TRAPPING UNKNOWN MALWARE IN A CONTEXT WEB

Numaan Huq & Peter Szabo
Sophos, 580 Granville Street, Vancouver, BC V6C1W6, Canada

Email {Numaan.Huq, Peter.Szabo}@sophos.com

ABSTRACT

The escalation of web-based threats has triggered the adoption of URL blocklists and web-object scanning techniques in anti-virus products. Often these techniques are employed in isolation, applying traditional on-access/on-demand scans for downloaded web content and blocklists for URLs, with little to no context sharing between the two layers. In this paper, we demonstrate combining URL information, e.g. keywords, patterns, paths, etc., with file properties to create web-context detections (WCD). WCD targets malware from web sources for which we have no file detections or URL reputation information.

We discuss how WCD:

- Proactively detects files based on web source. Scanning in context of a web source allows the creation of broader and more effective detections.
- Is effective against threats such as zero-days, malware, C&C traffic, etc.
- Uses unstructured patterns such as keywords, file properties, paths, etc. and is more resilient to file and URL morphing than traditional pattern/sequential evaluations.
- Performs better than traditional URL blocklists in cases like compromised sites.
- Resists field testing by malware authors.

Using almost a year's worth of attack data, we describe some WCD strategies for popular threats like zero-days, compromised sites and exploit kits. In conclusion, we demonstrate that WCD can be used to plug the detection gap between file detections and URL blocklists.

INTRODUCTION

In today's world, the web is the d*e facto* malware propagation channel [1]. Millions of malicious files get delivered via attack vectors such as spam, drive-by-downloads, compromised sites, botnets, etc. Long gone are the days when the attackers were enthusiasts and script kiddies. They have been replaced by well-organized and well-funded criminal organizations that are out to steal money and information.

Attackers have extended their reach to newer platforms such as social networks, cloud services and mobile devices in addition to the conventional platforms like personal computers and servers [2]. Attackers respond quickly to newly discovered vulnerabilities and leverage zero-day exploits using various exploit kits. Every day, attacks on poorly configured websites and databases continue to steal information and deliver malware; ransomware and FakeAV continue to successfully be used in social engineering attacks.

The escalation of web-based threats triggered the adoption of URL blocklists/reputation sets and web-object scanning techniques in anti-virus products. Often these techniques are employed in isolation, applying traditional on-access/on-demand scans for downloaded web content and blocklists/reputation for URLs, with little to no context sharing between the two layers. By combining URL information such as keywords, patterns, paths, etc. with file properties, we created web-context detections (WCD). WCD targets malware from web sources for which we have no file detections or URL reputation information.

In this paper, we provide an architectural overview of our WCD framework. We discuss our set-up and the sources used for gathering telemetry and reporting data. We demonstrate how we created WCD identities using unstructured detection elements such as keywords, paths and file properties, and show how such detection logic makes them more resilient to file and URL morphing compared to traditional pattern/sequential evaluations. The downloaded context and URL inspection also allows us to incorporate the social engineering aspects of the threat into the detection logic – something that traditional pattern/sequential evaluations cannot achieve. Using WCD identities, we successfully blocked known families of malware as well as unknown malware for which generic detections were lacking. Our WCD identities also performed better than traditional URL blocklists in detecting compromised sites hosting malware. We present case studies on four published WCD identities showcasing the different detection strategies, provide comparative statistics and make observations based on our findings.

Because WCD identities run in a web context, a malicious file uploaded by an attacker to a file scanning service will result in apparent misses. This is because the scanners are missing the required web context, limiting the attacker's field-testing capabilities. We also discuss our release procedures for WCD identities designed to mitigate risks. In our experiments with WCD, we restricted ourselves to detecting PE files only but they can easily be extended to include any file format scanned by the anti-virus scanner, such as PDF, DOC, SWF, etc.

At the conclusion of this paper, we aim to demonstrate that WCD can be used to increase the overall endpoint protection coverage.

