on user privacy while still providing protection. To achieve this goal, the browser leverages a whitelist to limit the number of decisions which require server interaction. Even when the browser does ask the server for a decision, only a small set of features is sent. These features are stored for up to two weeks, and afterwards only aggregated information is stored, but no longer than a few months. Despite severely limiting the data available to the system, our evaluation shows that CAMP exhibits high accuracy rates.

While the content-agnostic nature of CAMP helps to reduce its privacy impact, it also means that CAMP can be applied to any operating system and to any content type that can be labeled. For example, with an accurate labeling mechanism for browser add-ons, CAMP could render reputation-based verdicts when a browser add-on is downloaded.

While CAMP is currently only available in Google Chrome, we plan on making the service available to all web browsers once we have gained more operational understanding of the system and have further improved CAMP's detection rates.

VII. Conclusion

Although browsers have become more secure, the world wide web continues to be a significant contributor to malware infections. Many of the defenses available to users such as blacklists or AV engines face challenges as adversaries can evade detection by frequently changing hosting domains or mutating their malware binaries until they are no longer detected.

This paper introduced CAMP, a content-agnostic malware protection system, which employs a reputation system that detects malware independently of the actual binary contents. CAMP protects browser users from malware downloads while also minimizing the impact on user privacy. To get a reputation decision for a binary download, the web browser contacts CAMP's servers which automatically build reputation for downloads and render reputation-based verdicts. If a download is deemed malicious, the web browser displays a warning to the user and offers to delete the downloaded file.

We provided a detailed overview of CAMP's design and architecture and discussed in detail all the components that constitute the reputation system. At its core, the reputation metric is calculated via a binary circuit that receives its input from statistical aggregates. The statistical aggregates are computed based on features derived from web browser requests and contain information on how often they occurred in a malicious context compared to the total number of occurrences.

In this paper, we performed an extensive six month evaluation of CAMP consisting of over 200 million unique users of Google Chrome and millions of daily reputation decisions. We showed that our content-agnostic detection approach is both accurate, with an accuracy of close to 99% relative to proprietary VM-based dynamic analysis, and well performing, processing requests in less than 130 ms on average.

In comparing CAMP with the current state of practice, we demonstrated that CAMP outperforms Anti-Virus, as well as various web services, e.g. McAfee's Site Advisor, Symantec's Safeweb, etc. Furthermore, CAMP augments Google's Safe Browsing API, flagging 5 million malware downloads per month that were not previously identified.

Acknowledgments

The authors would like to thank Michael Bailey for helpful suggestions for this paper. We also thank our shepherd Lenx Wei for his valuable suggestions to improve this paper.

REFERENCES

[1] VirusTotal. https://www.virustotal.com/.

[2] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a Dynamic Reputation System for DNS. In *Proceedings of the 19th USENIX Security Symposium (August 2010)*.

[3] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou II, and D. Dagon. Detecting malware domains at the upper dns hierarchy. In *Proceedings of the 20th USENIX Security Symposium, USENIX Security*, volume 11, 2011.

[4] A. Averbuch, M. Kiperberg, and N. Zaidenberg. An efficient vm-based software protection. In *Network and System Security (NSS), 2011 5th International Conference on*, pages 121--128. IEEE, 2011.

[5] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *NDSS*. The Internet Society, 2011.

[6] Bit9. Bit9 Global Software Registry. http://www.bit9.com/products/bit9-global-software-registry.php, July 2012.

[7] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring Pay-per-Install: The Commoditization of Malware Distribution. In *Proceedings of the the 20th USENIX Security Symposium*, San Francisco, CA, August 2011.

[8] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):1--26, 2008.

[9] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant. Semantics-aware malware detection. *IEEE Symposium on Security and Privacy*, pages 32--46, 2005.

[10] R. Colvin. Stranger Danger - Introducing SmartScreen Application Reputation. http://blogs.msdn.com/b/ie/archive/2010/10/13/stranger-danger-introducing-smartscreen-application-reputation.aspx, October 2010.

[11] CoreTrace. Application Whitelisting: A New Security Paradigm. http://www.coretrace.com/, July 2008.

[12] M. Cova, C. Leita, O. Thonnard, A. Keromytis, and M. Dacier. An analysis of rogue av campaigns. In *Recent Advances in Intrusion Detection*, pages 442--463. Springer, 2010.

[13] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert. Zozzle: Fast and precise in-browser javascript malware detection. In *USENIX Security Symposium*, 2011.

[14] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the Sixth Symposium on Operating System Design and Implementation*, pages 137--150, Dec 2004.

[15] P. Ferrie. Attacks on More Virtual Machine Emulators. Symantec White Paper, 2007.

[16] Google. Google Safe Browsing API. http://code.google.com/apis/safebrowsing/, July 2012.

[17] C. Grier, S. Tang, and S. King. Secure web browsing with the op web browser. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 402--416. Ieee, 2008.

[18] S. Hao, N. A. Syed, N. Feamster, A. G. Gray, and S. Krasser. Detecting spammers with snare: spatio-temporal network-level automatic reputation engine. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, pages 101--118, Berkeley, CA, USA, 2009. USENIX Association.

[19] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)*, 42(1):1, 2009.

[20] C. Kanich, N. Weaver, D. McCoy, T. Halvorson, C. Kreibich, K. Levchenko, V. Paxson, G. Voelker, and S. Savage. Show me the money: Characterizing spam-advertised revenue. In *Proceedings of the 20th USENIX Security Symposium, San Francisco, CA*, 2011.

[21] L. Lu, V. Yegneswaran, P. Porras, and W. Lee. Blade: an attack-agnostic approach for preventing drive-by malware infections. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 440--450. ACM, 2010.

[22] McAfee. McAfee Site Advisor. http://www.siteadvisor.com/, July 2012.

[23] Norton. Norton Safe Web. http://safeweb.norton.com, July 2012.

[24] J. Oberheide, E. Cooke, and F. Jahanian. Cloudav: N-version Antivirus in the Network Cloud. In *Proceedings of the 17th conference on Security symposium*, pages 91--106. USENIX Association, 2008.

[25] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All Your iFRAMEs Point to Us. In *USENIX Security Symposium*, pages 1--16, 2008.

[26] Z. Qian, Z. M. Mao, Y. Xie, and F. Yu. On network-level clusters for spam detection. In *NDSS*. The Internet Society, 2010.

[27] M. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos, and L. Schmidt. Trends in circumventing web-malware detection. Technical report, Google, Tech. Rep., July 2011.

[28] M. A. Rajab, L. Ballard, P. Mavrommatis, N. Provos, and X. Zhao. The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution. In *Proceedings of the 3^{rd} USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, April 2010.

[29] C. Reis, A. Barth, and C. Pizano. Browser security: lessons from google chrome. *Queue*, 7(5):3, 2009.

[30] P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee. Polyunpack: Automating the hidden-code extraction of unpack-executing malware. In *Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual*, pages 289--300. IEEE, 2006.

[31] C. Song, P. Royal, and W. Lee. Impeding automated malware analysis with environment-sensitive malware. In *Proceedings of the 7th USENIX conference on Hot Topics in Security*, HotSec'12, pages 4--4, Berkeley, CA, USA, 2012. USENIX Association.

[32] H. Wang, C. Grier, A. Moshchuk, S. King, P. Choudhury, and H. Venter. The multi-principal os construction of the gazelle web browser. In *Proceedings of the 18th conference on USENIX security symposium*, pages 417--432. USENIX Association, 2009.

[33] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda. Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis. In *Proceedings of the 14th ACM Conference of Computer and Communication Security*, October 2007.