Research Report: Building a Wide Reach Corpus for Secure Parser Development

Tim Allison*, Wayne Burke†, Valentino Constantinou‡, Edwin Goh§,
Chris Mattmann¶, Anastasija Mensikova‖, Philip Southam**,
Ryan Stonebraker††, Virisha Timmaraju‡‡
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, California
*timothy.b.allison@jpl.nasa.gov, †wayne.m.burke@jpl.nasa.gov, ‡vconstan@jpl.nasa.gov,
§edwin.y.goh@jpl.nasa.gov ¶chris.a.mattmann@jpl.nasa.gov, ‖anastasia.menshikova@jpl.nasa.gov,
**philip.southam@jpl.nasa.gov, ††ryan.a.stonebraker@jpl.nasa.gov, ‡‡virisha.timmaraju@jpl.nasa.gov

***Abstract*—Computer software that parses electronic files is often vulnerable to maliciously crafted input data. Rather than relying on developers to implement *ad hoc* defenses against such data, the Language-theoretic security (LangSec) philosophy offers formally correct and verifiable input handling throughout the software development lifecycle. Whether developing from a specification or deriving parsers from samples, LangSec parser developers require wide-reach corpora of their target file format in order to identify key edge cases or common deviations from the format's specification. In this research report, we provide the details of several methods we have used to gather approximately 30 million files, extract features and make these features amenable to search and use in analytics. Additionally, we provide documentation on opportunities and limitations of some popular open-source datasets and annotation tools that will benefit researchers which need to efficiently gather a large file corpus for the purposes of LangSec parser development.**

***Index Terms*—LangSec, language-theoretic security, file corpus creation, file forensics, text extraction, parser resources**

I. Introduction

Software that processes electronic files is notoriously vulnerable to maliciously crafted input data. Language-theoretic security (LangSec) is one software development method that offers assurance of software free from common classes of vulnerabilities. Whether LangSec parsers are built from formal specifications or are derived from samples, these parsers require wide-reach corpora for inference and/or integration testing throughout the development cycle. In this paper, we report on work to date in building a wide reach corpus to support the development of LangSec-based parsers. Specifically, in the early stages of this work, colleagues are applying LangSec techniques to build assured parsers for the Portable Document Format (PDF) file type.

The Portable Document Format – initially released by Adobe in 1993 – is immensely popular and in use globally in many applications, and has been the subject of research into malware detection [1] [2] [3] since the first virus was discovered in the PDF file type in 2001 (the *OUTLOOK.PDFWorm* or *Peachy* virus) [4]. The file type – which encapsulates text, fonts, images, vector graphics, and other information needed to display the document – is prone to manipulation by malicious actors and inconsistent implementations against the International Organization for Standardization (ISO) specifications outlined for each version of PDF (e.g. producing valid PDFs from malformed files) [5]. These challenges and continued wide-spread use of the file type provide the motivation for an initial focus on this file format.

We share our findings and report our work to-date in building a large-scale, wide-reach corpus; further, we discuss initial steps towards search and analytics on this corpus to enable research into features of "files in the wild" and the construction of development corpora for LangSec-based parsers using the attributes available for each file as filters in search. We believe that our lessons learned and work to-date will help address some of the challenges faced by researchers and parser developers who need to generate their own corpora. Further, we have plans to release our corpus generation and annotation tools to the general public to support LangSec-based parser development.

II. Background and Related Work

Since at least Garfinkel *et al.*'s seminal work in gathering and publishing a collection of one million files – GovDocs1 [6] – researchers and parser developers have recognized the value of publicly available, large scale corpora for developing and testing file parsers and forensic tools. In the open source world, for example, at least three Apache Software Foundation projects (Apache Tika [7], Apache PDFBox [8] and Apache POI [9]) rely on Garfinkel *et al.*'s corpus for large scale regression testing and have extended this corpus to include a richer set of more diverse and more recent file types [10] [11]. Additionally, researchers writing digital forensics (DF) tools have noted the value in curating large-scale corpora for development of these tools and their ability to enable direct comparison of different approaches and tools under develop-

The research was carried out at the NASA (National Aeronautics and Space Administration) Jet Propulsion Laboratory, California Institute of Technology under a contract with the Defense Advanced Research Projects Agency (DARPA) SafeDocs program.