FRAMEWORK ARCHITECTURE

In our experiment, we restricted ourselves to downloaded PE files only, but the framework can easily be extended to support other file formats scanned by the anti-virus scanner, e.g. PDF, SWF, DOC, HTML. The implementation for other file formats will be similar to that for PE files. The framework used for creating WCD identities is shown in Figure 1.

A web application such as a browser fetches a URL pointing to a PE file. An intermediate network layer such as Layered Service Provider (LSP), Firewall, Proxy, etc. intercepts the fetched URL and associated PE file. This interception layer communicates with an anti-virus scanner installed on the client computer and passes the URL and PE file to the scanner for further processing.

Prior to the downloaded PE file being made available to the user, the anti-virus scanner extracts various properties such as version information, PE headers, resources, packer information, code and emulated code. The anti-virus scanner

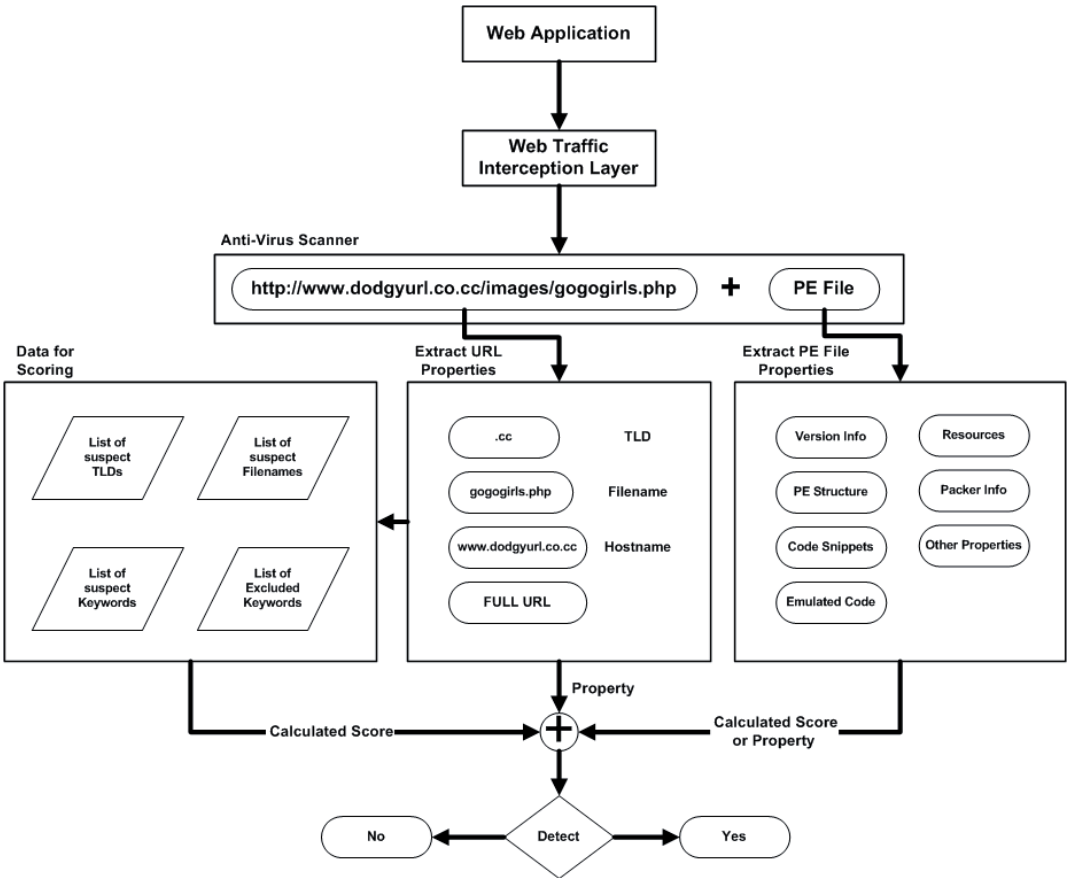

Figure 1: WCD framework architecture.

also parses the associated URL and similarly extracts URL-based properties: top-level domain (TLD), filename, hostname and full URL. It additionally applies stored lists of data to the extracted URL properties and calculates scores. The lists used were:

