A Broad View of the Ecosystem of Socially Engineered Exploit Documents

Stevens Le Blond, Cédric Gilbert, Utkarsh Upadhyay, Manuel Gomez Rodriguez
Max Planck Institute for Software Systems
{stevens, cedricg, utkarshu, manuelgr}@mpi-sws.org

David Choffnes
Northeastern University
choffnes@ccs.neu.edu

***Abstract*—Our understanding of exploit documents as a vector to deliver targeted malware is limited to a handful of studies done in collaboration with the Tibetans, Uyghurs, and political dissidents in the Middle East. In this measurement study, we present a complementary methodology relying only on publicly available data to capture and analyze targeted attacks with both greater scale and depth. In particular, we detect exploit documents uploaded over one year to a large anti-virus aggregator (VirusTotal) and then mine the social engineering information they embed to infer their likely targets and contextual information of the attacks. We identify attacks against two ethnic groups (Tibet and Uyghur) as well as 12 countries spanning America, Asia, and Europe. We then analyze the exploit documents dynamically in sandboxes to correlate and compare the exploited vulnerabilities and malware families targeting different groups. Finally, we use machine learning to infer the role of the uploaders of these documents to VirusTotal (i.e., attacker, targeted victim, or third-party), which enables their classification based only on their metadata, without any dynamic analysis. We make our datasets available to the academic community.**

I. Introduction

Targeted attacks are low-volume, socially engineered communications that convince specific victims to install malware. By definition, these attacks are hard to capture, making it challenging to design effective countermeasures against them. Recently, several researchers started collaborating with the Tibetans, Uyghurs, as well as political dissidents in the Middle East, to shed light on the threats faced by these groups [11], [16], [17]. In particular, all these projects found that modern targeted attacks often relied on *exploit documents*, i.e., documents exploiting vulnerabilities in Microsoft Office and Adobe Acrobat Reader on Microsoft Windows, to deliver malware to their victims. A fundamental limitation of the approach taken by previous work, however, is that it is labor- and time-intensive, limiting coverage to a few chosen groups. For example, each of the above papers represented a multi-year effort to build trust and to share suspicious samples acquired manually over substantial time periods.

In contrast with the approaches above, Anti-Virus (AV) aggregators acquire large numbers of suspicious samples via public web portals that allow anyone to scan files against multiple, different AV products. Aggregators leverage the fact that a collection of AV products from different vendors often detect complementary, partially overlapping sets of malware, so their aggregation provides a useful service to improve overall detection over any single AV product. Such aggregators are popular, e.g., VirusTotal receives millions of weekly submissions from all over the world [32]. Interestingly, AV aggregators can scan all file types supported by partnering AV software: Although most submitted files are executables, VirusTotal receives over 12 million (non-executable) document submissions per year.

We focus here on exploit documents because they are the most common vector of targeted attacks identified by related work [11], [16], [17] and they are arguably more dangerous than Office macros, which require additional user approval and can be forcibly disabled by system administrators. While malicious Office macros are commonly used in large-scale, opportunistic attacks such as those delivering ransomware [13], they tend to be used less frequently in targeted attacks. For example, out of the hundreds of malicious documents analyzed by the related work only one embedded a malicious Office macro [17]. Finally, recent reports indicate that exploit documents represent an important attack vector against a range of targets including NGOs [14], news agencies [6], and military, governmental and intelligence agencies [18], motivating the need to study the corresponding ecosystem.

This paper presents a measurement study of the ecosystem of exploit documents and in particular, their likely targets, exploited vulnerabilities, embedded malware families, and uploaders, using only documents uploaded to VirusTotal. This approach allows us to scale our analysis to hundreds of thousands of documents from tens of thousands of users, but poses substantial challenges such as reliably filtering benign and other types of malicious documents, clustering exploit documents based on their targets, and inferring the likely role of their uploaders. By addressing these challenges, we identify a variety of targeted attacks not reported in previous work and perform a comparative analysis of their respective characteristics in the wild. We specifically seek to answer the following questions:

- *Do targeted groups upload exploit documents on VirusTotal and if so, can we distinguish them from the bulk of other submissions?*

NDSS '17, 26 February - 1 March 2017, San Diego, CA, USA
Copyright 2017 Internet Society, ISBN 1-891562-46-0
http://dx.doi.org/10.14722/ndss.2017.23250

- *Can we automate the detection and mining of exploit documents to scale our analysis to hundreds of thousands of samples?*
- *How do the attacks faced by different targeted groups compare with each other?*
- *Is VirusTotal used by other actors (e.g., attackers and researchers) and if so can we reliably classify and quantify them?*

To answer these questions, we must first address three main challenges. *First*, we must determine the provenance of exploit documents uploaded on VirusTotal despite their acquisition through a third party. Although both the related work [1], [15] and AV reports (e.g., [5], [8], [22], [28], [29], [30], [31]) offer/use methods to reliably cluster a large number of malware samples based on their behavior or family, target inference requires a higher level of abstraction to cluster attacks based on their targets, independent of the exploited vulnerability and malware family used. *Second*, we must distinguish exploit documents from legitimate ones, determine the version of the vulnerable reader, and extract the embedded files and URLs. Analyzing exploit documents in this way typically requires a significant amount of manual intervention, which cannot scale to hundred of thousands of documents. This is due to a lack of specialized toolchains and methodologies for analyzing exploit documents. Although traditional malware sandboxes work well for executables, they tend to be agnostic of the specificities of exploit documents such as the diversity of software configurations in which exploits might trigger. *Third*, because VirusTotal allows many different types of anonymous users, it is difficult to distinguish among victims, attackers, and third parities such as researchers or companies submitting samples in bulk. For example, attackers and targeted victims (*targets*) may be using VirusTotal to evade AV software and detect targeted attacks, respectively. Indeed, there is anecdotal evidence of attackers abusing VirusTotal to drive down the detection rate of AV software [35]. Targets may also use VirusTotal, for example, to scan document attachments before opening them. Combining the analysis of exploit documents with the uploaders' behavioral information can enable us to classify uploaders and to infer the context in which attacks took place.

Our key findings are as follows. We use social engineering information embedded in exploit documents, along with malware behavior, to identify hundreds of attacks against two ethnic groups (Tibet and Uyghur) and 12 countries spanning America (the US), Asia (India, Indonesia, Japan, Mongolia, Myanmar, Philippines, Russia, South Korea, Taiwan, Thailand, and Vietnam), and Europe (France). The decoy documents targeting these groups use a variety of techniques to allay the victim's suspicion: They are often written in the victims' native languages (and/or English), frequently use content likely acquired from compromised hosts, cover a small number of targeted topics, and tend to use fairly recently created documents.

Second, we find that attacks against different groups tend to use disjoint malware families. Namely, 79% (219/277) of families are found in only one country in our dataset; further, malware found in multiple countries tend to be located in a relatively confined region. This specialization makes it difficult to generalize any findings based on analysis of attacks targeted only at one group (e.g., Tibetans or Uyghurs). The vast majority of samples exploit vulnerabilities reported two or more years prior to our measurement period. Thus, these attacks often rely on victims using unpatched software.

Third, we observe that the uploaders of exploit documents on VirusTotal exhibit distinguishable behaviors that enable automatic and reliable role inference. Using ground-truth information from VirusTotal complemented with manual inference for a subset of users, we train a machine-learning classifier that accurately predicts whether a user is an attacker, target, or third-party. We find that a semi-supervised algorithm based on label-propagation is able to achieve about 80% accuracy with few labeled seeds.

To summarize, our main contributions are as follows:

- A methodology to analyze targeted attacks relying on exploit documents without requiring direct acquisition from the targeted groups.
- A comparative analysis of exploit documents targeting various groups uploaded to VirusTotal over one year.
- The inference of the roles of the uploaders of these exploit documents.
- An interactive service to analyze exploit documents which we make available to the academic community at https://slingshot.dedis.ch.

Finally, to make our results reproducible and foster research in this area, we release our datasets of exploit documents and extracted malware, their CVEs, decoys' coding, and malware tagging.

II. Background

A. Terminology and Infection Process

We focus on documents exploiting vulnerabilities in Microsoft Office or Adobe Acrobat Reader. Such documents embed an exploit, a shellcode or gadgets' addresses and optionally, a decoy document and malware (as defined below).

Exploit. An exploit is a malicious input subverting the control or data plane of the vulnerable application [12].

Shellcode/gadgets' addresses. The goal of an exploit document is to execute malware on the victim's computer. Although historically, the (shell)code responsible for malware execution was injected into the vulnerable process' address space, mitigations such as DEP now preclude code injection. Therefore, modern attacks rely on legitimate code units mapped in the process' address space (*gadgets*) that are located and executed out-of-order by leveraging addresses embedded into the exploit document.

Decoy document. A decoy document is an optional, benign document that is embedded into the exploit document, written to disk, and opened with the reader application after a successful exploitation. As the purpose of a decoy is to conceal the infection from the victim, attackers have an incentive to tailor decoys to their victims.

Malware. Finally, the malware is the malicious software's binary that is executed on the victim's computer.

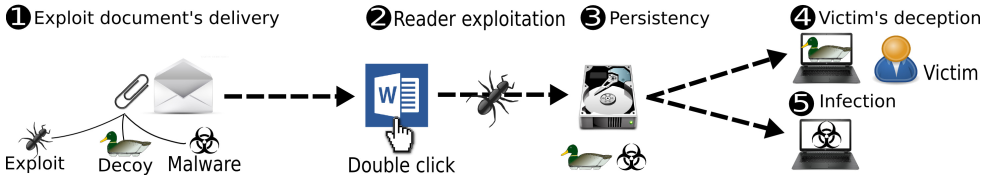

Fig. 1: Exploit document infection process.