DOI 10.1109/SPW50608.2020.00066

ment, in addition to providing the means for reproducibility [6] [12] [13].

Forensics tools and LangSec-based parsers are typically applied to datasets that are large and generated by human beings [6], and unique challenges exist when developing large-scale, wide-reaching corpora for digital forensics and development of LangSec-based parsers. There is a need for data diversity across file types, their content, and temporal attributes (date of creation, last modification date, and others) [12]. Due to the ubiquity of file formats such as PDF – which themselves contain near limitless combinations of content (fonts, images, links, etc.) – corpora used for development of LangSec-based parsers must be as expressive and diverse as possible in order to ensure coverage of possible stresses and edge cases presented to parsers. It is not enough to benchmark parser performance against files from a single creator tool, source organization or individual, point in time, geographic location, and so on.

There are also challenges in gathering files for and hosting multi-terabyte corpora for use by the research community. While cloud-based solutions are now available for corpora generation (file gathering or crawling) and both access to internet and connection speeds have improved dramatically, downloading multi-terabyte corpora in bulk is still not practical and may not be feasible, depending on the resources of the researcher(s). In addition, it is arguable that – at least from the perspective of generating corpora – that Jevon's paradox may be applicable in this environment [14], in that the scale and diversity of generating corpora is only limited by its cost computationally and financially. In other words, the growth in scale and diversity of corpora is a function of the efficiency in which a resource is used (in this case, computational resources for gathering and annotating files). Filtering and search tools are needed in order to locate specific files of interest.

As such, a parallel effort is being undertaken to facilitate search and descriptive analytics on extracted features in conjunction with gathering the corpus. Our model for this is VirusTotal [15], which allows users to search by a rich set of features [16]. VirusTotal offers a useful ontology as a basis for file types and features that should be supported in an analytics and retrieval system. In practice, however, researchers and parser developers require far more features to target specific aspects of the file format of interest - features that not only provide information about the characteristics of the files themselves but also the structure of their content. Further research and development effort is required to identify and provide features of interest to the LangSec community (like those features provided by VirusTotal) to enable search and analytics on file corpora and provide the ability to locate (and subset from a corpus) specific files of interest for research or development of secure parsers.

Corpus-Generation Architecture Overview

For the present purpose of developing a corpus for use by LangSec parser developers, we developed the pipeline shown in Figure 1. Various data sources that have been identified are piped into pre-processing blocks, which then store the files in an Amazon Web Services (AWS) S3 bucket. The key data source—Common Crawl—and the associated pre-processing steps will be further discussed in Section III.

These pre-processed PDF files are then sent from the AWS S3 bucket to several feature-extraction tools ranging from PDF parsers to anti-virus software, of which Apache Tika and Clam Anti-Virus are detailed in Section IV. The combination of tools will ideally generate a set of features that sufficiently characterize the content, structure, and malicious/adversarial nature of a given file. However, in the event that more features are required, the modular nature of this architecture allows for additional tools to be incorporated in the feature extraction step. For example, one could include a plethora of different anti-virus software to further explore the correlation between file content/structure and false positives.

Previous work has shown that providing files as stand-alone corpora significantly simplifies the level of effort needed in meta-data and text extraction [6], which is also applicable to the development of LangSec-based parsers. In our pipeline, features provided from feature-extraction tools are merged into Amazon's Athena database, which serves as the back-end to store features which are then merged and indexed into an Elasticsearch service. In conjunction with Kibana, the Elasticsearch Application Program Interface (API) enables researchers and developers to perform data analytics and visualization. Functionality is also in development to enable users to download subsets of the corpus that can be based on both simple and complex filters. Preliminary results performed on a subset of 20,000 PDF files using this pipeline are discussed in Section V.

III. Gathering Files

In the following sections, we describe three methods for gathering files: a) using Common Crawl data, b) focused, intelligent, link-based crawling with Sparkler, and c) custom API usage and/or scraping for high-value sites that may not have the traditional link structure required for link-based crawlers.

A. Common Crawl

Crawling the web is notoriously challenging and resource intensive [17]. However, the web offers a tremendous amount of real world, wide-reach data. The Common Crawl project [18] offers researchers one option for working with large amounts of "pre-crawled" data. In the next section, we offer an introduction to Common Crawl and then a brief description of how we gathered nearly 30 million PDFs from Common Crawl.