- Suspect TLDs – these are TLDs commonly abused by malware, e.g. .cc, .tv, .xxx. This list is applied to the extracted TLD.
- Suspect filename keywords – these filenames and keywords are commonly used in social engineering attacks, e.g. facebook, adobe.exe, video.scr. This list is applied to the extracted filename.
- Suspect keywords – these keywords are either used for identifying social engineering or they indicate compromised hosts, e.g. 'voyeur', 'medication', '/wp-content/', and are applied either to the hostname or to the full URL.
- Excluded keywords – these keywords or hostnames are used for averting FPs, e.g. .microsoft.com, .vmware.com. This keyword list was compiled from URL reputation sets and is applied either to the hostname or to the full URL.

A WCD identity uses combinations of these PE and URL properties/scores in its detection logic to determine whether the URL + PE file being fetched is likely to be malicious.

SET-UP AND DATA SOURCES

In our set-up, the web traffic interception layer was implemented using a custom LSP for all browser traffic on the endpoint and using the *Sophos Web Appliance* (acting as web proxy) for all traffic. These products support web-content inspection, of which we utilize HTTP, HTTPS, FTP, and FTPS. We used the *Sophos Anti-Virus* scanner for extracting properties and detecting files. WCD reporting identities were released as CXwebs [3]. Data was collected using the cloud from *Sophos* customers – primarily business users – with *Sophos Live Protection* enabled between November 2012 and April 2013. *SophosLabs* and third-party URL feeds were used as a source of malicious URLs.

URLS TELL TALES

URLs are an important component of a social engineering attack. The attacker's goal is to make URL links appear legitimate or interesting enough to tempt the victims into clicking the link. When a URL is passed to the anti-virus scanner, the following properties are extracted: TLD, filename, hostname and full URL. These properties are then made available for querying by WCD identities. In example URL http://www.dodgyurl.co.cc/images/gogogirls.php, the extracted properties are:

- TLD = .cc
- Filename = gogogirls.php
- Hostname = www.dodgyurl.co.cc
- Full URL = http://www.dodgyurl.co.cc/images/gogogirls.php

(Note: one frequently seen social engineering trick is to obscure an executable by using a benign extension such as

.php or .jpg.) When analysing known malicious URLs, we found that particular TLDs are strongly associated with malware repositories. Examples of suspect TLDs include:

- .cc – Cocos Islands
- .xxx – porn domain
- .tv – Tuvalu.

The *Malware Patrol* site (www.malware.com.br) contains a comprehensive list of suspect TLDs [4]. A PE file hosted by one of these TLDs is not necessarily malicious, but is treated as suspicious. A score is assigned to the URL when a suspect TLD is present; the score is determined by how likely it is, based on observed data, that a given TLD hosts malware. Common TLDs like .com or .org would not get a score because no reasonable conclusion can be drawn about the content being hosted there solely based on the TLD.

We similarly analysed filenames from known malicious URLs for suspicious keywords. We do case-insensitive pattern matching and support substring matches. Examples of suspect filename keywords include:

- adobe – claims to be an *Adobe* executable
- java.exe – claims to be a Java executable
- xamateurs – pornographic websites
- avi., bmp., pdf. – possible double extensions
- hinklescreativeimages.com/invoice.pdf.exe – example malicious URL claiming to be a PDF invoice. Filename keywords: invoice, .pdf.exe.

Examples of suspect keywords found in malicious URLs (both hostname and full URL) include:

- password – possibly warez
- bikini – possibly clothing or mildly/strongly sexually suggestive
- credit – possibly financial in nature
- viagra – possibly pharmaceutical in nature
- windowsupdaters – possibly fake OS update
- fedex-online-pharmacies.blog.hr/. – example malicious pharmacy URL. Keywords: pharma, fedex. .hr. (Note: pharma is a substring of pharmacy and pharmacies.)

Each keyword is assigned a score which is determined by how likely it is, based on observed data, for a given keyword to be associated with malware. We return an aggregate score for each of the filename, hostname and full URL.