From the time a victim opens an exploit document to the installation of malware on their system, there is a chain of events that underpins a successful exploitation. We present a sample infection process for a real exploit document in Figure 1, obfuscating only the victims' identity and content of the decoy. The process begins with an exploit document being delivered to the victim, e.g., via a malicious email (❶). If the victim opens the attached document and her reader is compatible with the exploit, the exploit is triggered (❷). The shellcode or gadget then writes the decoy and/or malware to persistent storage (❸), opens the decoy with the reader (❹), and executes the malware (❺).

B. Related work

Drive-by downloads. Several studies focused on opportunistic exploitations of web browsers (drive-by downloads) [10], [20], [24], [26]. In 2005, Moshchuk et al. [20] crawled 18 million URLs to identify spyware and drive-by downloads. They found that 13.4% of the crawled URLs led to spyware, and 5.9% to drive-by downloads. More recently, Provos et al. showed that drive-by downloads represented the largest threat to average users and examined how exploit sites appeared to users in search results and through syndicated advertisements [26]. Finally, Grier et al. [10] built on Provos et al. by examining the emergence of exploit-as-a-service whereby host exploitation is uncoupled from its monetization. In contrast with all these works, we focus on *low-volume*, *socially engineered exploit documents* luring *specific victims* into installing malware.

Targeted malware. Recently, researchers mined malware sandboxes available publicly to detect new malware families [9]. They found that such samples, including targeted malware, were sometimes submitted before being reported in the wild. Our work is complementary and focuses on the analysis of exploit documents as an attack vector to deliver such targeted malware.

Targeted attacks. The three most related studies were done in collaboration with the Tibetans [11], Uyghurs [16], and political dissidents in the Middle East [17]. They revealed that attacks against these groups were performed via socially engineered emails with malicious attachments. Exploit documents represented 81% $(481/592)$ and 71% $(799/1,116)$ of the malicious attachments targeting the Tibetans [3] and the Uyghurs [16], respectively. (Although exploit documents were also employed against Middle East dissidents, they were not quantified in the related work.) The majority of the remaining malicious attachments comprised of executables attached directly, or in an archive, and only one malicious attachment was a macro document [17]. We presume that executables and macro documents were a minority because they are blocked by popular webmails (e.g., GMail) and they are disabled by default in Microsoft Office, respectively. Because each of these studies focused on a single group, they could not compare the exploit documents targeting different communities.

C. Scope of this paper

This paper is a measurement study that composes existing tools and techniques to build a novel methodology enabling the analysis of exploit documents that embed socially engineered decoys. Though our methodology should be applicable to any dataset of documents, the analysis results presented in this paper are limited to the data in our study (documents uploaded to VirusTotal), which introduce two main observational biases. First, despite VirusTotal's popularity its coverage of targeted attacks is limited to those users and organizations who upload suspicious files. Second, in addition to its partial coverage, VirusTotal's visibility is likely skewed towards users who work with non-classified material. As a result of these biases, our VirusTotal dataset offers a partial coverage of attacks where individuals and NGOs are likely over-represented. Despite this limitation, we show that this dataset captures attacks against a wide range of targets including corporations and governmental institutions located in 12 countries located in America, Asia, and Europe. Finally, this paper focuses on the ecosystem of exploit documents with respects to their targets, exploited vulnerabilities, malware families, and VirusTotal uploaders. Generalization to different AV aggregators, malicious Office macros, and detailed analyses of exploitation and evasion techniques (both for reader exploitation and role inference) are the subject of future work (cf. Section VI).

III. Methodology

In this section, we describe our methodology for acquiring, processing, and validating the VirusTotal dataset. Figure 2 summarizes our datasets, methods, and important statistics.

The input to our analysis is a large feed of documents from VirusTotal and a publicly available dataset of exploit documents from the World Uyghur Congress [16] (❶). We use the Enhanced Mitigation Experience Toolkit (EMET) [19] in a variety of controlled environments to detect exploit documents (❷), then use an extractor that separates the embedded malware and decoys from the exploit documents (❸). Native speakers manually annotate the extracted decoys to give contextual information about the attacks (❹). We tag the extracted malware families using two enterprise-grade malware sandboxes (❺). Finally, we infer and analyze the role of uploaders of exploit documents (❻). (With the exception of Step ❹, all steps were fully automated.)

The rest of this section describes all these steps in detail except for ❻ which we defer to Section V.

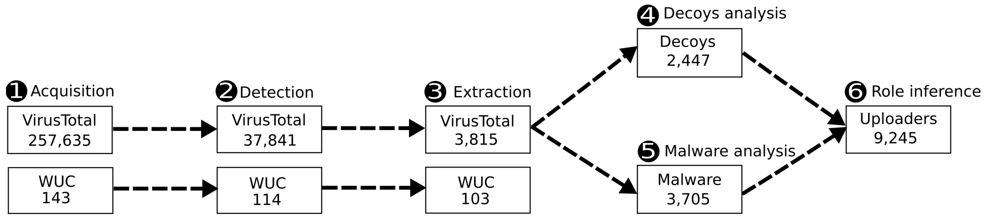

Fig. 2: Our data acquisition and processing workflow. Each box displays the dataset type and the number of corresponding entries after each step.

A. *Acquiring Suspicious Documents*

Our analysis relies on two complementary datasets: A large dataset of suspicious documents uploaded to VirusTotal and a small dataset of exploit documents acquired directly from the World Uyghur Congress (WUC) [16] (❶). We used the WUC dataset as ground truth to validate that our analysis has no or few false negatives and that decoys are indeed tailored to their victims.

VirusTotal. VirusTotal is the largest publicly available AV aggregator at the time of writing. Upon submitting a file to VirusTotal, a user is provided with a list of scan results from different AV software (71 at the time of this writing). This functionality makes VirusTotal popular among victims of targeted attacks as they generally have a low detection rate by individual AV software. Behind the scenes, AV vendors who do not detect a file detected by others are given its payload, providing visibility into attacks that may otherwise go unnoticed by that specific vendor.

VirusTotal Intelligence API. To facilitate sharing with AV vendors and other members of the security community, VirusTotal implements a rich API providing registered users with access to the uploaded content and its meta-data. VirusTotal gave us access to this API between April 2014 and March 2015. We used the following three API features to retrieve documents and their metadata.

- ***Distribution.*** The *distribution* function gives access to a live feed of VirusTotal uploads (in our case, all Office and PDF documents). We use this function to *acquire our dataset of documents*.
- ***Report.*** The *report* function returns the detailed results of the AV scans and additional meta data such as the first upload date. We use this function to determine the *detection rate* of our documents.
- ***Submissions.*** The *submissions* function returns a list of submissions with for each submission, the inferred geolocation and the salted hash of the uploader's IP address. We use this function to determine the *uploader* of an exploit document.

Characteristics of the VirusTotal dataset. Our dataset contains all Office and PDF documents (13.8 TB) uploaded between April 2014 and March 2015. The corresponding uploaders were geolocated in 210 countries with $2,574$ daily users and $49,048$ daily submissions on average. A concern is that VirusTotal may not catch a significant fraction of targeted attacks. As we find in Section IV-A, VirusTotal false negative rate[1] is very low for the uploaded samples ($< 0.1\%$). We then perform extraction on all documents detected by at least one AV software over one year ($257,635$). We analyze their decoys in Section IV-B, associated malware and CVEs in Section IV-C, and their uploaders in Section V. We summarize these statistics in Table I.

TABLE I: Summary of our dataset. *Time frame* corresponds to the acquisition period, *Size* to the volume of documents in terabytes, and *Users* and *Countries* to the number of unique uploaders and their countries. Finally, *Extraction* ($\geqslant 1$ *detection*) refers to all the detected documents uploaded over one year and on which we performed our measurement study.

Time frame	Apr 2014 - Mar 2015
Size (TB)	13.8
Users	$255,926$
Countries	210
Extraction ($\geqslant 1$ *detection)*	$257,635$

World Uyghur Congress (WUC) dataset. We include a public dataset of 143 exploit documents collected over four years by one WUC volunteer [16]. This dataset has been extensively analyzed in the related work and constitutes a ground truth to validate our methodology. Unlike VirusTotal, this dataset is composed exclusively of exploit documents that have targeted the Uyghur ethnic group.

B. *Detecting Exploit Documents*

The next step is to identify exploit documents for the purpose of extracting their malware and decoys (❷). A key challenge in this step is that the dynamic analysis tools we use may exhibit false negatives resulting in files not being written to disk by exploit documents. We address this by performing detection in Virtual Machines (VMs) with multiple software configurations and manually inspecting all discrepancies.

Dynamic analysis platform. Our initial analysis revealed that most exploit documents target different versions of readers on Microsoft Windows. Thus, to increase the attack surface and diversity of our controlled environments, we created multiple VMs, each of which uses Windows XP as a guest operating system and one version of Microsoft Office or Adobe Acrobat Reader. We created eleven VM images for Office and 12 for Acrobat Reader: Office 2003 and 2007 (each one with and without Service Packs $0 - 3$), and 2010 (with and without Service Packs $0 - 2$), and Acrobat Reader VIII (0.0 and 1.0),

[1] The AV detection rates were collected on the day of the upload.

TABLE II: Breakdown of the filtered data at each step of the processing workflow. *Office macro* corresponds to Office documents requesting user permission to activate macros, *Cannot open* to documents which could not be opened by any versions of our readers, *Crashes* to documents making at least one reader version crash, *Passwords* to password protected documents, *False positives* to potential AV false positives, *Neutralized* corresponds to a subset of detected documents which had been neutralized, and *Others* to the remaining detected samples. Finally, *Download* and *No executable or decoy* correspond to exploit documents downloading files from the network and not embedding any executables or decoys, respectively.

