FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors

M. Zubair Rafique and Juan Caballero

IMDEA Software Institute
{zubair.rafique, juan.caballero}@imdea.org

Abstract. The ever-increasing number of malware families and polymorphic variants creates a pressing need for automatic tools to cluster the collected malware into families and generate behavioral signatures for their detection. Among these, network traffic is a powerful behavioral signature and network signatures are widely used by network administrators. In this paper we present FIRMA, a tool that given a large pool of network traffic obtained by executing unlabeled malware binaries, generates a clustering of the malware binaries into families and a set of network signatures for each family. Compared with prior tools, FIRMA produces network signatures for each of the network behaviors of a family, regardless of the type of traffic the malware uses (e.g., HTTP, IRC, SMTP, TCP, UDP). We have implemented FIRMA and evaluated it on two recent datasets comprising nearly 16,000 unique malware binaries. Our results show that FIRMA's clustering has very high precision (100% on a labeled dataset) and recall (97.7%). We compare FIRMA's signatures with manually generated ones, showing that they are as good (often better), while generated in a fraction of the time.
Keywords: Network Signatures, Malware Clustering, Signature Generation

1 Introduction

Malware analysts face the challenge of detecting and classifying an ever-growing number of malware families and a flood of polymorphic variants. While this problem has been observed for years it is only getting worse as malware packing rates keep increasing [14, 24] and malware operations become easier to launch thanks to services that enable outsourcing key steps such as malware creation and distribution [5, 12].

Behavioral signatures detect polymorphic variants by capturing behaviors specific to a malware family and stable across its variants [4, 19, 26]. To build behavioral signatures, defenders collect large numbers of unlabeled malware using honeyclients, honeypots, spam traps, and malware analysis services [2]. Classifying those malware binaries into families is important not only for understanding the malware landscape, but also for generating behavioral signatures specific to a family, as it is very difficult to find behaviors common to all malware samples.

A powerful behavioral signature is network traffic because the large majority of malware families require network communication to receive commands, perform nefarious activities (e.g., clickfraud, spam, data exfiltration, DoS), and notify controllers on the results. Network signatures are widely used by administrators for detecting malware-infected hosts in their networks, and for identifying remote malicious servers for abuse reporting and takedown. They are easier to deploy than host-based signatures, requiring a signature-matching IDS at a vantage network point rather than virtual environments at every end host [19, 42].

In this work we propose FIRMA, a tool that given a large number of network traces obtained by executing unlabeled malware samples produces: (1) a clustering of the malware samples into families, and (2) a set of network signatures for each family cluster. Each signature in the set for a family captures a different network behavior of the family. This is important because a malware family may use multiple C&C protocols (e.g., one binary and another HTTP-based), a C&C protocol and another protocol for malicious activities (e.g., SMTP for sending spam), or multiple messages in the same protocol with different structure and content. Generating a single signature for each family combining different behaviors would lead to signatures with high false positives.

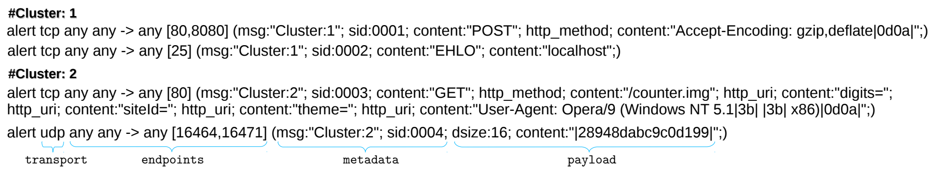
FIRMA offers a combined solution for the problems of automatic malware clustering and signature generation. The only prior work we are aware of offering a combined solution to these problems is by Perdisci et al. [26]. However, their approach exclusively deals with HTTP traffic and generates signatures that cover only the HTTP method and the HTTP URL. In contrast, FIRMA analyzes, and generates signatures for, all traffic sent by the malware, regardless of protocol and field. This is fundamental because 20–45% of the signatures FIRMA generates are for non-HTTP traffic. In our largest dataset, 34% of the malware families have no characteristic HTTP traffic and for another 44% of the families the HTTP signatures generated by FIRMA contain tokens outside the HTTP method and URL (i.e., in headers or the body). Our performance results also indicate that FIRMA is at least 4.5 times faster than the tool in [26].

While there exists a wealth of research in automatic signature generation for worm detection [18,20,22,25,32] these works focus on a single worm and a single network behavior, i.e., the protocol interactions needed to exploit a vulnerability used for propagation. They generate a single signature for the worm and can only handle small amounts of noise in the worm traffic [22], not a pool of malicious traffic generated by a large number of unlabeled binaries from different families and with multiple network behaviors. In addition, they rely on worm traffic containing invariants needed to exploit the vulnerability [25], while malware C&C traffic is not constrained in that way. There are also other works on malware clustering using network traffic, but those do not address network signature generation [13] or generate network signatures manually [5, 12].

For matching the network signatures, FIRMA outputs the produced signatures in the syntax used by 2 popular signature-matching IDSes: Snort [33] and Suricata [34]. This enables one entity to produce signatures, which are then distributed to many other entities using these IDSes. There is no need to deploy a new signature matching component. This model is widely used in the industry, e.g., with the Sourcefire and Emerging Threats rule sets[1]. However, those rule sets are largely produced manually.

We evaluate FIRMA on two recently collected malware datasets, comprising nearly 16,000 malware binaries. The largest of these datasets is publicly available as part of the MALICIA project and has the malware binaries labeled [23, 24]. Using those labels we show that FIRMA achieves perfect precision and 97.7% recall on its malware clustering, and a F-Measure of 98.8%. On live traffic, the generated signatures achieve a low false

[1] http://www.emergingthreats.net/, http://www.sourcefire.com/



```
#Cluster: 1
alert tcp any any -> any [80,8080] (msg:"Cluster:1"; sid:0001; content:"POST"; http_method; content:"Accept-Encoding: gzip,deflate|0d0a|";)
alert tcp any any -> any [25] (msg:"Cluster:1"; sid:0002; content:"EHLO"; content:"localhost";)
#Cluster: 2
alert tcp any any -> any [80] (msg:"Cluster:2"; sid:0003; content:"GET"; http_method; content:"/counter.img"; http_uri; content:"digits=";
http_uri; content:"siteId="; http_uri; content:"theme="; http_uri; content:"User-Agent: Opera/9 (Windows NT 5.1|3b| |3b| x86)|0d0a|";)
alert udp any any -> any [16464,16471] (msg:"Cluster:2"; sid:0004; dsize:16; content:"|28948dabc9c0d199|";)
```

Fig. 1. An example signature file produced by FIRMA.

positive rate of 0.00001%. In addition, we have access to manually generated network signatures for the malware in that dataset, produced as part of [24]. We use those to demonstrate that the signatures automatically generated by FIRMA are as good (and often better) than the signatures manually generated by analysts, and are generated in a fraction of the time.