The returned scores provide an approximate measure of the maliciousness of the URL. Using the scores in identities is optional but provides a good filter for further processing. If the returned scores are sufficiently high or the URL contains known bad patterns, then the URL + PE file can be blocked based on URL alone. Generally we combine the URL information with the PE file properties before deciding if the URL + PE file is malicious.

MEASURING MALICIOUSNESS

In this section we introduce two techniques we developed for quickly measuring the purported maliciousness of any PE file. These techniques were primarily used for filtering suspect files but were also used in detection logic.

Malicious index

The anti-virus scanner extracts PE file properties such as:

- Version information – e.g. company name, filename, internal name.
- PE structure – e.g. sizes, section attributes, section names.
- Resources – e.g. exports, imports, icons.
- Code snippets – e.g. malicious code, clean code, packer code.
- Emulated code – depends on the anti-virus scanner's emulator.

(Note: version information is a subset of resources but we treat it independently.) Suspicious PE properties – either properties that are commonly seen in malware or some that are defined by analysts – are grouped together and used to calculate a Malicious Index (MI). For example, we can calculate the Version_MI by combining the properties:

```
Version_MI = agg_PE_properties(SUSPICIOUS_
TIMESTAMP,DUPLICATE_NAME,FILE_VERSION_
1.0.0.0,PRODUCT_VERSION_1.0.0.0,ONE_WORD_LEGAL_
DESCRIPTION,RANDOM_STRINGS_IN_VERSION,NO_VERSION_
INFORMATION);
```

In our implementation, the existence of one of these properties adds a score of 1 to the Version_MI. For our example Version_MI formula, the following file (SHA1: ed55 6e03405ef4916dba49713993a47220164801) will trigger properties FILE_VERSION_1.0.0.0, PRODUCT_VERSION_ 1.0.0.0, and DUPLICATE_NAME:

```
File Description:
Legal Copyright:     Copyright 2011
Assembly Version:    1.0.0.0
File Version:        1.0.0.0
Product Version:     1.0.0.0
Original Filename:   Installer.exe
Internal Name:       Installer.exe
```

We get a Version_MI score of 3 for this file. A clean file such as NOTEPAD.EXE (SHA1: ed55ad0a7078651857bd8fc0eed d8b07f94594cc) will trigger none of the listed properties and would have a Version_MI score of 0:

```
File Description:  Notepad
Legal Copyright:   Microsoft Corporation. All
                   rights reserved.
File Version:      5.1.2600.2180
                   (xpsp_sp2_rtm.040803-2158
Company Name:      Microsoft Corporation
Product Version:   5.1.2600.2180
Product Name:      Microsoft Windows Operating
                   System
Original Filename: NOTEPAD.EXE
Internal Name:     Notepad
```

For WCD identities, we created mutually exclusive indices for:

- Code section properties
- PE structure properties
- Resource properties
- Version information properties
- Packer properties.

Based on our testing, we have found that a low threshold score for these indices will detect large volumes in both our

clean and malware collections. Conversely, a high threshold score detects files in our malware set while triggering only on a small set of clean files, but the overall effectiveness is greatly reduced. The challenge was to find an optimum score for each MI where we detected the highest number of malicious files and smallest number of clean files.

Odd* properties

In our experiments, we also derived a set of properties from PE files that we call Odd* (read odd-star) for filtering out potentially clean files and targeting suspicious files. We created exclusion lists for features that are expected in clean files; everything else is considered odd. Our current Odd* feature list consists of:

- Odd first instruction
- Odd first API
- Odd section name
- Odd appended data.

It is assumed that a single Odd* property will never be a 100% malicious/clean differentiator because there will always be new compilers, new section names etc. that will fall outside of the exclusion. There was significant effort at the onset to collect data for the exclusion lists to bring the Odd* properties up to a level of maturity where we could confidently incorporate them in detections. We conclude that if enough of these Odd* properties exist in a file, a well-chosen threshold can predict, with a reasonable degree of confidence, whether the file is suspicious or not.