Steps	*Filtered categories*	*# documents*
❷ *Detection*		257,635
	Office macros	−129,532
	Cannot open	−17,177
	Crashes	−3,370
	Passwords	−1,001
	False positives	−45,342
	Neutralized	−5,574
	Others	−17,798
❸ Extraction		37,841
	Downloads	−32,387
	No executable or decoy	−1,639
❹-❺ *Analysis*		3,815

IX (0.0, 1.0 2.0, 3.0, 4.0, and 5.0) X (0.0, 1.0, and 1.4), and XI (0.0).

We chose these variety of reader versions to expose exploit documents to potentially vulnerable software, and used Windows XP because it has fewer exploit mitigation mechanisms than more recent Windows versions. We also automatically scrolled down documents to trigger exploits that might not be triggered by viewing only the first page. Our software was compiled for an x86 architecture. The analyses below required over two years of total CPU time.

Enhanced Mitigation Experience Toolkit (EMET). EMET enhances Microsoft Windows with modern mitigation techniques [19]. To detect exploit documents, we opened documents in a controlled environment with EMET installed, parsed logs for successful mitigations, and manually investigated detections for which no files were extracted. Although EMET has limitations [4], [25], we found no evidence of evasion in our dataset and also cross validated our results with ground truth from the WUC dataset (Section III-D).

Filtering of malicious documents. To validate that our methodology has no or few false negatives, we also quantified documents that were detected by at least one AV but that were not detected by EMET or for which no files were written to disk. To do so, we categorized the filtered documents that were not detected by EMET into six categories: *Office macros*, *Cannot open*, reader *Crashes*, *Password* protected documents, AV *False Positives*, and *Neutralized* (Table II). We further categorized the documents that were detected by EMET but for which no files were written to disk as *Downloads* and *No Executable or decoy*.

Office Macros. To determine whether documents contained *Office macros*, we opened them with Office 2003 and automatically captured the text contained in the message dialog. Under default security settings, Office warns the user that documents containing macros can be dangerous. By determining whether the message text corresponded to a macro warning, we identified 85,088 macro documents. In addition, macros created using versions predating Office 2003 are deactivated in recent Office versions (i.e., they do not trigger a dialog and cannot be executed under normal security settings). To capture these macros, we repeated the above procedure after modifying Office's security settings to *Medium*. Doing so detected 44,444 additional macro documents (129,532 in total).

Cannot open. Other documents could not be opened because they were corrupted, had the wrong file extension or format. As for *Office macros*, problematic documents generate a message dialog indicating that Office or Acrobat Reader cannot open the document. We identified 17,177 such documents which could not be opened using any of our eleven Office versions.

Crashes. We observed 3,370 documents which made at least one version of our readers terminate unexpectedly. While the root causes of these crashes varied and could not easily be determined, they could be due to, for example, legitimate documents triggering a reader bug, malicious documents exploiting a flaw in an unsupported reader version, or dysfunctional exploit documents such as the ones we will discuss in Section IV-A.

Password. 1,001 document were protected and could not be opened without the correct password.

False positives. AV software may exhibit false positives due to static signatures being triggered on legitimate samples or dysfunctional ones that dynamic analysis does not detect. Although the size of our dataset precludes a manual analysis, 45,342 documents in our dataset were detected by only one or two AV vendors suggesting that they may be false positives.

Others and Neutralized. After filtering the above, disjoint categories, we are left with 23,372 samples belonging to other categories. Manual inspection of the remaining samples, revealed 5,574 samples embedding an executable binary but which had been neutralized. In particular, the document opened normally in Microsoft Office but did not exhibit any malicious behavior. We believe that such cases might correspond to *crypting*, whereby attackers repeatedly tweak a malicious payload to reduce its detection rate by AV software. These neutralized samples illustrate the difficulty of classifying all samples uploaded to VirusTotal, since many of them contain malicious components that are not functional and thus cannot be detected by dynamic analysis.

Downloaded and no executable or decoys. Finally, some documents were detected by EMET but did not write any files to the filesystem. 32,387 of these documents unsuccessfully attempted to download files from the network and 1,639 did not embed any executable or documents.

C. Analyzing decoys, malware and CVEs

This step consists of extracting the malware and decoys from the 37,841 exploit documents detected by EMET (❸), labeling their distinguishing characteristics (❹), and analyzing the dynamic behavior of their embedded malware (❺).

Extraction system. To inform our analysis, we designed and implemented a Windows driver ($9,884$ lines of C and assembly code) that records filesystem and network activities performed by Microsoft Office and Acrobat Reader upon opening documents in a controlled environment. The use of syscalls related to filesystem and/or networking API calls by these processes and their children is recorded by our driver, enabling us to recover the content of the files written to disk after a successful exploitation, and the IP addresses, hostnames, and URLs contacted by shellcodes or gadgets.

Coding of decoys. We manually annotated (*coded*) the extracted decoys according to their languages, the countries they refer to, ethnic groups and dates, whether they targeted specific individuals or organizations, and whether they were likely exfiltrated from compromised systems and used as decoys in exploit documents targeting new, related victims (*replayed*). In contrast to replayed decoys, some decoys were empty or just contained a few words, e.g., "hello", "wrong version", or "passphrase". We coded these decoys as *empty* or *poor*, respectively.

In the first phase of coding, one researcher coded all documents. Those written in a language different from English were translated using a translation software. In the second phase, volunteer native speakers independently coded the documents written in Russian, Traditional Chinese, Uyghur, and Vietnamese. Finally, we compared the sets of code, merged the original notes with the volunteers', and fixed minor discrepancies (e.g., missing dates).

Malware analysis. We also dynamically analyzed the malware extracted from exploit documents in two malware sandboxes specialized in the analysis of targeted malware (a FireEye AX appliance and another one whose vendor wished to remain anonymous). To the best of our knowledge, commercial malware sandboxes are the only solutions readily available to analyze thousands of samples.

We used two such sandboxes to tag the malware. The malware sandboxes determined the malware family using static (e.g., YARA rules) and dynamic signatures (e.g., fingerprinting of communications).

Vulnerability analysis. We collaborated with a large AV vendor to determine the CVE tags of the exploited reader vulnerabilities. To do so, the vendor scanned all the exploit documents that we detected and compared the resulting CVE with the majority of VirusTotal tags. If the two CVEs matched, no further action was taken unless there was a doubt that the exploit was a zero-day vulnerability. Samples for which the CVE release date was posterior to the date of the upload on VirusTotal were examined manually to determine the CVE's correctness. Finally, if the AV's scanner did not reveal any CVE or if the CVE did not match, the sample was also analyzed manually.

D. Discussion

Limitations. We have described our methodology to detect exploit documents and separate and analyze their malware and decoys. This methodology provides a rich dataset for analysis of exploit documents, but may be subject to the following limitations.

First, dynamic analysis can exhibit false negatives, preventing us from extracting malware and decoys from exploit documents. We took steps to limit the impact of these false negatives, namely by systematically comparing our extraction results with EMET and manually accounted for all discrepancies (as described in Section IV). One reason we may fail to extract malware is when exploit documents target third-party plugins (i.e., Flash) or versions of Office and Acrobat Reader that we do not support. We found that very few documents used Flash (52 out of $843,483$) and did not install it on our platform. To reduce false negatives due to unsupported versions, we opened all documents with many reader versions. For comparison, EMET detected 114 out of the 143 malicious documents (79.7%) from the WUC dataset. Out of the 29 documents that were not detected by EMET, 16 targeted Mac OS X, nine affected a different reader version, two were password protected, and two were ciphered XLS documents in which we could not identify any exploits. We presume that the absence of evasion is due to the complexity of concealing exploitation and the lack of widespread exploitation mitigation in the wild. Importantly, none of our analyses depends on the lack of evasion techniques in the malware embedded in exploit documents.

Second, our analysis of decoys and malware is limited to documents embedding these files as opposed to downloading them over the network. Despite this limitation, our analysis revealed $3,815$ exploit documents with embedded decoys or malware. The WUC dataset indicates that this approach was preferable for attacks targeting the Uyghur community, as the vast majority (90.3%) of exploit documents embed malware or decoys. One explanation for this behavior is that embedding files offers the advantage of enabling the compromise of hosts even when they are not connected to the Internet.

Finally, our decoy analysis is limited to exploit documents that embed intelligible information for analysis. Although decoys are not fundamental to the malware used in the attacks, in practice, they are essential to present the victim with a document that is not suspicious. As a result, we expect most targeted attacks to employ decoys that are relevant to the victim. The WUC dataset confirmed this hypothesis with 41 out of 59 documents with decoys (69%) pertaining to the Uyghur community. Out of the 18 remaining decoys, 12 were poor or empty, two pertained to Tibet or Taiwan, and we could not classify the remaining four decoys (e.g., only pictures).

Ethical considerations. The analysis of data that may contain private information raises ethical concerns. As one of the goals of VirusTotal is to promote security research, its users agree to share their submissions with the security community. Our study does not collect any user's personal information beyond what is submitted to VirusTotal, nor does it expose users to additional risks. The results we present in this study can be split into two categories: aggregate and anonymized. In most cases, aggregate results are sufficient however; it is sometimes necessary to present results for individual users to validate aggregates. In these cases, we anonymized the results so that it was not possible to identify users. This study was approved by our IRB.