To facilitate future research and enable other groups to compare their results to ours, we are releasing a new version of the MALICIA dataset that adds FIRMA's clustering results, and both the manually generated signatures and the ones produced by FIRMA.

2 Overview and Problem Definition

FIRMA takes as input a set of network traces obtained by running unlabeled malware binaries in a contained environment. It outputs: (1) a *clusters file* with a partition of the malware binaries that produced the network traces into *family clusters*, (2) a *signature file* with network signatures annotated with the family cluster they correspond to, and (3) an *endpoints file* with the C&C domains and IP addresses that the malware binaries in each family cluster contacted across the input network traces.

A fundamental characteristic of FIRMA is that a family cluster has an associated signature set where each signature captures a different network behavior of the family. Figure 1 shows an example signature file produced by FIRMA. It contains two family clusters, each of them with two signatures. The malware binaries in the first family cluster use a C&C protocol built on top of HTTP POST messages as well as SMTP traffic for testing whether the infected host can spam. The second family cluster (corresponding to the zeroaccess family [39]) shows an HTTP C&C that uses a GET message and a separate UDP-based C&C protocol on ports 16464 and 16471.

Both families exhibit two very different network behaviors that should not be combined, otherwise the resulting signature would be too general and cause many false positives. To avoid this, we propose a novel design for FIRMA in which (at 10,000 feet) the traffic in the network traces is first partitioned into *traffic clusters* using features that identify similar traffic, then signatures are created for each traffic cluster, and finally a sequence of steps merges similar signatures and groups signatures for the same family into *signature clusters*. Importantly, through the whole process FIRMA tracks which malware binaries belong to which cluster.

Other salient features of FIRMA are that it is not limited to a specific type of traffic (e.g., HTTP) or specific fields (e.g., HTTP Method, URL), and that it is *protocol-aware*. The first is important because malware can use any type of C&C traffic. For example, it may build a C&C protocol directly on top of a transport protocol (e.g., TCP or UDP) or on top of an application protocol (e.g., HTTP or IRC). Also, because any part of a message may contain the distinctive content that enables building a signature. For example, the first signature in Figure 1 captures a typo that the malware author made in a custom Accept-Encoding HTTP header: there is no space after the comma in the "gzip,deflate" value, which does not happen in benign traffic.

FIRMA performs a protocol-aware traffic clustering and signature generation. If the C&C traffic uses a known application protocol such as HTTP, IRC, or SMTP the traffic is parsed into fields and the signatures capture that a token may be specific to a field and should only be matched on that field. We have designed FIRMA to leverage increasing protocol support in off-the-self IDSes. For example, both Snort and Suricata partially parse HTTP requests and provide modifiers (e.g., `http_method`, `http_uri`, `http_header`) to indicate that matching should happen on the HTTP method, URL, or headers buffers, rather than on the full packet buffer.

Benign traffic pool. In addition to the network traces, FIRMA also takes as input a pool of benign traffic used to identify benign domains and content that should not be included in the signatures. Our benign pool comprises four traces: two of HTTP and HTTPS traffic produced by visiting the top Alexa sites[2]. and another two with all traffic seen at the personal computers of two volunteers for 2 days. The latter comprise a variety of TCP and UDP traffic including SMTP and IRC. We examine the computers with commercial AV software to verify they are clean throughout the collection. As signatures are typically shared among administrators, it is difficult to generate a benign traffic pool that is representative of the traffic in the different networks where the signatures may be deployed. Thus, FIRMA enhances the benign traffic pool with 3 whitelists: the Alexa list of most visited domains, a list of HTTP User-Agent strings used by benign software[3], and a list of protocol keywords extracted from Internet standards.

2.1 Network Signatures

One important design goal of FIRMA is to generate network signatures that can be matched by the open source signature-matching IDSes Snort [33] and Suricata [34]. This decision influences the type of signatures that FIRMA generates. As a newer IDS, Suricata decided to be compatible with the popular Snort signatures, so its syntax is a superset of the one used by Snort. These network signatures comprise 4 parts (bottom of Figure 1): *carrier protocol*, *endpoints*, *payload signature*, and *metadata*.

The carrier protocol can be TCP, UDP, ICMP, or IP for Snort. Suricata in addition supports some application protocols such as HTTP, SMTP, SSL, and IRC. FIRMA currently generates protocol-aware signatures for HTTP, SMTP, and IRC and raw signatures for other TCP and UDP traffic. The metadata stores additional information such as a unique signature identifier (`sid`) and the message to display when the signature is matched (`msg`), which FIRMA sets to the family cluster that the signature belongs to.

The endpoints capture source and destination IPs and ports. FIRMA sets only the destination ports and uses a wildcard (e.g., `any`) for the rest. The list of C&C domains

[2] http://www.alexa.com/topsites/

[3] http://www.useragentstring.com/

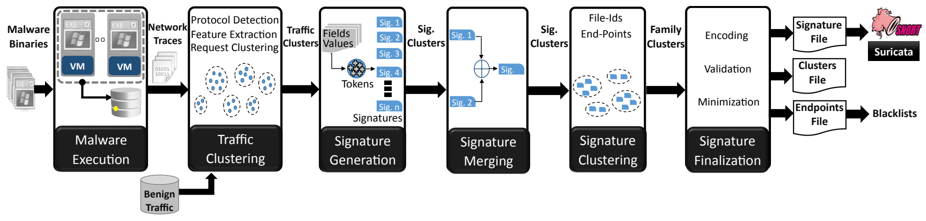


Fig. 2. Architecture overview.

and IP addresses contacted by each family is output into a separate endpoints file, so that signatures match traffic involving servers not observed in the input network traces.

The payload signature captures content invariants, which often exist even in obfuscated or encrypted C&C protocols, as malware often fails to use robust cryptographic algorithms and random initialization vectors. For example, the last signature in Figure 1 captures an obfuscated 16-byte UDP packet with 8 distinctive bytes in its payload (represented as a hexadecimal string). However, while worm traffic has been shown to contain such invariants due to the requirements to exploit a vulnerability [7], C&C protocols are not constrained in this way and can potentially be fully polymorphic [30]. While FIRMA cannot generate payload signatures for fully polymorphic traffic, it enables to quickly identify those families and queue them for further analysis.

While many types of payload signatures have been proposed, most signature-based IDSes like Snort and Suricata only support 3 types: token sets, token subsequences, and regular expressions. These 3 types can be applied on the buffer holding the full packet or on smaller field buffers that the IDS may parse (i.e., protocol-aware). Probabilistic signatures [25, 29] and Turing-complete signatures used to decrypt obfuscated payloads [30] are not supported and require deploying a dedicated matching engine.