1) Common Crawl Basics: The Common Crawl project runs a monthly crawl across a large amount of the internet. The December 2019 crawl contained 2.45 billion URLs, comprising 234 terabytes (TB) of uncompressed content. For each crawl, the project offers four types of data [19] and [20]:

1) WARC – WebARChive format. This is a standardized format that for web archiving that includes the HTTP

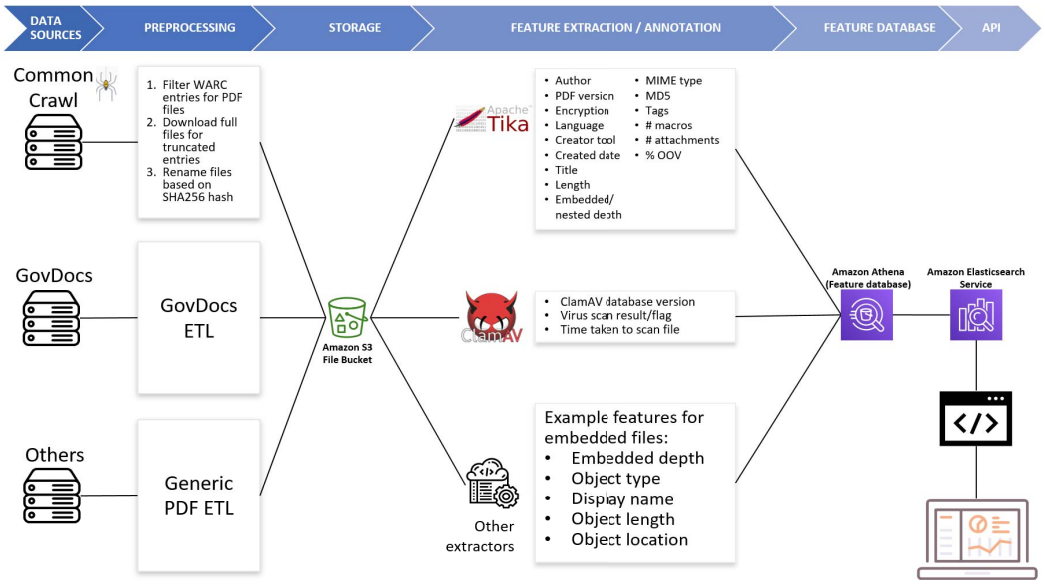

Fig. 1. Data pipeline for corpus generation illustrating file-gathering, preprocessing, storage, feature extraction, and subsequent deployment and analysis.

response status and headers (see Fig. 2 below), other provenance metadata and the raw bytes retrieved for a given URL (50 TB compressed)

2) WAT – Metadata files about the crawl (17.6 TB compressed)
3) WET – Text extracted from HTML, XHTML and text files (8 TB compressed)
4) URL Index Files – metadata for each URL including HTTP response status code, HTTP header content-type, detected content-type, detected language, whether the content was truncated

{date=Wed, 03 Jun 2015 21:34:52 GMT, server=Apache/2.2.3 (CentOS), expires=Tue, 03 Jun 2014 21:34:58 +0000, vary=Accept-Encoding, content-encoding=gzip, x-highwire-cache-cache-control=no-cache, content-disposition=inline; filename="1606.full.pdf", x-highwire-filestream-for=http://pdf.highwire.org/stamped/brain/135/5/1606.full.pdf, x-highwire-cache=no-cache, x-highwire-sitecode=brain, connection=close, content-type=application/pdf, cache-control=no-cache, max-age=0, must-revalidate, proxy-revalidate}

Fig. 2. An example of HTTP headers stored in a WARC file

Not surprisingly, the majority of retrieved files were HTML or XHTML. In Figure 3 and Table I, we report the top 10 most common file types in the December 2019 crawl as detected by Apache Tika.

Amazon hosts the data in AWS Public Data Sets, and researchers can process the files on AWS or download all the files or specific files from the web for local processing. Common Crawl publishes indices of the content to enable selection and extraction of specific files by original URL, detected file type, detected language or several other features.