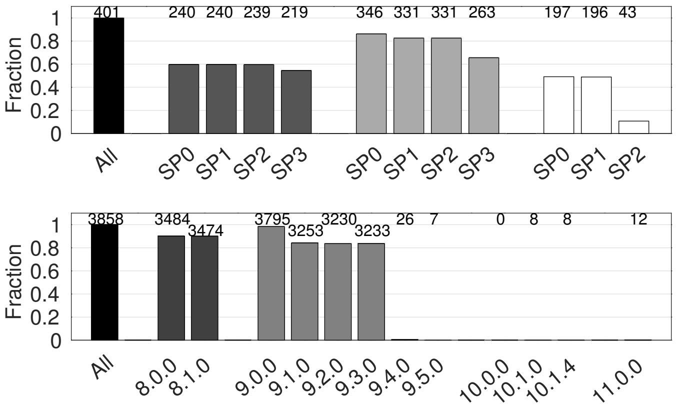

Fig. 3: Fraction of exploit documents compatible with each version and subversion of Microsoft Office (top) and Adobe Acrobat Reader (bottom). For Office, the subversion correspond to Service Packs (SPs). There is one bar group for each reader version: Office 2003, 2007, and 2010, and Acrobat 8, 9, 10, and 11. **Although exploits do not affect all readers' versions, an exploit against a given version tends to affect all its subversions.**

IV. Analysis of exploit documents

In this section, we seek to answer the following questions by analyzing documents submitted to VirusTotal:

- *Can we reliably detect exploit documents?* What is the impact of reader versions? How many versions do exploits generally affect?
- *Can we leverage decoy documents to infer the likely targets of attacks?* If so, can we identify attack trends in the VirusTotal dataset based on their targets?
- *Do targeted attacks against different target groups exhibit the same characteristics or do they differ from each others?* Do they use the same or different malware?
- *How does our observed malware coverage compares with previous work focusing on single communities?*

Previous work focuses on targeted groups [11], [16], [17] or individual malware families [5], [8], [22], [28], [29], [30], [31]. Our analysis extends this complementary work by linking malware families with the decoy documents used for targeting groups.

A. Extraction effectiveness

Before analyzing the decoys and malware embedded in exploit documents, we first demonstrate the our extraction process experiences few false negatives. To this end, we dynamically analyze all the documents uploaded to VirusTotal in February 2015 to identify exploit documents and validate that we properly extract the decoys and malware written to disk. Our goal is to scale extraction to the entire dataset while minimizing the number of false negatives (i.e., exploit documents from which we do not extract the corresponding files). To do so, we test that *(a)* documents detected by the Enhanced Mitigation Experience Toolkit (EMET) were also extracted and that *(b)* exploit documents detected by EMET were also detected by at least one AV software. These conditions would indicate that extraction has no or few false negatives as compared to EMET detections and that, in aggregate, AV software can be used as an oracle for the exploit documents submitted to VirusTotal, respectively.

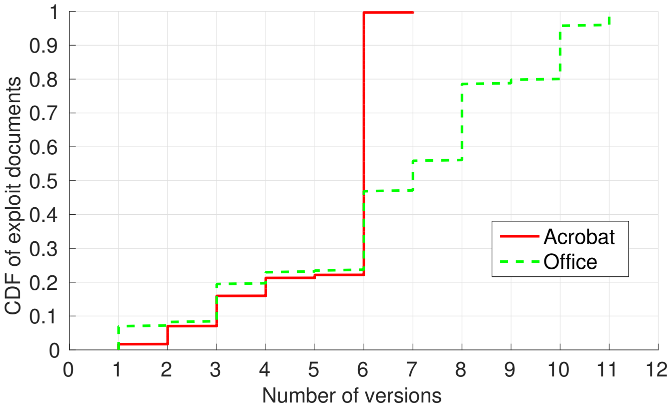

Fig. 4: CDFs of the number of versions affected by Acrobat and Office exploit documents. **Few exploits are portable across all reader versions.**

Impact of reader versions. Out of the documents submitted for one month, EMET detected $4,259$ unique exploit documents (401 Office and $3,858$ PDF files). We found that an exploit affecting a given reader's version tends to affect many subversions (Figure 3). This result indicates that knowing a target's reader version is sufficient for an attacker to achieve a near-perfect trigger rate across all its subversions. At the same time, we found that few exploits are portable across all reader's versions (Figure 4). While this result is solely based on empirical observation, it is clear that any analysis toolchain should support many different reader versions.

Unextracted samples detected by EMET. Out of $4,259$ documents detected by EMET, 29 did not write any files to disk. Manual inspection of these 29 exploit documents revealed that none of them were extraction false negatives.

- *Crashes.* There were six cases where exploit documents made the reader crash during exploitation.
- *Experimental samples.* In four cases, the document merely executed a command such as *calc.exe*.
- *Dysfunctional samples.* In the remaining 19 cases, the exploit relied on an absent property of the document (e.g., incorrect file size). Such cases led to a bug (e.g., an infinite loop in shellcode or gadgets) that prevented exploitation.

AV detection of exploit documents. All but three of the $4,259$ exploit documents detected by EMET were detected by at least one AV software. This result indicates that, in aggregate, AV software has very few false negatives for exploit documents uploaded to VirusTotal.

Summary of results. We showed that extraction had no or few false negatives as compared to EMET and that, in aggregate, AV software retroactively detected the vast majority of exploit documents. In particular, we confirmed that all the exploit documents that we could not extract were dysfunctional or experimental samples, or crashed before performing disk write operations, and all the extracted documents but three were detected by at least one AV software. In the following, we leverage these results by focusing our study on documents uploaded over one year and detected by at least one AV software. Doing so enables us to greatly reduce the number of analyzed samples while keeping the number of extraction false negatives to a minimum.

B. *Social engineering*

In this section, we characterize the decoys embedded in exploit documents. In total, $2,447$ (64%) out of the $3,815$ exploit documents embedding files contained decoys. (Note that we now focus on the subset of all detected files which wrote files to disk.) 965 (39%) of them were empty or poor, leaving us with $1,482$ (60%) meaningful decoys.

Languages, ethnic groups, and countries. The most common languages in our dataset were English, Chinese, Uyghur, Vietnamese, and Russian. In particular, 659 (44%) of our decoys were written in English, 235 (15%) in Chinese, 179 (12%) in Uyghur, 140 (9%) in Vietnamese, 68 (4%) in Russian, and 150 (10%) in another language. The decoys sometimes also enabled us to infer the likely targeted ethnic groups or countries. For example, 237 (15%) and 115 (7%) decoys referred to the Uyghur and Tibetan ethnic groups, respectively, and 145 (9%) to Vietnam, 118 (7%) to the USA, 100 (6%) to Taiwan, 72 (4%) to India 51 (3%) to Russia, 50 (3%) to Japan, 38 (2%) to the Philippines, 19 ($<2\%$) to South Korea, 17 to Myanmar, 14 to Mongolia, 9 to Thailand, and 7 to Indonesia. We show the distribution of languages and dates for each of the main groups in Figure 5. We see that these attacks tend to employ the mother tongue and official languages of ethnic groups and countries, with the most common other language being English.

Replayed decoys and organizational targeting. A significant fraction of decoys pertaining to the Uyghur, Vietnam, and Philippines were likely replayed, as we see in Figure 6. In addition, we found that these groups with a high rate of replayed decoys tended to exhibit organizational targeting. For example, out of the 237 Uyghur decoys, 210 (88%) referred to the World Uyghur Congress (WUC) and were likely replayed. Furthermore, all 38 Filipino decoys (19 of which replayed) were military documents referring to the navy or armed forces. They included four lists of navy and air force personnel (names, ranks, date of births, and telephone numbers). Finally, we additionally identified 12 decoys describing a research project in which a French defense contractor was involved, all of which were replayed. (We will cover these attacks in details in Section V-F).

Dates of decoy documents. We found dates in 662 (44%) of the decoys in our dataset. If such dates are pertinent to the decoy document, we would expect that exploit documents would use ones with more recent dates to increase the likelihood of a user opening the document.

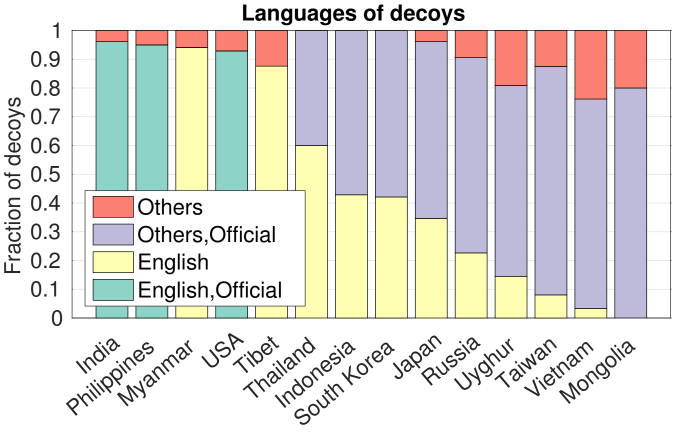

Fig. 5: Distribution of languages employed in the decoys targeting the various groups in our dataset. For each group, we show the the fraction of decoys written in *English*, the *Official* language of the group (when different from English), and *Other* languages. **Decoys tended to use the official language of the groups they target.**

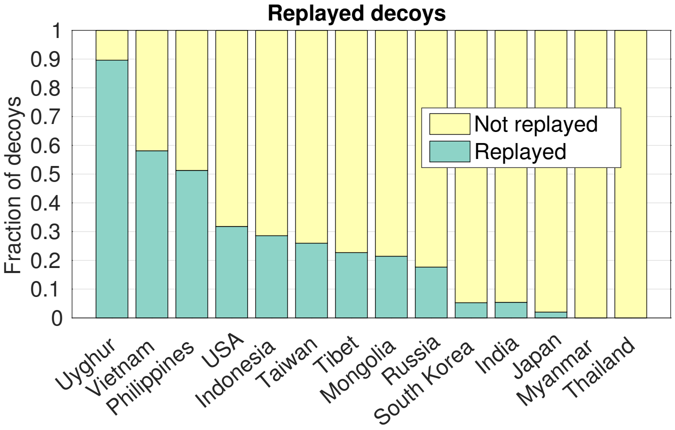

Fig. 6: Fraction of the decoys that were likely acquired from compromised hosts and embedded into exploit documents (*Replayed*). **The majority of Uyghur, Vietnamese, and Filipino decoys were replayed, indicating that these groups were deeply compromised. All groups but Myanmar, Thailand, and Japan had replayed decoys.**

To investigate this, we plotted the years found in decoy documents for different regions in Figure 7. We find that the documents they span several years with many attacks referring to dates before 2012. The time of attack varied substantially across groups, with attacks against groups under Chinese influence (e.g., Taiwan, Uyghur, and Tibet) tending to be older than those against Asian Pacific countries (e.g., Myanmar, Vietnam, and Thailand). For example, $80-85\%$ of attacks against Vietnam and Myanmar occurred in $2013-2015$.