FIRMA builds *protocol-aware token-set* payload signatures. A token set is an unordered set of binary strings (i.e., tokens) that matches the content of a buffer if all tokens in the signature appear in the buffer, in any order. The more tokens and the longer each token the more specific the signature, but longer tokens are preferable, i.e., a 3-byte token is more specific than 3 one-byte tokens. Token subsequences are more specific because they impose an ordering on the set of tokens. This is problematic with protocols such as HTTP where reordering some fields does not affect the semantics of the message, allowing the attacker to easily evade the signature. Regular expressions are more expressive than token sets and token subsequences and can be used to represent both, but are more expensive to match. They also impose an ordering constraint introducing similar issues as token subsequences.

Signature lifetime. Network signatures have a lifetime and need to be updated over time. The endpoint information is typically short-lived and of limited value for online detection. However, it is useful for clustering as we observe malware executables of the same family, collected nearby in time, reusing endpoints even if their payloads are polymorphic. Payload signatures are typically longer-lived than endpoints, especially for binary C&C protocols [6]. However, eventually the C&C protocol may change or be replaced with another protocol, so they also need updating.

2.2 Architecture Overview

Our approach comprises 6 steps illustrated in Figure 2: *malware execution*, *traffic clustering*, *signature generation*, *signature merging*, *signature clustering*, and *signature finalization*. Malware execution (Section 3) runs a malware binary on a VM in a contained environment and outputs a network trace capturing the traffic generated during the run. This step may happen multiple times for a malware binary, e.g., on different VMs, for different amounts of time, and with different containment policies.

The network traces are the input to FIRMA. First, traffic clustering (Section 4) groups similar traffic, regardless of which run it comes from and which malware binary produced it. Traffic clustering operates on all traffic in the network traces so it is designed to be cheap; expensive operations (e.g., tokenization) are left for later. It uses protocol-aware features for C&C protocols built on top of standard application protocols and packet-level features for the remaining traffic.

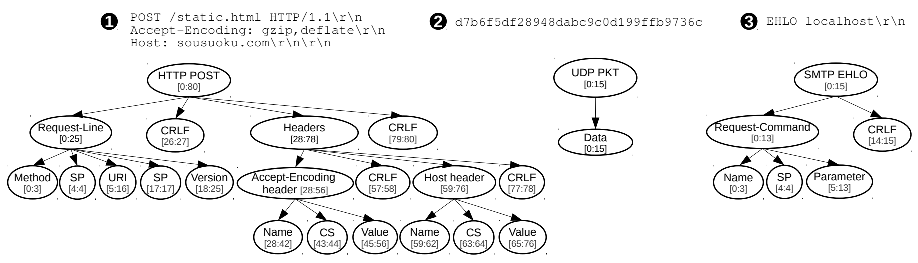
Next, signature generation (Section 5.1) produces an initial set of signatures for each traffic cluster. For each field in the messages in the cluster (or full packets if the protocol is unknown) it tokenizes the field contents, identifying distinctive tokens that cover a significant number of messages in the cluster. It outputs a signature cluster for each traffic cluster, containing one or more signatures, e.g., if there are distinctive tokens that do not appear in all cluster messages.

Signature merging (Section 5.2) identifies signatures across clusters that share tokens in their data fields and merges those signatures and their corresponding clusters. Then, signature clustering (Section 5.3) merges signature clusters generated from traffic produced by the same malware binary, or containing traffic sent to the same endpoint, as these indicate that the clusters belong to the same family. This step produces a smaller set of family clusters but does not modify the signatures.

Signature finalization (Section 5.4) encodes the signatures in the syntax expected by the IDS and removes signatures that create false positives or have little coverage. Optionally, the set of signatures for each family cluster is minimized. Finally, FIRMA outputs the clusters, signatures, and endpoints files.

3 Malware Execution

Executing malware in a contained environment is a widely studied problem [11, 17, 21, 31, 36] and not a contribution of this work. The main goals of malware execution are to incite the malware to produce traffic, to collect a variety of traffic, and to prevent contamination. Inciting the malware to produce network traffic often requires running the same binary multiple times with different configurations. In our environment if a binary fails to produce traffic in the default configuration, it is queued to be rerun on a different VM (e.g., on QEMU if originally run on VMWare) and for an extended period of time (e.g., doubling the execution timer). In addition, all executions replicate some common user actions such as moving the mouse or opening the browser. Malware binaries that do produce network traffic are also rerun when capacity is available to help the signature generation account for non-determinism in the network traffic and to remove artifacts of the environment such as local IP addresses.



```
❶ POST /static.html HTTP/1.1\r\n
  Accept-Encoding: gzip,deflate\r\n
  Host: sousuoku.com\r\n\r\n
❷ d7b6f5df28948dabc9c0d199ffb9736c
❸ EHLO localhost\r\n
```

Fig. 3. Message field tree examples.

Malware binaries may be run with 3 containment policies, designed for different purposes. The default *sink* containment policy sinks all outgoing traffic, except DNS requests and HTTP requests to the top Alexa sites, which are proxied. If the proxy receives no external response it sends a predefined successful response to the malware. The sink policy enables capturing the request sent by the malware even if the remote endpoint is down, so that a signature can still be built for this initial request. The sink policy also avoids remote installations where a malware binary downloads and executes additional components, or malware from other families if involved in the pay-per-install business model [5]. Such remote installations can contaminate the network trace with traffic from different malware families.

The *endpoint failure* policy aborts any outgoing communication from the malware by sending error responses to DNS requests, resets to SYN packets, and sinking outgoing UDP traffic. This policy is designed to trick the malware into revealing all endpoints it knows, as it tries to find a working endpoint. The *restricted access* policy allows the malware limited connection to the Internet, enabling deeper C&C dialogs. To prevent remote installations any connection with a payload larger than 4 KB (the minimum size of a working malware we have observed) is blocked.

The output of a malware execution is a network trace annotated with a unique run identifier, and a database entry stating the configuration for the run such as the malware binary executed, VM software, execution duration, containment policy, and result.

4 Traffic Clustering

Traffic clustering takes as input the network traces from the malware executions and groups together similar traffic. Its goal is to distinguish traffic that corresponds to different network behaviors so that separate signatures can be later built for each. It may produce multiple traffic clusters for the same behavior, which will be later merged during signature merging and clustering. Traffic clustering first extracts a feature vector for each request (Section 4.1). Then, it computes a partition of the feature vectors into traffic clusters. For HTTP, IRC, and SMTP messages it applies a protocol-aware clustering that uses different features for each protocol (Section 4.2). For other traffic, it applies a generic clustering based on transport level information (Section 4.3).

4.1 Feature Extraction

For each request sent by a malware binary FIRMA extracts the following feature vector:

$$\langle fid, rid, proto, msg, sport, dport, sip, dip, dsize, endpoint, ptree\rangle$$

where fid and rid are unique identifiers for the malware binary and malware execution respectively, $proto$ and msg capture the protocol and message type, $sport$, $dport$, sip, dip are the ports and IP addresses, $dsize$ is the size in bytes, and $endpoint$ is the domain name used to resolve the destination IP (dip), or the destination IP if the malware did not use DNS. The $ptree$ feature uniquely identifies the message field tree output by Wireshark[4] when parsing the request.