When working with data from Common Crawl, the team noticed one major limitation and three areas that required consideration and/or further processing.

MIME	Number of Files
text/html	1,602,196,927
application/xhtml+xml	376,252,298
text/plain	50,931,060
application/octet-stream	23,184,879
UNKNOWN*	11,110,346
message/rfc822	2,680,373
application/atom+xml	2,660,439
image/jpeg	2,350,339
application/rss+xml	2,301,081
application/pdf	2,030,356

TABLE I
TOP 10 FILE TYPES IN THE DECEMBER 2019 CRAWL

The major limitation that the team noticed is that Common Crawl does not crawl the entire web nor even entire sites. As one example, we compared the number of pages returned by Google and Bing for the 'jpl.nasa.gov' domain, and we compared that with the number of documents in the Common Crawl index for the December 2019 crawl (II). The first three data rows represent the total number of files. The second three report the number of PDFs found on the site.

Search Engine	Condition	Number of Files
Google	site:jpl.nasa.gov	1.2 million
Bing	site:jpl.nasa.gov	1.8 million
Common Crawl	*.jpl.nasa.gov	128,406
Google	site:jpl.nasa.gov filetype:pdf	50,700
Bing	site:jpl.nasa.gov filetype:pdf	64,300
Common Crawl	*.jpl.nasa.gov mime= pdf	7

TABLE II
NUMBER OF PAGES BY SEARCH ENGINE AND FILE TYPE FOR 'JPL.NASA.GOV'

While the cause of this incomplete crawl is not clear, the team suspects that the cause may be that the 'jpl.nasa.gov'

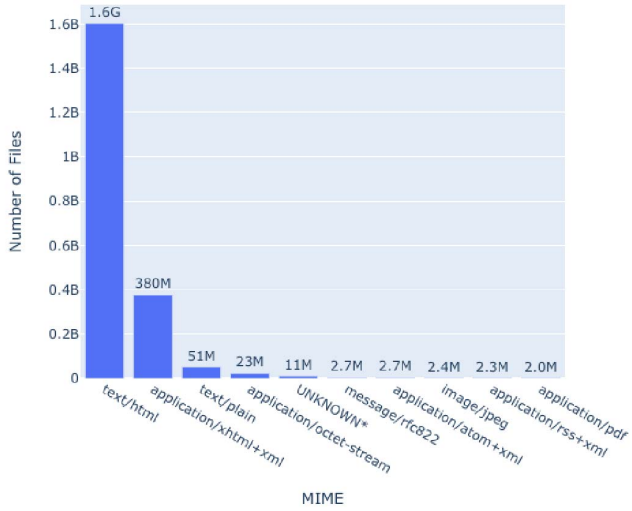

Fig. 3. Number of files by MIME type

site relies heavily on javascript and a crawler would need to render the javascript to extract all the links (with, e.g. headless chrome). If a crawler is only crawling links within HTML for this site (and others that rely heavily on javascript), the crawler will only be able to reach a small portion of any website.

While working with Common Crawl data, our team identified areas for further processing:

1) Common Crawl truncates files at 1 MB. If researchers require intact files, they must re-pull truncated files from the original websites. In the December, 2019 crawl, nearly 430,000 PDFs (22%) were truncated.
2) For rarer file types, one must gather files from across different crawls. The earlier crawls do not include the results of automatic file type detection, which means users have three options to select files by type:
   a) process all the data and run automatic file type detection on every file
   b) rely on the HTTP content-type header information
   c) rely on the file extension as represented in the URL
3) The datasets are large, and even the indices are large – the compressed index for the December 2019 crawl requires 300GB of storage. If not working in AWS or other cloud-based environments, researchers need to have appropriate resources (network bandwidth, storage and processing) to handle these data sets.

The team's takeaway from the above is that Common Crawl is an extremely useful resource for quickly gathering files generally, but it cannot be relied upon for a complete crawl of the web nor of specific sites.

2) Extracting PDFs: As an initial step, we selected crawls from 2013 to present. Because Common Crawl was not running file type detection on the earlier crawls, we performed file type detection on every file in the selected crawls and extracted the PDFs to S3 buckets. We added processing to re-pull "truncated" PDFs from the original URLs. As of this writing, we've gathered 30 million PDFs for researchers to use.