Summary. Collectively, we find that exploit documents may employ a variety of strategies to minimize suspicion from victims. Namely, many use content in the victims' native language, (presumably stolen) content pertaining to specific region or organization, and use relatively fresh content as opposed to older documents. While different exploit documents use different subsets of these strategies, our analysis reveals a

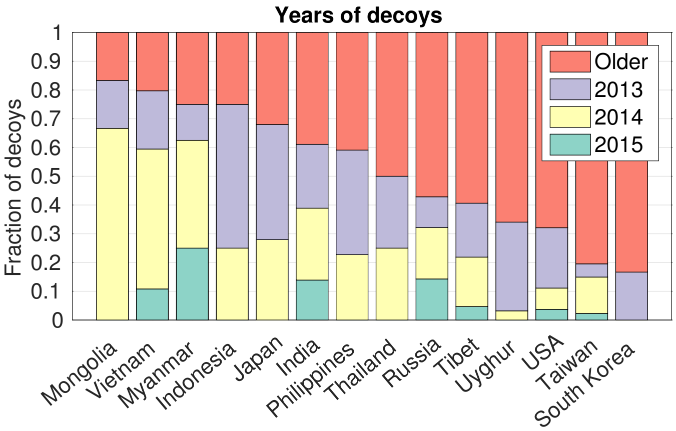

Fig. 7: Distribution of the likely times of attacks for the decoys embedding dating information. We show decoys from *2015-2013* as well as *Older* decoys. **The timing for the bulk of attacks greatly varied from one group to another. All groups uploaded exploit documents containing decoys referring to a least one year in the range of 2013-2015.**

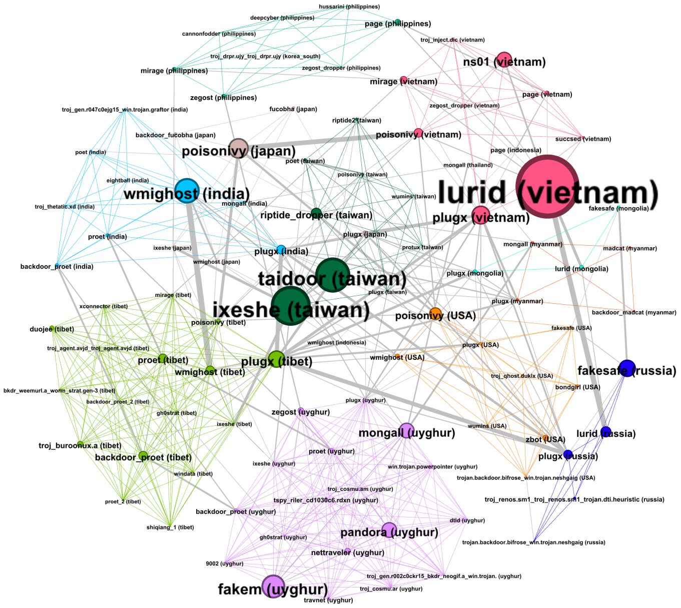

Fig. 8: Graph of malware families for each country where two family nodes are linked if they target the same country (different color for each region). In addition, we use gray links to indicate when the same malware family is used in different regions. The node colors and sizes correspond to their countries and the number of samples targeting it, respectively. **The malware families in our dataset tend to target one main ethnic group or country.**

reasonably high level of sophistication for social engineering.

C. Malware and CVEs

In the following, we leverage decoys to study the characteristics of attacks targeting different groups.

Malware families. Our malware sandboxes tagged the family for 91% of the 3, 958 exploit documents that wrote malware to disk. The most popular malware families in our dataset were *Miniduke* [33], *PlugX* [22], *Lurid* [29], *IXESHE* [28], *Taidoor* [30], *Zeus* [34], *FakeM* [31], *WMI* [8], and *Elise* [5]. Miniduke was the most represented family in our dataset due to several different exploit documents embedding the same Miniduke executable; however, fewer than 1% of these samples were used jointly with a decoy. This is in sharp contrast with other above families, where a decoy was written to disk in 41–100% of the cases, indicating *some* targeting. In the case of Zeus, only 41% of all documents were used jointly with decoys. This is likely because Zeus is a banking trojan, as we discuss below. Excluding Zeus, all remaining families are commonly used for espionage and had a decoy in 79% of the cases or more.

Malware distribution. To show regional clustering of the countries in which malware families were found, we plot the families using a graph (Figure 8). Each node represents a (family, country) pair and its size is proportional to the number of samples in our dataset. Then we use two types of edges: colored ones represents links to other malware in the same country, and gray ones represent the same malware in other countries. Note that most edges are colored, meaning that most malware is found in only one country. Further, even when the same malware appears in multiple countries, most cases appear in a small number. This has implications for both direct acquisition and AV reports (as we will discuss below).

We now focus on the minority of malware samples impacting victims across multiple regions in Figure 9. For each malware family, the bar shows the fraction of samples submitted from each region. The figure shows that several malware families are targeted broadly across regions (e.g., PlugX, Mongall, Mirage, PoisonIvy) with no region seeing a majority of samples, while others tend to focus on a small number of regions (e.g., Lurid, FakeM, FakeSafe). We did not see any cases of exploit documents affecting every region in our dataset, indicating that even when distributed beyond a single country, the reach of exploit documents is still quite limited.

The time distributions varied from one family to another. IXESHE, Taidoor, and FakeM have not been actively used since 2011-2012, whereas PlugX, WMI, and Elise have been used uniformly during the observation period. Interestingly, IXESHE, Taidoor and FakeM's inactivity loosely correlates with the publication of the corresponding AV reports (2012 for IXESHE [28] and Taidoor [30] and 2013 for FakeM [31]). To the best of our knowledge, no comprehensive AV report on PlugX has been published to date and the first AV report on Elise was published in 2015 [5]. *Thus, our analysis reveals that malware authors are responsive to AV detection and adapt accordingly.*

Distribution of CVEs. Surprisingly, despite the diversity of the decoys and malware families, the exploit documents in our dataset used a relatively small number of vulnerabilities, as seen in Figure 10. CVE-2012-0158 was by far the most common CVE in our dataset with 1, 504 occurences. Recent CVEs were minoritary with 39 samples with CVEs from 2014. These results indicate that old vulnerabilities with patches available still constitute the bulk of the attacks. We did not

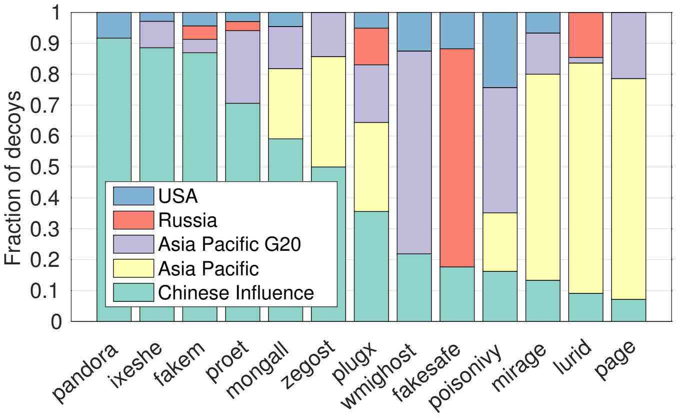

Fig. 9: Distribution of regions where each malware family was found, *only for malware found in multiple countries and more than 20 samples (13/282 families)*. *Chinese Influence* includes Uyghur, Tibet, and Taiwan; *Asia Pacific G20* includes India, Indonesia, Japan, and South Korea; and *Asia Pacific* includes Myanmar, the Philippines, Thailand, and Vietnam. **We find with the exception of PlugX, malware tends to be found in one or two main regions; further, the primary region(s) are different across malware families.** Thus, socially engineered malware tends to have a limited set of targets.

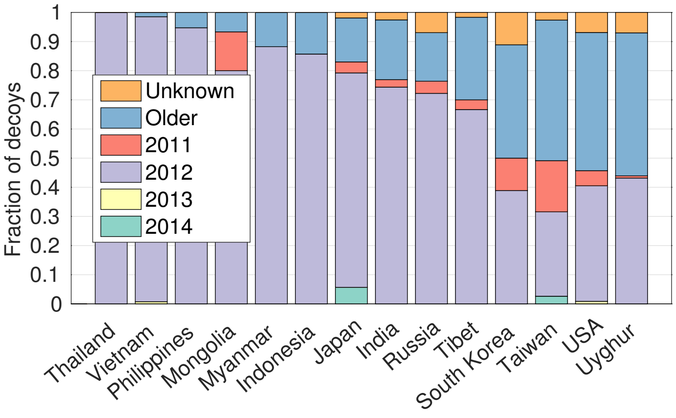

Fig. 10: Distribution of the CVEs' release years for the exploited reader vulnerabilities. *2014-2011* and *Older* correspond to CVEs released during the corresponding years and to older CVEs, respectively, and *Unknown* to vulnerabilities that could not be associated with a CVE. **Most CVEs in our dataset were released in** 2012 **and before and relied on unpatched readers to compromise users during the measurement period.**

find evidence of zero-day exploits in our dataset.