The protocol feature ($proto$) can have 5 values: HTTP, IRC, SMTP, TCP, and UDP. To identify HTTP, IRC, and SMTP traffic FIRMA uses protocol signatures, which capture protocol keywords present in the early parts of a message (e.g., GET in HTTP or EHLO in SMTP) [10, 15]. Protocol signatures are applied to all TCP connections regardless of the ports used and matching traffic is parsed into messages and fields using the appropriate Wireshark dissector. Note that Wireshark uses the destination port to select the right protocol parser. If FIRMA did not use protocol signatures, Wireshark would not parse standard protocols on non-standard ports and it would try to parse proprietary protocols on standard ports. Both situations are common with malware. For packets not from these 3 protocols, the protocol feature is the transport protocol.

The message feature (msg) is the value of the type field for messages from application protocols (i.e., Method field in HTTP and Command field in IRC and SMTP) and null for TCP and UDP packets. The message field tree feature ($ptree$) captures the hierarchical field structure of a message. Figure 3 shows the trees for 3 different requests: an HTTP POST message, a SMTP EHLO message, and a UDP packet with a single data field because its structure is unknown. The advantage of using Wireshark is that it has dissectors for many protocols, so supporting other application protocols requires only new protocol signatures.

As an optimization, HTTP requests where $endpoint$ is one of the top 200,000 Alexa domains are discarded. Requests to benign domains are often used by malware to check connectivity and signatures derived from them would be discarded later as causing false positives, so avoiding their construction is more efficient.

4.2 Application Protocol Clustering

To group similar HTTP, IRC, and SMTP messages into traffic clusters, FIRMA uses the following set of features for each protocol.

HTTP. FIRMA groups HTTP requests that have the same message type and satisfy one of these conditions:

- **Same URL path.** The path component of the URL is the same in both requests and not empty (i.e., not the root page).
- **Similar URL parameters.** The Jaccard index of the sets of URL parameters (without values) is larger than an experimentally selected threshold of 0.4.

[4] http://www.wireshark.org/

– **Similar header value.** The value of any of the HTTP headers, except the Content-Length and Host headers, is the same in both requests and that value does not appear in the pool of benign HTTP traffic, the User-Agent whitelist or the whitelist of protocol keywords. We exclude the Content-Length because the content of POST requests often has variable length and the Host header because endpoint information is used later in the signature merging step (Section 5.2).

Compared to prior work that clusters HTTP requests [26] our HTTP clustering does not exclusively rely on URL information, but also leverages the HTTP headers. This is important because a number of malware families build custom HTTP requests, which may include distinctive values. For example, the third signature in Figure 1 has a User-Agent header that impersonates the Opera browser. However, the VMs where the malware executes do not have this browser installed and more importantly the real Opera browser includes a minor version in its User-Agent strings. Note that at this step the body of HTTP POST requests is not compared. For efficiency and accuracy these are tokenized and compared during the signature generation and merging steps (Section 5).

IRC. An IRC message comprises an optional prefix, a command, and a list of command parameters. FIRMA groups together messages if they have the same command and the same list of command parameters and the list of command parameters does not appear in the benign traffic pool.

SMTP. An SMTP message comprises a command and a parameter. The SMTP clustering focuses on commands specific to the SMTP engine, rather than to the email messages sent. Currently, it only considers EHLO and HELO messages. FIRMA groups together messages with the same command and parameter value, where the parameter value does not appear in the benign traffic pool.

4.3 Transport Protocol Clustering

For the remaining requests the traffic clustering uses transport features and, similar to HTTP POST requests, the payload comparison is left for the signature generation and merging steps. In particular, it groups packets from the same transport protocol satisfying one of these conditions: same size and sent to the same destination port and endpoint, or same size and Wireshark does not identify their protocol. This differs from application protocols in that the endpoint information is added at this stage because it is unlikely that a malware family will use multiple binary protocols with messages of the same size.

The output of the traffic clustering is the union of the traffic clusters output by the 3 protocol-aware clusterings and the transport protocol clustering. Each traffic cluster contains the feature vectors for the requests in the cluster and the clusters do not overlap.

5 Signatures

This section describes the process of generating an initial set of signatures for each traffic cluster (Section 5.1), merging signatures with similar payload tokens (Section 5.2), clustering signatures by endpoints and file identifiers (Section 5.3), and finalizing the signatures (Section 5.4).

Algorithm 1 Signature Generation Algorithm

```
def generate_signatures($traffic\_cluster$) {
    signatures = []
    full_cov_tokens = []
    # Get unique fields for requests in cluster
    unique_fields = get_distinct_fields($traffic\_cluster$)
    for $field$ in $unique\_fields$
      # Get unique values for field
      unique_values = get_distinct_fields_values($field$)
      # Tokenize unique field values
      tokens = get_tokens(unique_values)
      for $token$ in tokens
        # Get false positives and coverage for token
        [$t_{fp}$,$t_{cov}$] = get_cov_fp($token$)
        # Get requests that contain the token
        $token\_request\_set$ = get_token_requests($token$)
        # Ignore tokens with high false positive or small coverage
        if $t_{fp} > thres_{fp}$ or $t_{cov} < thres_{cov}$
          continue
        # Accumulate tokens with full coverage
        if $t_{cov}$ == 1.0
          full_cov_tokens.append($token$)
          continue
        new_sig = True
        for $sig$ in signatures
          $sig\_request\_set$ = get_signature_requests($sig$)
          # check if for same requests we already have signature
          if $token\_request\_set$ = $sig\_request\_set$
            $sig$.append($token$)
            new_sig = False
            break
        # If new token, add it to signature
        if new_sig or len(signatures) == 0
           sig = new_sig(token)
           signatures.append(sig)
     # Add full coverage tokens to all signatures
     for $sig$ in signatures
       for $full\_cov\_token$ in full_cov_tokens
         $sig$.append($full\_cov\_token$)
  return signatures
}
```

5.1 Signature Generation

For each traffic cluster, signature generation creates a signature cluster comprising a set of signatures and, for each signature, a set of requests (i.e., feature vectors) used to generate it. From the feature vectors of a signature it is straightforward to obtain the set of ports, IPs, endpoints, and malware binaries of the signature.

Algorithm 1 describes the signature generation. Its salient characteristics are that the tokenization is performed on fields and that multiple signatures may be generated for each traffic cluster. For each field in the requests in the traffic cluster, it identifies distinctive tokens i.e., tokens with high coverage and low false positives. We define the *false positive rate* of a token in a field to be the fraction of requests in the benign pool that contain the token in the field, over the total number of requests in the benign pool. We define two coverage metrics. The *request coverage* is the fraction of requests in the traffic cluster with the token in that field, over the total number of requests in the traffic cluster. The *file coverage* is the fraction of malware binaries that have a request in the cluster with the token in that field, over the total number of malware binaries with at least one request in the cluster. A token is distinctive if it has a file coverage larger than 0.4 and a false positive rate below 10^{-9}. The reason to use the file coverage to consider a token distinctive is that we are interested in signatures that match as many binaries as possible from the same family. The request coverage is used by Algorithm 1 for identifying tokens present on the same requests.