BUILDING WEB-CONTEXT IDENTITIES

WCD identities are 'blocking' by design. For example, if a WCD identity triggers on a downloaded PE file, the file will be blocked as malicious. The user does not have access to the malicious PE file and the file is not executed on the system. This blocking nature means we don't need to include malware clean-up code in WCD identities.

We use a three-pronged strategy in developing WCD identities:

1. PE file only detection: the PE file has distinct properties that the WCD identity incorporates to create flexible detection logic as opposed to the comparatively stricter detection logic present in generic identities for the malware family. By itself the flexible detection logic may not be strong enough for a traditional file block, but combined with the downloaded context it creates a strong detection.
2. URL and PE file combination detection: this is the ideal WCD identity development strategy where we combine both URL and PE file properties. In this strategy, we may use PE file properties such as MI and Odd*, described in the previous section, together with URL properties to decide whether the PE file is malicious or not.
3. URL only detection: the URL has distinct/known properties that are strong enough to trigger an outright block without the PE file requiring inspection. These properties may include known bad keywords, bad path fragments, bad filename, etc.

One of the advantages of using unstructured properties in WCD identities is that they make the identity resilient to file and URL morphing. We demonstrate this using an example FakeAV detection written in pseudo-code, as shown in Listing 1.

In this example, we aggregate ten PE file properties associated with the target FakeAV family. We require at least FAKEAV_PE_THRESHOLD (value=6) of these properties, in any combination, to be present for the PE properties test to return TRUE. Two downloaded PE files belonging to this FakeAV family have six and eight of the listed properties respectively; this strongly implies morphing. The PE properties test will return TRUE on the morphed files as long as they meet the minimum threshold.

This strategy is also applied to URL properties. Our FakeAV WCD identity is detecting filenames that contain at least two of the three keywords 'anti', 'virus' or 'spyware'. The URL properties test returns TRUE for filenames: 'Anti-Virus 2013.exe', '2013_spyware_virus_removal.exe' and 'Platinum Anti Spyware 2014.exe'. This combined with the PE file properties test creates a flexible and yet strong detection.

The fact that the WCD identities utilize both file and URL properties, makes it difficult for attackers to field test their malware without going to great lengths in recreating the web

```
Name:WCD/FakeAV
{
     #define FAKEAV_PE_THRESHOLD 6
     #define FILENAME_THRESHOLD 2

     // Check that the detection is firing in context of web
     if(get_context()!="web") { return(FALSE); }

     // Check that the filetype is PE
     if(get_filetype()!="PE") { return(FALSE); }

     // Aggregate the URL properties
     if(agg_URL_filename_properties("anti","virus","spyware") < FILENAME_THRESHOLD)
     {
         return(FALSE); }

     // Aggregate PE file properties
     if(agg_PE_properties(P1,P2,P3,P4,P5,P6,P7,P8,P9,P10) < FAKEAV_PE_THRESHOLD) {
         return(FALSE); }

     // Check exclusions
     if(check_exclusion(url,file)!=0){ return(FALSE); }

     // Detect the file
     return(TRUE);
}
```

Listing 1: Example FakeAV detection.

context. A malicious file uploaded by an attacker to a file-scanning service may result in apparent WCD misses. This is because their scanners are missing the required web context needed to trigger the WCD identities and is not indicative of the WCD's ability to block the PE file. If an attacker does figure out the required set-up and create a context scanner, the unstructured property-based detection logic makes it difficult for them to reverse what properties need manipulation in order to defeat the detection.

EVERYDAY EFFECTIVENESS

We collected data for PE files downloaded via a browser between November 2012 and April 2013. This data was used in two ways:

- Exclusion – high download volume data strongly implies popular software being downloaded; we verified and used this data in building exclusion lists. We postulated that finding malware in this group would be highly unlikely as 'popular' malware would very quickly be detected by anti-virus products and thus cut down on the volume.
- Detection – low download volume data was where we found most of our WCD target samples (note: low download volume does not imply that a PE file is suspicious or malicious). For example, we found a ZAccess variant, SHA1:0cf0656bf720ac451ae5f197679ef53413f7353c, being hosted on the site: 'xpornstarsagq.dnset' (partial domain provided). We observed only 13 downloads from this host to seven unique IPs. This undetected variant required an update to an existing generic identity; a WCD identity would have blocked the download of this file and prevented a possible infection.