B. Sparkler

Fig. 4. Sparkler dashboard showing collected PDFs from ghostscript and the IRS website.

At its core, Sparkler is a Java-based web crawler that extends the functionality of Apache Spark [21], alongside other Apache projects, such as Kafka, Lucene/Solr, Tika, and pf4j. Sparkler is a much more extensible version of Apache Nutch that runs on Apache Spark Cluster. Sparkler was initially designed for use in DARPA MEMEX. However, due to its general purpose crawling capabilities, it has been employed in a variety of other projects, including DARPA SafeDocs. Alongside its universality, one of the biggest benefits of using Sparkler is its high performance, coupled with an extensive real-time analytics dashboard, which allows for controlled large-scale crawls. Sparkler also has an extensible plugin framework and comes prepackaged with many useful add-ons, including a plugin for JavaScript rendering, which allows webpages to be searched in their final rendered state. Sparkler Crawl Environment (SCE), on the other hand, is a set of tools built on top of Sparkler that provides an efficient software architecture that is used to enrich a domain by expanding its collection of artifacts. SCE conveniently provides a Docker-based command line interface (CLI) for building and running jobs. Due to the extensible nature of both, we have used Sparkler and SCE for experimental PDF crawls. Although not always successful, Sparkler has yielded many interesting results.

In order to use Sparkler, a crawling profile is first configured in YAML. For our purposes, we enabled plugins that allowed Sparkler to render JavaScript to resolve a wider range of links and restricted crawling to the initially specified host so as to not deviate too far from our target source. Additionally, Sparkler was configured to be "polite" and we restricted the amount of requests per second we made to any given website. Lastly, Sparkler is initially configured with a regular expressions filter to limit the pages it traverses. This filter is

setup to avoid pages that identify their content-type as a file or any links that end in common file extensions. In order to make Sparkler collect PDF files, this filter was appropriately modified. Once pages are crawled, Sparkler uses Apache Tika to extract the text content and concatenates them and stores them in Solr. While all documents can be recreated from this, we decided to slightly modify the Sparkler source code to dump found PDFs for automation purposes.

In order to use Sparkler at a more institutional level, we explored many options of scaling its deployment. Initially we tried using AWS Elastic Map Reduce (EMR) to host Sparkler. However, this proved to require significant manual configuration and was not as scalable as initially hoped, so we switched over to using Kubernetes on AWS Elastic Kubernetes Service (EKS). While Sparkler included a Kubernetes deployment configuration, it was slightly out of date with the Kubernetes version used by EKS and the latest version of Solr and ZooKeeper, so the deployment was modified and a Pull Request was submitted to the Sparkler repository.

Sparkler serves as a solution for on-demand crawling and collection of PDFs. Since it assumes no prior knowledge of the sites it crawls, it is much less efficient than individual website scrapers and it cannot easily traverse websites built around search APIs. Despite these limitations, we experimented with using Sparkler to crawl ghostscript's bug tracker and the Internal Revenue Service (IRS) website 4. Results were mixed and Sparkler was unable to return all available PDFs in a reasonable search depth, but after crawling through 71,508 pages, 311 PDFs were found and collected. Going forward, we aim to incorporate and extend related work that will allow Sparkler to more intelligently traverse links based on search relevance into our deployment.

C. Custom Crawlers

For a small number of critical sites, the team developed custom scrapers or relied on APIs to retrieve files. For example, rather than crawling every issue on ghostscript's Bugzilla issue tracker [22], the team used Bugzilla's API to query for issues that contained attachments with mime-types including the term 'application'. Further, the team relied on JIRA's API [23] to retrieve attachments from PDFBox's and Apache Tika's JIRA sites programmatically [24] [25].

IV. Extracting Features

It has been previously noted that gathering data for corpora is easier than analyzing it and generating helpful features [12]. An important driver of effective search – especially over millions of files or documents – is the availability of rich, expressive features (attributes) of the files contained in the search corpus [26]. Features providing information about a file can describe the type and structure of the file, its contents, or the results of a virus scan using existing open-source tools. Features may also be generated through rule-based encodings or through the use of natural language processing (NLP), machine learning (ML), or byte-frequency analysis approaches which could be used to generate subsequent features of interest. Part-of-speech and word-dependency tags provided from CoreNLP [27] provide the means for extracting measurements and their relations from text [28], and machine learning techniques are enabling search capabilities on scientific data [26].