Algorithm 1 can generate multiple signatures because distinctive tokens do not need to appear in all requests in the traffic cluster. For example, the requests in a traffic cluster may have been grouped because they all have the same URL path. In addition, 50% of them could have a distinctive User-Agent value and the other 50% a different one. In this case, the signature generation may output two signatures, each with the distinctive URL path and one of the two User-Agent values.

The get_distinct_fields function returns all fields in the tree, except fields that encode integer values, which should not be tokenized (e.g., the Content-Length HTTP header), and fields that contain endpoints (e.g., Host HTTP header) because this information is used in later steps. The tokenize function uses a suffix array [1] to extract tokens larger than 5 bytes that appear in the set of unique field values.

5.2 Signature Merging

Signature merging identifies signatures in different signature clusters with similar tokens. It detects requests with similar content in their data fields, which ended up in different traffic clusters because they were not similar on other fields or transport features. For each pair of signatures from different clusters, it computes the longest common subsequence between each pair of signature tokens. If it finds a common subsequence larger than 7 bytes, it merges the two signatures into one and combines their corresponding clusters. For example, if the signature generation returns the following two signature clusters:

```
SC-153 S1: "|9ad698334c|", |deadbeef5f01000001000000|"
SC-172 S1: "|deadbeef5f01000001000000|"
       S2: "|98760a3d78675d|"
```

the signature merging identifies the common token between the first signature in cluster 153 and the first signature in cluster 172. It merges both signatures, unions their feature vector sets, and combines their clusters. The resulting signature cluster is:

```
SC-(153+172) S1: "|deadbeef5f01000001000000|"
```

5.3 Signature Clustering

Signature clustering identifies signature clusters that correspond to different network behaviors of the same family and merges them into family clusters. For this, it uses the file identifiers and the endpoint information. The intuition for using the file identifiers is that a malware binary belongs to a single family. Thus, if two signatures have been generated from traffic by the same malware binary, those signatures belong to the same family and should be part of the same family cluster. The intuition for using the endpoint information is that C&C servers are specific to a family. Thus, if two signatures have been generated from traffic sent to the same endpoint, they belong to the same family. Note that benign endpoints (e.g., yahoo.com) may be contacted by multiple families but those have been removed in previous steps. Note also that even if the C&C IPs and domains of a family are fully polymorphic (i.e., never reused) the binaries in the family may already have been grouped at prior steps due to other similarities in their traffic.

Signature clustering extracts the set of endpoints and file identifiers for a signature cluster from the feature vectors for each signature. For each pair of signature clusters, if the intersection of their endpoint sets or file identifier sets is not empty, both clusters are merged by doing the union of their signature sets.

5.4 Signature Finalization

This section describes the final steps required to output the signatures.

Signature encoding. The encoding component outputs the signatures in a format suitable for Snort and Suricata. While both use a similar syntax there are some differences between them, e.g., their protocol support. For each signature, it extracts the set of ports from the feature vectors, selects the carrier protocol, adds the family cluster and signature identifiers to the metadata, and for tokens in fields parsed by the IDS, it selects the appropriate modifiers for the content (e.g., http_method, http_header).

Signature validation. The validation component removes signatures that produce false positives or have little coverage. First, it removes signatures with no content tokens and counts how many such signatures it removes from each family cluster as these are highly indicative of fully polymorphic traffic. Then, it runs the signatures using the appropriate IDS on the benign traffic pool and removes signatures that match any traffic since their false positive rate will only increase on live traffic. Then, it runs the remaining signatures on the input network traces, tracking which signatures match traffic from which malware binary. If the file coverage of a signature in its cluster is below 0.4%, the signature is removed since it is too specific and unlikely to match other binaries of the same family.

Signature minimization. The resulting signatures for a family cluster may overlap, i.e., the file coverage of a signature in a cluster may be a superset of the file coverage of another signature in the same cluster. Overlapping signatures provide additional robustness for online monitoring. However, for offline classification with a fixed malware dataset, the analyst may be interested in removing those overlaps for efficiency. If so, FIRMA offers an option for minimizing the signatures for each family cluster, while guaranteeing that all malware binaries in the cluster would be matched by at least one remaining signature. This optional feature is an instance of the optimization version of the set-cover problem where the universe is all malware binaries in the cluster, and the sets correspond to the file coverage of each signature in the cluster. This problem is known to be NP-hard but a greedy algorithm can efficiently approximate it by choosing at each step the set that contains the largest number of uncovered elements [8].

6 Evaluation

This section presents our evaluation of FIRMA. We first describe our datasets (Section 6.1), then we present a quantitative evaluation of the different steps in FIRMA (Section 6.2), and finally we perform a qualitative comparison of the signatures produced by FIRMA with manual ones (Section 6.3).

Dataset	Dates	Binaries	Runs	Requests	HTTP	SMTP	IRC	TCP	UDP
MALICIA	03/2012 - 02/2013	10,600	20,724	495,042	15.9%	1.1%	0%	3.0%	80.0%
MIXED	03/2012 - 04/2012	5,250	10,520	97,559	94.5%	0.7%	0.02%	2.0%	2.8%

Table 1. Datasets used in the evaluation.

6.1 Datasets

To evaluate FIRMA we use two malware datasets [12, 24], summarized in Table 1. Both datasets contain a variety of recent malware and their traffic exhibits common obfuscation techniques such as encryption and polymorphism (in IPs, domains, and payloads). The MALICIA dataset is publicly available and comprises malware binaries collected from drive-by downloads from March 2012 to February 2013 [23, 24]. The malware binaries have been classified into families using static icon information, as well as screenshots and network traffic obtained by executing the binaries. In addition to the public dataset, we have the network signatures manually generated in that project. We use the given classification as ground truth to evaluate the malware clustering produced by FIRMA and compare the signatures automatically generated by FIRMA with the manually generated ones.

The MIXED dataset comprises 10,520 network traces obtained by executing 5,250 binaries. These binaries are a subset of the ones analyzed in [12] and were collected from a variety of feeds that include drive-by downloads, P2P, and spam. We do not have access to the malware binaries themselves but only to the network traces, the mapping from each network trace to the MD5 hash of the binary that produced it, and the containment policy used in the run.

Table 1 shows for each dataset the malware binaries that exhibit network traffic, the malware executions, and the requests sent by the malware, as well as the split of the requests by protocol. The average number of pcaps for each malware is close to 2 in both datasets as some malware binaries are run multiple times with different VM software, execution duration, and containment policies. In the MIXED dataset, HTTP traffic is most common, followed by generic TCP and UDP traffic and smaller amounts of SMTP and IRC traffic. The MALICIA dataset shows a surprisingly large number of UDP requests, which are due to the highly verbose zeroaccess family that produces 87% of the UDP requests. FIRMA does not make assumptions about the input dataset and is not affected by unbalanced family traffic distributions.