Over the same period, we collected data for WCD identity hits on browser-downloaded PE files. In raw numbers:

- total blocks by WCD identities: 69,178
- unique SHAs blocked by WCD identities: 12,519
- unique SHAs we managed to collect for further inspection: 7,275 (58.11% of 12,519).

Generic identities were added/updated for the collected files. To get a true measure of the effectiveness of WCD identities, we plotted the eventual detections (Figure 2).

The largest detection slice was known bad packers (EncPk), followed by Zbot, FakeAV, ransomware and ZAccess. Dorkbot, Brazilian bankers, keygens and keyloggers each accounted for less than 1% of total detection. We also detected 2% of the WCD blocked files as exploits. The 'Other' slice was a mixed bag of detections that didn't belong to any of the popular malware families. Overall, this chart demonstrates how WCD identities effectively blocked different malware for which detections were absent and no URL reputation information was available at the time of the WCD block.

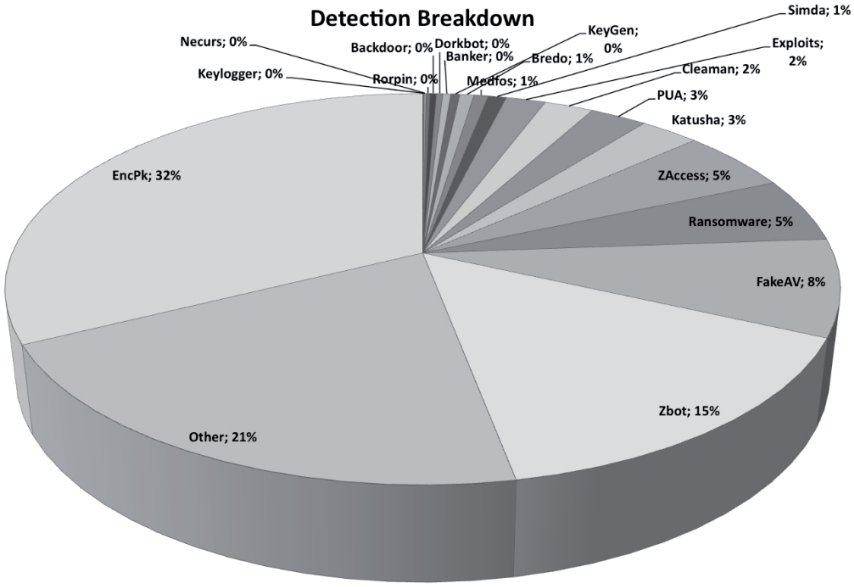

Figure 2: Malware family distribution of WCD identity blocked files.

CASE STUDIES

We released our WCD identities as CXwebs in the *Sophos Anti-Virus* product. In this section, we present case studies for four CXweb identities: CXweb/Pwned-A, CXweb/Zbot-A, CXweb/FkFlsh-A and CXweb/BadDlod-A. We describe the detection logic, provide a reference table of statistics and make observations.

CXweb/Pwned-A

CXweb/Pwned-A is a family-agnostic identity and is designed to detect likely compromised websites by utilizing URL path fragment information (one would not normally expect an executable to reside in, say, '/images/') in conjunction with Odd* properties of the file. We developed this identity using the following strategy:

1. Collect telemetry by mining URLFeed data for common compromised URL path fragments.
2. Select family-agnostic Odd* file properties.
3. Determine optimum threshold to reduce false positive risks.

A partial list of commonly observed path fragments in malicious/compromised URLs includes:

Path fragment	% Hits in known malicious URLFeed data
/wp-content/	0.8239
/images/	0.8225
/wp-admin/	0.0323

Table 1: Path fragment statistics.

In CXweb/Pwned-A, we don't inspect the hostname or use URL whitelists because we assume that any domain or host can be compromised. We require the PE file to have at least >2 Odd* features for the identity to block. We regularly add new path fragments observed in ongoing telemetry to improve CXweb/Pwned-A.