We apply this thinking for the purposes of developing search over a wide reach corpus to support the development of LangSec-based parsers by using features of interest in the generation of test corpora (which reduces corpus sizes and is easier than distributing many terabytes of data [12]). As an initial proof of concept, we randomly selected 20,000 files from GovDocs1 and from our Common Crawl files in which to provide features to enable search incorporating attributes of interest to the LangSec community using the open-source tools Clam Anti-Virus and Apache Tika.

A. Clam Anti-Virus

Identifying which files in corpora are malicious is important to researchers and developers in the areas of parser and forensic tool development. Tools such as VirusTotal – which provide indications as to which files are malicious and the types of malicious threats contained within those files – are valuable tools for use in annotating documents and are leveraged here for annotating those documents contained within a corpus.

One of these tools is the open-source software Clam Anti-Virus, which supports a wide variety of file formats including PDF [29], which is often utilized as a server-side email scanner which reports the virus signatures detected within scanned files or related heuristics. The utility allows users to refresh virus signatures automatically or manually, and supports users providing their own signatures for use within the software. The utility also provides a multi-threaded daemon which may be used to integrate the utility's capabilities into various types of software. While in use across many domains - including email security, endpoint protection, and web scanning - comprehensive reports on the effectiveness of Clam anti-virus against corpora of malicious files are not known to the authors. The effectiveness of current open-source tools such as Clam Anti-Virus in detecting malicious files is not well documented, yet the annotations provided by these tools may serve as valuable information for researchers and software engineers developing LangSec based, safe parsers.

The output of tools is often accepted on the basis of the vendor or development team's reputation and formal evaluation of such tools is difficult without the availability of test data that can be shared easily [6]. In order to establish a point of comparison for the effectiveness of Clam Anti-Virus, we explored the software's capabilities using a corpus of internal user-reported abusive emails from the NASA Jet Propulsion Laboratory (JPL) for the year 2017 (n=3,115). The corpus and email files contained within – provided by the laboratory's Security Operations Center (SOC) – is annotated by expert security personnel following user submission into categories pertaining to phishing attacks, malware, extortion, and others (as shown in Table (III). Emails classified as *malware* are those

that contain either a link, hidden pixel, or attachment which downloads or installs malware used to infect the user's system or provide Trojan backdoor access.

Category	Email Count
Credential Phishing	1,319
False Positives	495
Malware	**3,115**
Phishing Training	4,186
Propaganda	273
Recon	178
Social Engineering	1,190
Spam	1,312
Unknown	122

TABLE III
THE NUMBER OF EMAIL FILES REPORTED BY USERS FOR EACH OF THE CATEGORIES DEFINED BY THE JPL SECURITY OPERATIONS CENTER (SOC).

25.55% of user-reported emails (n=3,115) in the corpus are annotated by the SOC as emails containing malicious content. Given the advertised use Clam Anti-Virus as an email server scanner, we evaluate the software's ability to detect and report the viruses and their types for files labeled as *malicious* in the context that similar annotations on our PDF corpus will be used for search and retrieval by LangSec parser researchers and developers. Using the Go programming language together with Clam Anti-Virus's multi-threaded daemon, we scan files labeled as *malicious* using the default software configuration and report whether malicious content was detected and, if so, the virus signature which was detected within the file. The types of malware detected according to the virus signatures available in Clam Anti-Virus at the time of writing are shown in Table (IV) (while not the central focus of this work, it may be noted that using Clam anti-virus in this way resulted in a scan speed of approximately 183 files per second using a 2.4Ghz Intel Core i9 chip (a single container); replication of this pipeline through containerization or other means can provide further scalability).

Virus Signature Type	Email Count
None (false negatives)	2,854
Doc.Dropper.Agent	**43**
Java.Malware.Agent	9
Xls.Dropper.Agent	7
Pdf.Dropper.Agent	5
Doc.Dropper.Downloader	3
Doc.Downloader.Jaff	2
Other Types	205

TABLE IV
THE VIRUS SIGNATURE TYPES OF MALICIOUS FILES PROVIDED FROM THE OUTPUT OF A CLAM ANTI-VIRUS SCAN.