6.2 Quantitative Results

Table 2 summarizes the results for each step of our approach. On the left, it shows the number of traffic clusters, the initial number of signatures generated, the number of signature clusters and signatures after signature merging, and the number of signature clusters after signature clustering. On the right, it shows the final results: the number of family clusters and signatures, and the remaining signatures after minimization.

Initially a large number of traffic clusters is produced (2,360 and 976, respectively). Table 3 shows the split of traffic clusters by message type. There is an order of magnitude more HTTP GET traffic clusters than POST ones. This is due to downloaders

Dataset	# Traffic Clusters	# Sig. (initial)	# SCs (merging)	# Sig. (merging)	# SCs (clustering)	Families	Sigs	Sigs (min.)
MALICIA	2,360	1,196	1,699	535	57	57	116	63
MIXED	971	601	884	514	108	108	269	126

Table 2. Summary of results for each step.

Dataset	HTTP			SMTP		IRC			TCP	UDP
	GET	POST	HEAD	EHLO	HELO	NICK	USER	Other		
MALICIA	1,244	47	0	1	0	0	0	0	677	391
MIXED	488	50	1	2	3	6	6	6	127	282

Table 3. Traffic Clustering Results.

that perform GET requests to obtain other executables and where each malware binary randomizes the name of the file to download. Surprisingly, the initial number of signatures (column 3) is smaller than the number of traffic clusters (column 2). This is because some traffic clusters contain traffic that is not different enough from benign traffic. When parsed into fields, the field values are common in the benign traffic pool and no signature can be produced. The number of signature clusters and signatures reduces after merging (columns 4 and 5) because signatures with common tokens in their payloads are merged and their signature clusters combined.

The final numbers show that even if the number of initial traffic clusters is large, the subsequent steps are able to group the malware binaries into a small number of families (57 and 108 respectively). On average FIRMA produces 2.3 signatures for each family cluster, each capturing a different network behavior. This shows the prevalence of malware families with multiple network behaviors and demonstrates the importance of building a signature for each behavior. Note that if we had not built separate signatures for each behavior we would have been left instead with very general signatures with large false positives. If the optional minimization is applied, the number of signatures per family reduces to 1.1 because for many families each malware binary exhibits all network behaviors so the signatures for each behavior overlap.

Table 4 shows the distribution of the generated signatures by protocol. The largest number of signatures is for HTTP (55%–80%), but there is a large number of signatures for other network behaviors (20%–45%). There are 11 families (34%) in the MALICIA dataset for which no HTTP signature is generated. These families cannot be detected by prior tools that focus exclusively on the HTTP traffic [26]. In addition, for 14 other families (44%) their HTTP signatures contain tokens outside the HTTP method and URL (e.g., in headers or the body), which makes our HTTP signatures more specific than the ones generated by Perdisci et al.

Malware clustering accuracy. To evaluate how accurately FIRMA groups malware binaries into families we use the classification for each binary in the MALICIA dataset produced in [24]. Note that we only use the labels after FIRMA has output the results. They are not used during FIRMA's processing but only for quantifying precision and recall of the output clustering. Table 5 shows the precision, recall, and F-Measure for

Dataset	Total	TCP	UDP	HTTP	SMTP	IRC
MALICIA	116	18.1% (21)	0.9% (1)	80.1% (93)	0.9% (1)	0% (0)
MIXED	269	11.5% (31)	20.4% (55)	55.8% (150)	5.6% (15)	6.7% (18)

Table 4. Distribution of generated signatures by protocol.

	Traffic Clustering			Family Clustering		
Dataset	Precision	Recall	F-Measure	Precision	Recall	F-Measure
MALICIA	100%	84.1%	91.3%	100%	97.7%	98.8%

Table 5. Accuracy of the initial traffic clustering and the final family clustering.

Traffic Rate	Time Period	# Alarms	# Alarm Sig.	FPR
359 pps	5.5 days	21	2	10^{-7} (0.00001%)

Table 6. False positive analysis on live traffic.

both the initial traffic clustering and the final family clustering. The results shows very high precision in both clusterings and how the recall significantly improves after the signature merging and clustering steps. In the final malware clustering FIRMA achieves perfect precision and a very high 97.7% recall, with a F-Measure of 98.8%. These results indicate the accuracy of FIRMA when classifying a large number of unlabeled malware binaries with highly varied network traffic.

We examine the results to understand which families are split into multiple family clusters. The ground truth for the MALICIA dataset has 32 families and FIRMA finds an extra family that was missed in [24]. Four families are split into multiple family clusters by FIRMA. Zbot is split into 21 clusters. Being a malware kit, each malware owner configures the kit with a different set of C&C servers and a different key to encrypt its C&C traffic. FIRMA groups the malware binaries into multiple family clusters, likely corresponding to different operations. There is also an unknown family (CLUSTER:B) that splits into 3 family clusters. This is the only family for which FIRMA cannot generate any signature. An initial signature was generated for a 7 byte fully polymorphic packet but it was thrown away during validation because it created false positives. Malware binaries in this family are grouped only based on overlap in endpoint information.

False positive analysis. The generated signatures can produce false positives on live traffic if the benign traffic pool does not accurately or extensively represent the traffic of the monitored network. To measure the false positive rate, we deploy the signatures on a Snort IDS at the border of our lab's network for 5.5 days. This network comprises over 100 hosts, with a variety of operating systems, although being a research lab, only a small number of the hosts run Windows. Table 6 summarizes the results. The IDS sees a traffic rate of 359 packets per second (pps). Only two signatures were matched for a total of 21 alarms. To distinguish between true and false positives, we manually inspect the packets causing the alarms (logged by Snort when a signature triggers). One signature (18 alarms) corresponds to a SSLv2 handshake, which a malware family uses for C&C. Unfortunately, our benign traffic pool did not contain instances of SSLv2. We consider these 18 alarms false positives. The other signature (3 alarms) matches a

Dataset	Feature Extraction	Traffic Clustering	# Sig. (initial)	#SCs (merging)	# SCs (clustering)	Total Time
MALICIA	41m40s	78.0s	37.4s	1.8s	42.3s	44m9s
MIXED	19m8s	17.0s	35.0s	6.4s	0.1s	20m6s

Table 7. Runtime for each step in FIRMA.

`EHLO localhost` SMTP command by a spam bot. The 3 alarms were on incoming SMTP traffic to 3 of our servers from a single host located in China. One of those 3 servers is not used as an email server and has likely been identified through scanning. The logs of our two email servers show that the SMTP exchange did not send valid email. We believe these 3 alarms are true positives. Overall, the false positive rate of the generated signatures is 0.00001%. The low number of true positives is likely due to few Windows hosts in our network and to our signatures covering only malware in our datasets, excluding older malware that periodically scans networks (e.g., Conficker).