Limitations of direct acquisition and AV reports. The specialization of targeted malware has important implications for generalizing results found via direct acquisition from targeted groups and AV reports. In particular, specialization complicates the discovery of targeted samples as acquiring them from one group (e.g., Uyghur or Tibetans) is unlikely to reveal families targeting others, with the exception of PlugX. Similarly, an AV deployment would need to span enough groups to potentially gain access to all families. To make the matter worse, we find that the distribution of families per targeted groups is skewed with the top four common families representing just 18% of all malware.

Summary, validation, and new results. We showed that decoy documents often embedded sufficient contextual information to infer the targets of exploit documents. Once coded, decoys enable a more comprehensive analysis of exploit documents by clustering attacks independent of their malware family. Our second new result is that targeted malware is highly specialized, meaning that a collaboration and/or AV deployment would need to span many groups to over a good coverage of its landscape. Third, we showed that the distribution of malware families targeting individual groups was skewed, further reducing the generalizability of AV reports.

V. Analysis of Uploaders

Next, we infer and analyze the role of users who uploaded exploit documents on VirusTotal using only meta-data. Role inference is useful to categorize users, study their relationships, and/or inform the analysis of their uploads. For example, it could enable companies such as VirusTotal to identify attackers abusing their services, or researchers working on targeted attacks to prioritize the analysis of documents uploaded by targeted victims. Importantly, role inference has the potential of doing all this by relying only on meta-data, i.e., *without* coding or dynamic analysis of the exploit documents or malware. To the best of our knowledge, this paper is the first attempt to rely on machine learning to systematically infer the role of an AV aggregator's users. We seek to answer the following questions.

- *Can we accurately infer the role of VirusTotal users based solely on their meta-data?* What is the distribution of users with each role? What are the most important features that can differentiate between users in different categories?

A. Definitions

The goal of the analysis in this section is to infer whether uploaders of exploit documents correspond to attackers, targeted victims or third-parties. Although these roles may not be the only interesting ones, we argue they are the most general and that other roles would be variants of these. We will describe the techniques to infer these roles in more details in Section V-D.

- **Attacker.** An attacker corresponds to the *source* of an exploit document.
- **Targeted victim.** A targeted victim is the *destination* of a exploit document with a targeted decoy. They are referred to as *targets* in the ensuing discussion.
- **Third party.** A third is neither the source or destination of an exploit document. Members of this category typically upload many documents but these are rarely exploit documents.

Limitations. The ground truth for the roles of the uploaders was only available for companies and we had to manually infer targets and attackers. As a result, the labeling of these two categories is likely imperfect. Despite this limitation, we will see that our machine-learning algorithm (relying on a different set of features) accurately classified users from each category.

B. Ground truth and manual inference

VirusTotal shared with us the role of several premium accounts that belonged to third-parties, giving us ground truth for that category. For the other categories (i.e., targets, and attackers), we inferred the roles manually as follows.

- *Targets* uploaded at least one extracted document with a targeted decoy, a plausible email attachment filename, as well as at least one benign document.
- *Attackers* uploaded at least one extracted document sharing a command and control server with targets but predating their uploads, and geolocated in a different country. We manually inspected the uploads of the corresponding users and kept only those who had uploaded experimental samples.

The purpose of the ground truth and manual inference is to validate the roles inferred by machine learning. Although machine learning uses only meta-data, manual inference leverages all the data made available through coding and dynamic analysis. As a result, researchers without the resources to code or analyze all malicious documents can use role inference to guide their analysis.

C. Features

Our goal is identifying a set of user features such that users with high feature similarity play the same role in the AV aggregator (be it an attacker, a target, or third-party). Here, we distinguish two types of features: user specific features and neighborhood features.

We build each user's specific features using her own uploading activity. We consider the following specific features:

- **Fraction of exploit documents uploaded, total number of uploads.** *Attackers* are expected to upload a disproportionate fraction of exploit documents.
- **Unique filenames vs unique hashes.** For attackers using VirusTotal as a crypting service, may upload files with different hashes but using the same filename.
- **Fraction of unique filenames used multiple times for documents with different detection rates.** For attackers using VirusTotal as a crypting service, we expect the detection rates of exploit documents uploaded consecutively over short time periods (e.g., hours) to decrease until they reach zero. To avoid false positives, we only consider exploit documents sharing the same filename.
- **Maximum upload delay, average of the top 10%-ile of upload delay.** *Upload delay* is defined as the interval between the first upload of a document (by anyone) and the time when this user uploaded it (identity being established using the hash). We expect this to be low for the attackers (close to zero) and high for the targets and researchers who receive their documents from the attackers.

To build each user's neighborhood features, we first construct a directed *co-hash* graph. In this graph, each node corresponds to a user that uploaded at least one of the exploit documents. We form an edge from a user u to user v if u uploaded a document which was later also uploaded by v. We use the *hash* of the documents to identify such identical uploads. We refer to a user's one hop neighborhood on this graph (including all edges contained within) it as the *ego-network* of the user. We extract the following neighborhood features from this subgraph:

- **Out-degree, in-degree and LCC.** *Attackers* are expected to have a large out-degree since they are often the first to upload an exploit document and then they send the documents to others in the network. Hence, the in-degree can help identify the receivers (targets/researchers). The Local Clustering Coefficient (LCC) (computed on an undirected version of the graph) helps in identifying users which received exploit documents as part of wide spread attacks, i.e., who were not singled out and targeted. Due to paucity of neighbors, these users are expected to have a high LCC.
- **Edges in, into and out of the ego network, Neighbors' out and in degree.** We count the number of edges in the ego network, the number of incoming edges from nodes outside the ego network to nodes in the ego network and the number of outgoing edges from nodes in the ego network to nodes outside the ego network. This takes into account the features of the one hop network around the node. These features can help identify "cascades" which bring a exploit document from an attacker, through a target to a researcher. We also calculate the average, variance, minimum and maximum in-degree and out-degree across the nodes in the co-hash ego network.
- **Neighbors' fraction of total and malicious files uploaded.** This summarizes the activity of the neighbors of nodes. We compute the average, variance, minimum and maximum fraction of all and exploit documents uploaded across the nodes in the co-hash ego network. The activity of neighbors can give hints about the nature of the current node.

D. Semi-supervised light-weight inference

We manually, with the help of roles provided by VirusTotal, identified the role of 246 users, i.e., we *labeled* a few *seed* users, and filtered out users who did not upload extractable documents, which leaves 9,245 users. Given such a labeled dataset, a natural approach would be to use a supervised learning method, such as logistic regression or support vector machines (SVMs), to find a mapping between features and labels and use this mapping to infer the role of the remaining users. Unfortunately, since the number of labeled users is several orders of magnitude smaller than the total number of users, such a mapping would not generalize to the overall set of (unlabeled) users. Instead, here, we resort to semi-supervised learning [2], which leverages both labeled and unlabeled data to infer the role of all users. In particular, we use label propagation [36], which works as follows.

TABLE III: Confusion matrix produced for the seed users (5-fold cross validation). **The most common error is users in category Third-party being marked as Targets.**

True Inf.	Third-party	Target	Attacker
Third-party	**42**	20	6
Target	10	**125**	3
Attacker	0	10	**30**

It first computes an affinity matrix W, which captures the feature similarity between every pair of users (i, j). Then, it infers each user's role $\hat{r}_i \in \{1, \ldots R\}$ by leveraging the roles of the labeled users and assuming that users with higher similarity will be more likely to have similar roles. Here, for each node i, we can think of nodes j as the neighbors of i in an underlying similarity graph if $W_{ij} > 0$. The algorithm consists of the following steps:

```
Semi-supervised role inference:
  In.   Bandwidth parameter $\sigma$, Tradeoff parameter $\alpha \in (0, 1)$, and, features (i.e., $x_i$) standardized to have 0 mean and 1 variance.
  1.    Construct affinity matrix $W$ such that $W_{ij} = \exp\left(-||x_i - x_j||^2/2\sigma^2\right)$ if $i \neq j$ and 0 otherwise.
  2.    Construct matrix $S = D^{-1/2}WD^{-1/2}$, where $D$ is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$.
  3.    Construct $F^0 = Y$ such that $Y_{ij} = 1$ if user $i$ has been manually labelled as $r_i = j$ and 0 otherwise.
  4.    Iterate $F^{t+1} = \alpha S F^t + (1-\alpha) Y$ until convergence.
  Out.  If $\lim_{t\to\infty} F^t = F^*$, then $\hat{r}_i = \mathrm{argmax}_j\, F^*_{ij}$.
```

We can think of each column F^t_i in the matrix F^t as the amount of evidence that supports each potential role for user i. In the third step, W is normalized symmetrically to ensure the convergence of the algorithm. During each iteration of the fourth step, each user tradeoffs between the information she receives from her neighbors (first term) and their initial information (second term) by means of the parameter α. In our experiments, for scalability reasons, we build a kNN graph from W, instead of using the entire (dense) affinity matrix W, as proposed elsewhere [37], and use the open source implementation of the algorithm in `scikit-learn`, a well-known machine learning library [23]. For the kNN graph, we set the number of neighbors to $k = 18$ by 5-fold cross-validation on the set of seed users, which leads to the best performance in terms of F1 score [27], as shown in Figure 11. We see that the performance hits a plateau and stays stable after $k = 18$.