Traditional URL blocklists require the URL to be evaluated and then included in the list. URLs that have been compromised will experience an evaluation delay before they are included in the URL blocklist. This delay opens up the possibility of an infection. CXweb/Pwned-A leverages compromised path fragments together with PE file content inspection to block instantly if the PE file is evaluated to be malicious and thus prevents an infection.

Example URLs blocked:

- http://www.digiacomoandassociates.com/images/1114.exe
- http://sentierifrassati.org/tmp/smt.exe.

CXweb/Zbot-A

The CXweb/Zbot-A identity targets malware belonging to the Zbot family. It uses some Odd* file properties and a host of loose Zbot properties to determine whether the PE file is malicious. We don't inspect the URL in CXweb/Zbot-A as the PE file properties queried in conjunction with the downloaded context of the file produces a strong detection.

Example URLs blocked:

- http://clean.comparateur-acheter.com/t/f.php?k=1&e=0&f=0
- http://alino-is.de/UBz5.exe.

CXweb/FkFlsh-A

The CXweb/FkFlsh-A identity targets malware purporting to be '*flash*player*'. Filenames such as: update_flash_player.exe, install_flashplayer11x32ax_aih.exe and s-flashplayer.rar would all pass the filename check in CXweb/FkFlsh-A. The identity further checks that the hostname is not a known Adobe-owned domain. Among other things, the PE file is checked to make sure it is compiled in Delphi or dotNET or Visual Basic; we know for a fact that *Flash Player* is not compiled with the listed compilers.

Example URLs blocked:

- http://paulo.net.br/auto/install_flashplayer11x32_mssd_aih.exe
- http://96.126.98.56/adobe/update_flash_player.exe.

CXweb/BadDlod-A

CXweb/BadDlod-A is a family-agnostic identity that targets malware being hosted on IP-based hostnames. The IP address is parsed and validated to make sure it does not fall within any private ranges. Different MIs for the PE file are combined and if the aggregate score from the MIs is above a defined threshold, the identity blocks.

Example URLs blocked:

- http://37.1.198.56/lobster1/file.php
- http://199.91.154.50/fdmq31fqvlxg/za8qook82xzvjdh/MCPatcher-HD-1.4.7.exe.

CXweb comparatives

Table 2 contrasts the various ways WCD components are combined in the showcased CXweb identities.

Table 3 shows detection statistics collected for browser-downloaded PE files.

Identity name	Domain	Path	Filename	PE file properties
CXweb/Pwned-A		√		√
CXweb/Zbot-A				√
CXweb/FkFlsh-A			√	√
CXweb/BadDlod-A	√			√

Table 2: WCD identity creation strategy.

Identity name	Total blocks observed	Unique malicious SHAs	Unique malicious hosts	Triggered on unique endpoints
CXweb/Pwned-A	4,136	611	405	1,135
CXweb/Zbot-A	5,166	2,018	1,576	2,617
CXweb/FkFlsh-A	2,889	860	416	1,587
CXweb/BadDlod-A	8,576	1,175	1,387	3,266

Table 3: WCD identity statistics.

Based on the detection statistics, we make a few observations:

- We count unique malicious hosts instead of unique URLs because that masks the URL morphing.
- The low number of blocks observed does not suggest that the WCD identities are ineffective/stringent; rather the malware was already blocked by an existing identity or URL blocklist. In most cases, we tested the WCD identities that would have triggered if a detection or URL block did not already exist. This supports our strategy of using WCD identities to bridge the detection gap.
- The ratio (Unique_Malicious_SHAs:Unique_Malicious_Hosts) suggests server-side polymorphism. The ratios for our four CXweb identities are shown in Table 4.

Identity name	Ratio
CXweb/Pwned-A	1.51
CXweb/Zbot-A	1.28
CXweb/FkFlsh-A	1.94
CXweb/BadDlod-A	0.85

Table 4: Unique_Malicious_SHAs:Unique_Malicious_Hosts ratio.

A ratio value >1 suggests that multiple SHAs are being served from the same host.