Performance. Table 7 shows the runtime of each step. We run FIRMA on a 32-bit 3.3 GHz host with 4 cores and 4GB RAM. The total runtime is 20 and 44 minutes for the MIXED and MALICIA datasets respectively. The most expensive step is extracting the features from the network traces, which also includes obtaining the message field trees from Wireshark. This step is IO bound and accounts for 94% of the runtime. All other steps are completed in less than 2.5 minutes in both datasets. As the feature extraction time is linear on the number of input network traces and their size, FIRMA scales well to larger datasets. A paper and pencil comparison with the runtime results by Perdisci et al. [26] (after adjusting for different number of cores and processor frequency) shows that FIRMA is up to 90 times faster (4.5 times if we include the feature extraction, which [26] does not report).

6.3 Qualitative Results

To assess the quality of the network signatures generated by FIRMA, we compare them with the signatures manually generated in [24]. Of the 32 families in the MALICIA dataset, for 11 FIRMA generates more signatures than the manual analysts, for 21 FIRMA generates the same number, and for 1 less. For 10 of the 11 families where FIRMA generates more signatures, FIRMA captures new network behaviors that were missed by the malware analysts. Some of the new signatures have better file coverage than the manual ones. For example, for the winwebsec fake antivirus each of the two manual signatures covers part of the winwebsec files, but one of FIRMA's signatures covers all of them. For the other family the manual signature captures similarity in a parameter value, which FIRMA currently does not support. However, FIRMA finds that all requests for the family have a common User-Agent value, missed by the analysts.

The family for which FIRMA generates less signatures has 3 manual signatures. The only signature generated by FIRMA matches exactly one of the manual ones. The other manual signatures match one binary each. This is an instance of the manual analysts selecting a behavior specific to a variant that does not generalize to others. We manually classify the 20 families with the same number of signatures into 3 groups:

```
M alert tcp any any -> any [80] (msg:"Cluster:1"; content:"/picture.php";)
A alert tcp any any -> any [80] (msg:"Cluster:1"; content:"GET";
    http_method; content:"/picture.php"; http_uri;)

M alert tcp any any -> any [80] (msg:"Cluster:2"; content:"POST";
    http_method; content:"pcre:"/aa1020R0=[^&]+%2/";)
A alert tcp any any -> any [80] (msg:"Cluster:2"; content:"POST";
    http_method; content:"aa1020R0="; content:"|253344253044253041|";)

M alert tcp any any -> any [80] (msg:"Cluster:3"; content:"GET";
    http_method; content:"/n09230945.asp"; http_uri;)
A alert tcp any any -> any [42633] (msg:"Cluster:3"; dsize:5;
    content:"|6e65770d0a|";)

M alert tcp any any -> any any (msg:"Cluster:4";
    content:"|04000001050000000007000100|";)
A alert tcp any any -> any [443,8014] (msg:"Cluster:4"; dsize:13;
    content:"|04000001050000000007000100|";)
A alert tcp any any -> any [9145] (msg:"Cluster:4"; dsize:181;
    content:"GNUTELLA CONNECT/0.6|0d0a|Listen-IP|3a|0.0.0.0|3a|22324|0d0a|
    Remote-IP|3a| 31.35.6.6|0d0a|User-Agent|3a| Shareaza|0d0a|";)
```

Fig. 4. Comparison of signatures manually (**M**) and automatically (**A**) generated by FIRMA. For simplicity, metadata has been removed and family names have been normalized.

for 12 FIRMA generates signatures that are more specific, 5 have identical signatures, and for 3 the manual signatures are more specific. In general FIRMA produces more specific signatures because the analysts tend to stop adding tokens when they feel the signature is specific enough. There are two cases where the manual signatures are more specific. For URL parameters, FIRMA generates a token set while the manual signatures sometimes use a regular expression, which imposes an ordering constraint and may limit the size of parameter values. In addition, some manual signatures capture the lack of HTTP headers. For example, one of the cleaman signatures captures that the requests from this family do not have a User-Agent or Accept header, which typically appear in HTTP requests. While the automatically generated signatures are still specific enough without these two features, we plan to support them in the future.

Figure 4 compares some manually generated signatures (M) with the corresponding ones generated by FIRMA (A). The signatures for Cluster 1 are very similar, but the manual one misses the GET method and the field attributes. This illustrates inconsistencies in manual signatures that FIRMA prevents. In Cluster 2 the manual signature uses a regular expression, but the equivalent signature by FIRMA is more specific and faster to match since it uses no regular expression. The manual signature for Cluster 3 captures traffic to the whatismyip.com web service that the malware uses for checking its public IP address. This signature can produce false positives in live traffic. Instead, the signature by FIRMA captures a 5-byte binary packet on port 42633, missed by the analysts. Finally, Cluster 4 shows one of the families for which FIRMA finds an extra signature that captures a new network behavior (Gnutella P2P traffic).

Overall, our qualitative evaluation shows that the signatures generated by FIRMA are of similar, and often better, quality than the ones we manually generated. Of course, more experienced analysts would generate better manual signatures. However, FIRMA provides a combined solution to the problems of malware clustering and network signature generation that significantly reduces the amount of effort required of analysts.

To facilitate external review of our signatures and enable other groups to compare their results to ours, we plan to release a new version of the MALICIA dataset that adds the manually generated signatures and the ones produced by FIRMA.

7 Related Work

A number of prior works propose systems to automatically generate different types of network signatures to identify worm traffic. Honeycomb [20], Autograph [18], and EarlyBird [32] propose signatures comprising a single contiguous string (i.e., token). Polygraph [25] proposes more expressive token set, token subsequence, and probabilistic Bayes signatures. Wang et al. extend PAYL [37] to generate token subsequence signatures for content common to ingress and egress traffic. Nemean [41] introduces semantics-aware signatures and Hamsa [22] generates token set signatures that can handle some noise in the input traffic pool. Beyond worms, Botzilla [29] generates signatures for the traffic produced by a malware binary run multiple times in a controlled environment. All these works assume a single malware family or small amounts of noise in the input traffic. In contrast, FIRMA handles input traffic from many malware families with multiple network behaviors.

Recently, ProVex [30] proposes signatures to detect fully polymorphic C&C traffic given the decryption function and keys used by the malware, which can be extracted with binary analysis [6]. FIRMA can be used to quickly identify such traffic but cannot generate signatures for it. Also related are AutoRE [40], which builds URL regular expression signatures from emails to identify spam botnets and ShieldGen [9], which produces protocol-aware network signatures for vulnerabilities. Wurzinger et al. [38] detect comprised hosts by monitoring the reaction from a host to a received command using network signatures. Compared to FIRMA they do not address how to cluster traffic from different malware binaries. The signatures produced by FIRMA are matched by off-the-self IDSes and techniques to improve the efficiency of signature matching are also related [35].