Table IV compares the performance of our semi-supervised light-weight inference method to two baseline methods: (i) a random classifier that assigns a class to each sample proportionally to the distribution of the classes in the seed set, and (ii) k-means, an unsupervised method that does not used the labels of the seed users [21]. Our method is the best performer across all classes, consistently beating the second best by a very large margin.

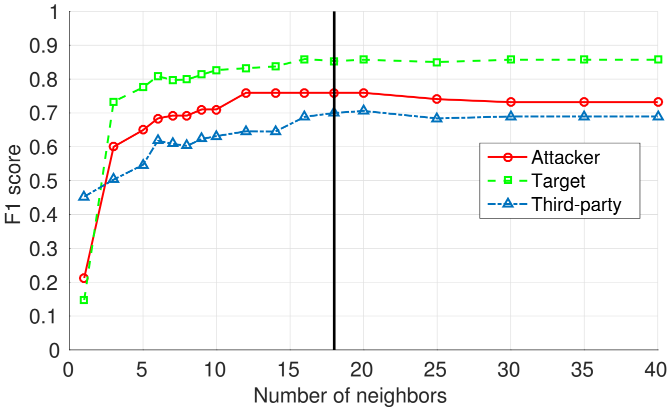

Fig. 11: Impact of the neighborhood size k on performance of the inference algorithm. The $F1$ score was calculated over 5-fold cross-validation [7]. The vertical red line indicates the k chosen for the analysis here. **The performance (F1 scores) becomes stable after $k = 18$ neighbours.**

TABLE IV: Comparing the performance (using the F1 score) of a random classifier and the classifier chosen after cross-validation. **Label propagation outperforms the random classifier and the k-means classifier by a large margin for all categories.**

Category	**Random classifier**	**k-means classifier**	**Semi-supervised classifier**
Third-party	0.28	0.61	0.70
Target	0.56	0.70	0.85
Attacker	0.16	0.00	0.76

The confusion matrix for the set of seed uses is shown in Table III. The targets are easy to classify while the third-party category is difficult to distinguish from targets. The overall accuracy of classification was 80.0%, compared to the baseline accuracy of 33.3% for random assignment and of 61.3% for k-means based classification.

E. Roles of users

Distribution of Roles. Next, we discuss inference on all users in our dataset. The distribution of users is shown in Table VI. As expected, there are many more targets identified in the dataset than other categories and relatively few third-parties when compared to the fraction of attackers in the seed set. This is because the third-parties often show features that are similar to those of targets and it is difficult to dis-entangle them completely.

Important features. After the inference was done, we investigated which features were most helpful in discriminating the different classes. Since label-propagation itself does not provide us with a measure of importance of each feature, we train several supervised learning methods using the user features and labels inferred using our semi-supervised method. Then, we use the corresponding feature *weights* given by these supervised methods as the proxy of their true importance.

Then we selected the features that the classifiers deemed the most important for each class. The features selected by different classifiers are shown in Table V.

TABLE V: Most important features for each category chosen by different classifiers. **These reveal the underlying characteristic behavior of users in each category (see Section V-E).**

Category	Decision Tree (Depth = 1)	Decision Tree (Depth = 10)	Extra Tree (Depth = 10)
Third-party	infected files	infected files max upload delay	unique filenames/unique hashes infected files
Target	total files	avg. num. of infected files uploaded by the ego net var. in num. of infected files uploaded by the net	out degree max upload delay
Attacker	unique files/unique hashes	edges into ego net unique files/unique hashes	edges into ego net unique files/unique hashes

TABLE VI: Distribution of roles of users in the *seed* set, which was manually labelled, and the inferred labels using the algorithm described in section V-D.

Category	Seed set	Inferred
Third-party	68	717
Target	138	7,148
Attacker	40	1,380
Total	246	9,245

For the third-party category, the feature *infected files* (which is highly correlated to the *total number of files uploaded*) helps distinguish them from other users. For the targets, the differentiating features are *average number of infected files uploaded by their ego network*. This is because the neighborhood of the targets is usually devoid of sources which upload a large number of infected files, i.e. researchers or heavy handed attackers. Attackers are relatively easily identified if they upload more than one file with the same filename but with different content (i.e. hash). Hence, the feature *unique files/unique hashes* is able to identify them. The importance of *edges into ego net* in identifying attackers also suggests that these attackers may be connected to targets who suffer coordinated attacks.

F. *Case study: French Defense contractor*

Next, we attempt to paint a holistic picture of an attack campaign against a French defense contractor by combining role inference, malware, and decoys. This type of postmortem analysis can inform how attacks are developed and evolve over time. Unlike the role analysis, we leverage all available indicators including Command and Control (C2), malware family, content of decoys, and benign documents.

Our dataset offered some coverage of the campaign against the defense contractor between April and November 2014. This campaign involved 12 participants that we verified manually: three targeted victims, eight attackers, and one researcher. We show the relationships between these participants, VirusTotal, and the C2s in Figure 12.

As expected, the three targeted victims involved in this campaign were geolocated in France and uploaded both benign and triggered documents. In total, they uploaded three triggered documents and 27 benign documents (most of them written in French). The three triggered documents uploaded by the victims contained the same replayed decoy discussed in Section IV.

More surprisingly, we also identified eight likely attackers who uploaded only malicious documents. Three of them were geolocated in France, two in Mainland China, two in Hong Kong, and one in the UK. In total, these attackers uploaded 14 triggered documents, nine with the replayed decoy, and five with empty of poor decoys. All empty/poor decoys were uploaded after June 13.

Interestingly, two triggered documents uploaded by the attackers dropped *calc.exe*. We assume that these uploads were final detection tests for the exploit and shellcode prior to launching the attacks. In one case, the attacker uploaded two documents embedding *calc.exe* and no decoy in less than ten minutes. Five hours later, the same attacker uploaded a document with the replayed decoy and embedded malware which connected to *sophos.skypetm.com.tw*. Around three hours later, one of the targets uploaded an identical document on VirusTotal.

We also see in Figure 12 that three C2s were used over the course of this campaign. The first C2 was used by the first five samples and the second one by all of the remaining ones but one. This phenomenon may correspond C2 rotation to evade network defenses however, it could also be an artifact of our partial coverage of this campaign. Despite the last upload from a victim having occurred in May, the fact that attackers continued to upload related documents in June suggests that the campaign may have continued during that period.

Finally, there was one researcher geolocated in India with a premium VirusTotal account. This researcher uploaded only one triggered file with a different hash than those that had been uploaded previously. The malware connected to *mca.av.store.tw* and other C2s and contained the replayed decoy. As we see in Figure 12, this researcher uploaded this document in November 2014, long after the other participants involved in this attack.

Summary, confirmations, and new results. We have presented a role inference algorithm which is able to achieve a reasonable accuracy over the dataset with a very small set of labeled seed users. We also verify that the features which emerge as important after the inference are indeed the features we would expect would classify the users. We are able to classify 9,245 users after manually labeling only 246 users, i.e. a ∼4000% reduction in amount of work needed for 80% accuracy. Hence, role inference stands to aid researchers by reducing the amount of manual work they will need to do to find "interesting" users in the dataset. As we see in the confusion matrix (Table III), it is difficult to differentiate between third-parties and targets. We speculate that third-parties

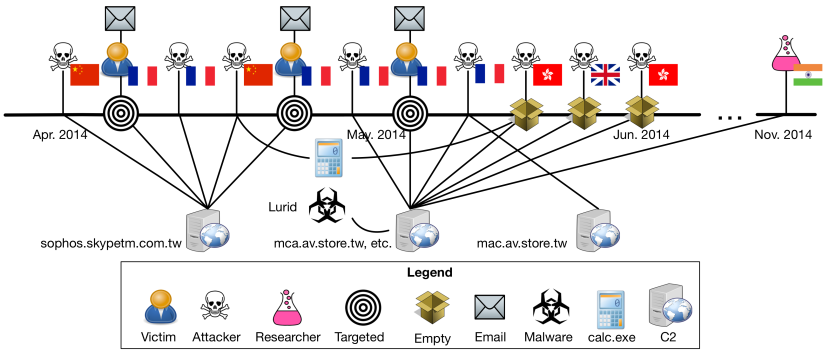

Fig. 12: Case study of the defense contractor. The horizontal line represents the timeline of uploads on VirusTotal by the participants involved in this campaign. We show the role of these participants (i.e., Targeted *Victim*, *Attacker*, and *Researcher*), their relationships, and the relationships between their uploads and C2s. **Several attackers participated in this campaign and their uploads were intertwined with those of the victims. The researcher uploaded last.**

are difficult to identify since their behaviour subsumes the behavior of targets and researchers. To resolve this ambiguity and to improve the performance of the algorithm, we will need to come up with better approaches to automatically classify the roles of users. It is possible that including more information (i.e. geo-location or IP addresses of users) can aid in role inference significantly.

VI. Discussion and Future Work

We presented a methodology for analyzing exploit documents without relying on direct acquisition from targeted groups. Importantly, although we used public tools to acquire and detect targeted samples (VirusTotal and EMET, respectively), our approach does not depend on any specific tool. For example, third parties such as targeted groups could adopt our approach to analyze internal document attachments, as we have shown with the WUC dataset. Similarly, EMET could be replaced or complemented with additional mechanisms to detect novel exploitation techniques. Below, we discuss some on-going and future work enabled by this paper.