- We observed that a good percentage of the URLs blocked by CXweb/Zbot-A were socially engineered.
- We observed similar URLs blocked by CXweb/BadDlod-A and CXweb/Zbot-A, suggesting that the Zbot authors are altering their features and deployment tactics.
- We observed that when generic detection for a family, e.g. Zbot, was updated, or a new batch of Zbot domains were blocked, the number of CXweb/Zbot-A hits decreased.
- The large volume of CXweb/BadDlod-A hits suggests possible drive-by-download attacks using browser exploits, plug-in exploits, Iframes, etc.

WCD DEPLOYMENT STRATEGIES

WCD identities are 'blocking' by design, as explained previously. The maximum impact of a false positive (FP) is users not being able to download that PE file. Smoke testing during release helps identify potential misfires on popular web content. To further reduce the risk of FPs, we created a safe release policy for WCD identities:

1. Initially release as a telemetry-gathering identity (silent mode) that reports results to the cloud.
2. The telemetry identity is actively monitored for the first 48 hours to identify detection spikes; users experience no blocks. Detection spikes must be analysed and vetted to make sure they are not FPs. If there are significant FPs, the detection logic is fixed and we go through a new deployment cycle.
3. Telemetry identities remain deployed for at least one week and are passively monitored.
4. Telemetry data is analysed and vetted. The detection logic is fixed for any FPs and we go through a time-reduced deployment cycle.
5. When the results are deemed acceptable, the telemetry identity is redeployed as a reporting identity.

FUTURE WORK AND DIRECTIONS

The WCD framework has matured significantly since its inception and we have several CXweb identities published. In our efforts to improve the WCD technology, we have identified a few key areas of improvement:

1. Implement an approximate string matching algorithm – we have observed in our URLFeed data malware using morphed filenames, e.g. filenames similar to ctfmon:
   - ctfm0n.exe – character substitution
   - ctmon.exe – character deletion
   - ctfmens.exe – character insertion + substitution
   - ctfmen.htm – character substitution + extension changed.

   An approximate string-matching algorithm will allow us to match the pattern ctfmon in the observed filenames. Malware uses this technique for social engineering and obfuscation; this problem is common in email spam.
2. We chose not to use HTTP Referer information in our experiments, but using it along with the URL information will allow for finer grained determination as to the origin of the 'click'.
3. Extend the WCD framework to support non-PE file formats e.g. PDF, SWF, DOC. These formats are very noisy (high download volume) in comparison to hosted PE files because they are found more commonly. Our current WCD framework partially supports non-PE files, but we do not extract and process all the URL information for them. In our initial testing, we observed that deep inspection for every downloaded element results in overhead and must be optimized with performance considerations. In the future, we plan on adding support for select non-PE file formats which are abused by malware, e.g. DOC, PDF, but which have relatively low-volume deployment compared to HTML, JPG, GIF, etc.

CONCLUSION

In this paper we introduced the concept of web-context detections and described how they are designed to bridge the detection gap that exists when no file detection or URL reputation information is available for a URL hosting a malicious PE file. We demonstrated techniques for building flexible WCD identities that resist field testing and are difficult to reverse. Through our research and field testing, we demonstrated that context information used during scan time is a valuable tool and adds another layer of proactive protection, particularly in the case of web threats where we can easily leverage the URL information to build broader and more effective detections. Using WCD identities, we successfully blocked malicious files that were later detected as FakeAV, Ransomware, Zbot, ZAccess, Katusha, etc. and improved the overall endpoint protection.

REFERENCES

[1] Sophos. What are Web Threats? http://www.sophos.com/en-us/security-news-trends/security-hubs/web/what-are-web-threats.aspx.

[2] Sophos. Security Threat Report 2013. http://www.sophos.com/en-us/security-news-trends/reports/security-threat-report.aspx.

[3] Sophos. Understanding Context-Based Detections. http://www.sophos.com/en-us/support/knowledgebase/113332.aspx.

[4] Malware Patrol. Malware Patrol – Stats. http://www.malware.com.br/stats.shtml.