There has also been extensive work on behavioral classification techniques for malware using a variety of features such as system calls, system changes, network traffic, and screenshots [3–5, 12, 13, 26, 28]. Most related to FIRMA are techniques that focus on network traffic. Botminer [13] clusters similar bot traffic for building detection profiles but does not generate network signatures. Perdisci et al. [26] cluster malware that uses HTTP traffic using sending profiles and features on the HTTP method and URL. They also build token subsequence signatures that cover the request method and the URL. In contrast, FIRMA clusters malware using all traffic it produces. For HTTP traffic, in addition to the method and the URL FIRMA also analyzes the content of the headers and the body and includes them in the signatures. Also related to our work are techniques to reduce the dimensionality in malware clustering [16] and proposals to evaluate malware clustering results using AV labels [27].

8 Conclusion

We have presented FIRMA, a tool that given a large pool of network traffic obtained by executing unlabeled malware binaries, generates a clustering of the malware binaries into families and a set of of network signatures for each family. FIRMA produces network signatures for each of the network behaviors of a family, regardless of the type of traffic the malware uses. It efficiently generates protocol-aware token-set signatures, which capture distinguishing characteristics in any of the fields of the requests. We have implemented FIRMA and evaluated it on two recent datasets comprising nearly 16,000 unique malware binaries. Our results show that the clustering produced by FIRMA has very high precision and recall. We compare FIRMA's signatures with manually generated ones, showing that they are of similar quality (often better), while offering large savings in analyst resources.

Acknowledgements. We would like to thank James Newsome for providing us with the code for Polygraph and Antonio Nappa for his help with the MALICIA dataset. We are also thankful to Corrado Leita and the anonymous reviewers for their insightful comments. This work was supported in part by the European Union through Grant FP7-ICT No.256980, by the Spanish Government through Grant TIN2012-39391-C04-01, and a Juan de la Cierva Fellowship for Juan Caballero. Opinions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

1. M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1), 2004.
2. Anubis: Analyzing unknown binaries. `http://anubis.iseclab.org/`.
3. M. Bailey, J. Oberheide, J. Andersen, Z. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of internet malware. In *raid*, 2007.
4. U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, behavior-based malware clustering. In *ndss*, 2009.
5. J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: The commoditization of malware distribution. In *usenixsecurity*, 2011.
6. J. Caballero, N. M. Johnson, S. McCamant, and D. Song. Binary code extraction and interface identification for security applications. In *ndss*, 2010.
7. J. Caballero, H. Yin, Z. Liang, and D. Song. Polyglot: Automatic extraction of protocol message format using dynamic binary analysis. In *ccs*, 2007.
8. V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3), 1979.
9. W. Cui, M. Peinado, H. J. Wang, and M. Locasto. shieldgen: Automatic data patch generation for unknown vulnerabilities with informed probing. In *oakland*, 2007.
10. H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. Sommer. Dynamic application-layer protocol analysis for network intrusion detection. In *usenixsecurity*, 2006.
11. M. Graziano, C. Leita, and D. Balzarotti. Towards network containment in malware analysis systems. In *acsac*, 2012.
12. C. Grier et al. Manufacturing compromise: The emergence of exploit-as-a-service. In *ccs*, 2012.
13. G. Gu, R. Perdisci, J. Zhang, and W. Lee. Botminer: Clustering analysis of network traffic for protocol and structure independent botnet detection. In *usenixsecurity*, 2008.
14. F. Guo, P. Ferrie, and T.-C. Chiueh. A study of the packer problem and its solutions. In *raid*, 2008.
15. P. Haffner, S. Sen, O. Spatscheck, and D. Wang. acas: Automated construction of application signatures. In *minenet*, 2005.
16. J. Jang, D. Brumley, and S. Venkataraman. Bitshred: Feature hashing malware for scalable triage and semantic analysis. In *ccs*, 2011.

17. J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy. Studying spamming botnets using botlab. In *nsdi*, 2009.
18. H.-A. Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In *usenixsecurity*, 2004.
19. E. Kirda, C. Kruegel, G. Banks, G. Vigna, and R. A. Kemmerer. Behavior-based spyware detection. In *usenixsecurity*, 2006.
20. C. Kreibich and J. Crowcroft. Honeycomb - creating intrusion detection signatures using honeypots. In *hotnets*, 2003.
21. C. Kreibich, N. Weaver, C. Kanich, W. Cui, and V. Paxson. gq: Practical containment for measuring modern malware systems. In *imc*, 2011.
22. Z. Li, M. Sanghi, B. Chavez, Y. Chen, and M.-Y. Kao. Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilience. In *oakland*, 2006.
23. The malicia project. `http://malicia-project.com/`.
24. A. Nappa, M. Z. Rafique, and J. Caballero. Driving in the cloud: An analysis of drive-by download operations and abuse reporting. In *dimva*, 2013.
25. J. Newsome, B. Karp, and D. Song. Polygraph: Automatically generating signatures for polymorphic worms. In *oakland*, 2005.
26. R. Perdisci, W. Lee, and N. Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *nsdi*, 2010.
27. R. Perdisci and M. U. Vamo: Towards a fully automated malware clustering validity analysis. In *acsac*, 2012.
28. K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov. Learning and classification of malware behavior. In *dimva*, 2008.
29. K. Rieck, G. Schwenk, T. Limmer, T. Holz, and P. Laskov. Botzilla: Detecting the phoning home of malicious software. In *ACM Symposium on Applied Computing*, 2010.
30. C. Rossow and C. J. Dietrich. Provex: Detecting botnets with encrypted command and control channels. In *dimva*, 2013.
31. C. Rossow, C. J. Dietrich, H. Bos, L. Cavallaro, M. van Steen, F. C. Freiling, and N. Pohlmann. Sandnet: Network traffic analysis of malicious software. In *badgers*, 2011.
32. S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *osdi*, 2004.
33. Snort. `http://www.snort.org/`.
34. Suricata. `http://suricata-ids.org/`.
35. N. Tuck, T. Sherwood, B. Calder, and G. Varghese. Deterministic memory-efficient string matching algorithms for intrusion detection. In *infocom*, 2004.
36. M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, and S. Savage. Scalability, fidelity, and containment in the potemkin virtual honeyfarm. In *sosp*, 2005.
37. K. Wang, G. Cretu, and S. J. Stolfo. Anomalous payload-based worm detection and signature generation. In *raid*, 2005.
38. P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda. Automatically generating models for botnet detection. In *esorics*, 2009.
39. J. Wyke. The zeroaccess botnet, 2012. `http://www.sophos.com/en-us/why-sophos/our-people/technical-papers/zeroaccess-botnet.aspx`.
40. Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming botnets: Signatures and characteristics. In *sigcomm*, 2008.
41. V. Yegneswaran, J. T. Giffin, P. Barford, and S. Jha. An architecture for generating semantics-aware signatures. In *usenixsecurity*, 2005.
42. H. Yin, D. Song, E. Manuel, C. Kruegel, and E. Kirda. Panorama: Capturing system-wide information flow for malware detection and analysis. In *ccs*, 2007.