Crowd-assisted detection. Common defenses deployed in email channels do not effectively block exploit documents. While the reason is not clear, we suspect it is due to a combination of email providers using too few AV software, the lack of static signatures for exploit documents in the wild, and/or attackers specifically evading them. Independent of the reasons, the poor efficacy of existing defenses motivates the deployment of more robust email defenses. We are currently experimenting with a Chrome extension that uploads document attachments to our analysis service and shows the detection results in the GMail interface. In the background, these documents are processed by our complete toolchain to help inform users of the threats they are facing. Our beta deployment with Tibetan, Uyghur, and journalist users already detected hundreds of attacks. By combining automated and manual detection, we hope to achieve a virtuous circle whereby attacks missed by EMET might be identified by users (e.g., because of poor social engineering), enabling us to improve mitigation.

Office macros. We focused on exploit documents because they are arguably more dangerous than macro documents which require additional user approval, can be blocked in the email channels, or disabled by system administrators. The comparative analysis between exploit and macro documents will be the subject of future work.

Evasion techniques. Both the mitigation of readers' exploitation and inference of uploaders' roles can be evaded. Although it is possible to evade EMET, we did not find evidence of evasion in Section III-D, presumably due to the lack of widespread exploit mitigations in the wild. Finally, we did not attempt to make our role inference techniques robust against evasion by attackers or privacy-conscious users. We leave the analysis of such evasion techniques as future work.

VII. Conclusion

We showed that AV aggregators constitute a rich, untapped source of information to study targeted attacks. We presented a novel methodology relying on publicly available tools and services for detecting exploit documents submitted to VirusTotal for one year. We leveraged EMET in a multiple of controlled environments to reliably detect exploitation of the Office and Adobe Acrobat readers. We then mined the social engineering information embedded into decoy documents to identifying hundreds of attacks against two ethnic groups (Tibet and Uyghur) and 12 countries spanning America (the US), Asia (India, Indonesia, Japan, Mongolia, Myanmar, Philippines, Russia, South Korea, Taiwan, Thailand, and Vietnam), and Europe (France). We additionally enriched our dataset with threat intelligence data to perform a comparative analysis of the malware families used against these groups and showed that socially engineered malware tends to be highly specialized for specific regions and groups, and that the exploits often use known vulnerabilities that rely on unpatched software (instead of zero-days). Finally, we showed that we could reliably and automatically classify the uploaders of these exploit documents using machine learning. It is our opinion that the dearth of academic research on targeted attacks is mainly due to the difficulty of acquiring datasets and we hope that this study will act as a catalyst for the community to pursue additional research on that topic. To this end, we release our datasets to researchers upon request and make our analysis system available as a service so others can reproduce and extend the results of this paper.

Acknowledgments

This paper is one of the most collaborative efforts initiated by the authors. We are particularly grateful to VirusTotal, the EMET developers, our translators, and vendors of AV software and malware sandboxes. We also thank the anonymous reviewers and our shepherd, Dongyan Xu, for their helpful feedback. This work was supported by the Max Planck Society and the European Research Council (ERC) under the imPACT Synergy Grant No. 610150. Finally, we thank the Decentralized/Distributed Systems (DEDIS) group at EPFL for hosting our services.

References

[1] M. Bailey, J. Andersen, Z. Morleymao, and F. Jahanian, "Automated classification and analysis of internet malware," in *Proceedings of Recent Advances in Intrusion Detection (RAID 2007)*, Queensland, Australia, 2007.

[2] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.

[3] Citizen Lab, "Malware indicators," https://raw.githubusercontent.com/citizenlab/malware-indicators/master/file-indicators.csv.

[4] J. DeMott, "Bypassing EMET 4.1," Bromium, Tech. Rep., February 2014, http://labs.bromium.com/2014/02/24/bypassing-emet-4-1/.

[5] R. Falcone, J. Grunzweig, J. Miller-Osborn, and R. Olson, "Operation Lotus Blossom," https://www.paloaltonetworks.com/resources/research/unit42-operation-lotus-blossom.html.

[6] FireEye, "The EPS Awakens - Part 2," https://www.fireeye.com/blog/threat-research/2015/12/the-eps-awakens-part-two.html.

[7] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement," *SIGKDD Explor. Newsl.*, vol. 12, no. 1, pp. 49–57, Nov. 2010.

[8] L. Galang, "WMI Abused for Malware Operations," http://blog.trendmicro.com/trendlabs-security-intelligence/windows-wmi-abused-for-malware-operations/.

[9] M. Graziano, D. Canali, L. Bilge, A. Lanzi, and D. Balzarotti, "Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence," in *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, Aug. 2015, pp. 1057–1072. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/graziano

[10] C. Grier, L. Ballard, J. Caballero, N. Chachra, C. J. Dietrich, K. Levchenko, P. Mavrommatis, D. Mccoy, A. Pitsillidis, N. Provos, M. Zubair, R. Moheeb, A. Rajab, C. Rossow, K. Thomas, V. Paxson, S. Savage, and G. M. Voelker, "Manufacturing compromise: The emergence of exploit-as-a-service," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS*, pp. 171–179.

[11] S. Hardy, M. Crete-Nishihata, K. Kleemola, A. Senft, B. Sonne, G. Wiseman, and P. Gill, "Targeted threat index: Characterizing and quantifying politically-motivated targeted malware," in *Proceedings of the 23rd USENIX Security Symposium*, San Diego, CA, August 2014.

[12] H. Hu, S. Shinde, S. Adrian, Z. L. Chua, P. Saxena, and Z. Liang, "Data-Oriented Programming: On the Expressiveness of Non-Control Data Attacks," in *IEEE Symposium on Security and Privacy (Oakland 2016)*, San Jose, CA, 2016.

[13] A. Kharraz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware," in *25th USENIX Security Symposium (USENIX Security'16)*, Austin, TX, 2016.

[14] K. Kleemola, M. Crete-Nishihata, and J. Scott-Railton, "Tibetan uprising day malware attacks," https://citizenlab.org/2015/03/tibetan-uprising-day-malware-attacks/.

[15] C. Kruegel, E. Kirda, P. M. Comparetti, U. Bayer, and C. Hlauschek, "Scalable, behavior-based malware clustering," in *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS 2009)*, San Diego, CA, 2009.

[16] S. Le Blond, A. Uritesc, C. Gilbert, Z. L. Chua, P. Saxena, and E. Kirda, "A look at targeted attacks through the lense of an ngo," in *USENIX Security*, 2014.

[17] W. R. Marczak, J. Scott-Railton, M. Marquis-Boire, and V. Paxson, "When governments hack opponents: A look at actors and technology," in *Proceedings of the 23rd USENIX Security Symposium*, San Diego, CA, August 2014.

[18] Microsoft, "PLATINUM: Targeted attacks in South and Southeast Asia," https://www.microsoft.com/en-us/download/details.aspx?id=51956.

[19] ——, "Enhanced Mitigation Experience Toolkit," 2015, http://www.microsoft.com/en-us/download/details.aspx?id=46366.

[20] E. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy, "A crawler-based study of spyware on the web," in *Proceedings of the Annual Network and Distributed System Security Symposium (NDSS 2006)*, San Diego, CA, 2006.

[21] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[22] R. D. Paz, "PlugX: New Tool For a Not So New Campaign," http://blog.trendmicro.com/trendlabs-security-intelligence/plugx-new-tool-for-a-not-so-new-campaign/.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] M. Polychronakis, P. Mavrommatis, and N. Provos, "Ghost turns zombie: Exploring the life cycle of web-based malware," in *1st Usenix Workshop on Large-scale Exploits and Emergent Threats*, 2008.

[25] A. Portnoy, "Bypassing All of the Things," 2013, https://www.exodusintel.com/files/Aaron_Portnoy-Bypassing_All_Of_The_Things.pdf.

[26] N. Provos, P. Mavrommatis, M. Abu, R. F. Monrose, G. Inc, N. Provos, P. Mavrommatis, M. Abu, and R. F. Monrose, "All your iframes point to us," in *17th USENIX Security Symposium (USENIX Security'08)*, vol. 2008, San Diego, CA.

[27] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

[28] D. Sancho, J. dela Torre, M. Bakuei, N. Villeneuve, and R. McArdle, "Taking a Bite Out of IXESHE," http://blog.trendmicro.com/trendlabs-security-intelligence/taking-a-bite-out-of-ixeshe/.

[29] D. Sancho and N. Villeneuve, "Trend Micro Exposes LURID APT," http://blog.trendmicro.com/trendlabs-security-intelligence/trend-micro-exposes-lurid-apt/.

[30] T. M. R. Team, "The Taidoor Campaign: An in-depth analysis)," http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp_the_taidoor_campaign.pdf.

[31] N. Villeneuve, "Hiding in Plain Sight: The FAKEM Remote Access Trojan," http://blog.trendmicro.com/trendlabs-security-intelligence/hiding-in-plain-sight-the-fakem-remote-access-trojan/.

[32] VirusTotal, "Virustotal file statistics," https://www.virustotal.com/en/statistics/.

[33] Wikipedia, "The miniduke attacks are back in force," http://www.kaspersky.com/about/news/virus/2014/The-Miniduke-attacks-are-back-in-force.

[34] ——, "Zeus (Malware)," https://en.wikipedia.org/wiki/Zeus_(malware).

[35] WIRED, "A google site meant to protect you is helping hackers attack you," http://www.wired.com/2014/09/how-hackers-use-virustotal/.

[36] D. Zhou, O. Bousquet, T. N. L., J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, no. 16, pp. 321–328, 2004.

[37] X. Zhu, *Semi-supervised learning with graphs*. CMU PhD Thesis, 2005.