Scanning the *malicious* files results in 274 files identified as containing malicious content by Clam Anti-Virus utility and its database of virus signatures, or 8.76% of the files in the *malicious* category. The software's support for the Portable Document Format (PDF) is noted on the website [29], and several emails with PDF attachments were labeled as malicious according to Clam anti-virus in the user-reported emails made available by the SOC. While a significant amount of false negatives are present – which needs further examination and is not a core focus of this work – the presence of detectable virus signatures and the ability to integrate Clam Anti-Virus into an annotation pipeline using the multi-threaded daemon provided enough justification to include it for annotating documents within the corpus.

As an early means of exploring the tools applicability in this domain – and with the goal of providing helpful file annotations to the LangSec community – we apply Clam Anti-Virus in its default configuration to the random selection of 20,000 PDF files from the GovDocs1 and Common Crawl datasets and report the results of the scan in Table (V). Clam Anti-Virus identified 68 malicious files in our corpus generated through random sampling (0.34%), with *Heuristics.OLE2.ContainsMacros* and *Pdf.Exploit.CVE_2017_2957* as the most frequent virus signatures (both with n=22 occurrences).

Virus Signature Type	Email Count
None	19,932
Heuristics.OLE2.ContainsMacros	**22**
Pdf.Exploit.CVE_2017_2957	**22**
Heuristics.Broken.Executable	17
Pdf.Exploit.CVE_2018_4993	2
Pdf.Exploit.CVE_2016_6948	1
Pdf.Exploit.CVE_2018_4882	1
Win.Exploit.E107-1	1
Win.Trojan.C99-15	1
Heuristics.PDF.ObfuscatedNameObject	1

TABLE V
THE VIRUS SIGNATURE TYPES OF THE RANDOM SELECTION OF 20,000 FILES PROVIDED FROM THE OUTPUT OF A CLAM ANTI-VIRUS SCAN.

While the number of virus signature detections from Clam Anti-Virus is small at 68 (0.0034% of the sample corpus), the annotations provided by Clam anti-virus may be used by researchers and developers in building LangSec based, safe parsers by searching for and retrieving files with specific types of known exploits. When scaled to many millions of documents, searching for documents with specific known exploits will allow for the generation of test corpora that meet specific requirements and for use in development and testing, with the hope that future parsers are not as vulnerable to today's known exploits. In this early work, we index the annotations provided by the Clam Anti-Virus utility and apply search and visualization capabilities as an exploration of this concept.

B. Apache Tika

The Apache Tika annotator uses a Python [30] wrapper for Apache Tika [31], a Java-based content detection and analysis framework. It is capable of detecting and extracting metadata and text from 1,400 different file types (such as PPT, HTML, PDF, JPEG and MP3). Tika finds applications in a wide range of areas such as search engine indexing [32], analysis of corpora or file contents [33] [34] and translation [35]. In addition, Solr, Drupal, Alfresco, Sparkler [21], ImageCat, DARPA

MEMEX [36] and many internal projects at the NASA Jet Propulsion Laboratory use Apache Tika for purposes related to search and content extraction.

The Tika annotator was used to extract several features from the 20,000 subset of PDF documents containing GovDocs1 and Common Crawl
(VI).

Feature	Description
Author	Author of the document
PDF version	PDF Version of the document
Digital Signature	Presence of digital cryptographic signatures
Creator Tool	Tool creating the original document
Producer	Tool converting from original format to PDF
Application Type	MIME Type
Number of Pages	Number of pages in the document
Number of Annotations	Additional objects added to a document

TABLE VI
FEATURES EXTRACTED BY TIKA ANNOTATOR ON 20,000 SUBSET OF PDF DOCUMENTS

The goal is to provide the results of this annotator to visualize the datasets and provide a targeted search of documents in the hopes of locating specific files of interest when developing LangSec-based parsers, as is the case with Clam Anti-Virus.

V. VISUALIZING FEATURES

As mentioned in the previous section, the following visualizations are based on a 20,000 file subset of GovDocs1 and Common Crawl data. The team indexed the extracted features in Elasticsearch [37] and used Kibana [38] for prototyping potential visualizations.

LangSec developers need to understand file type distributions. We intentionally selected only PDF files for our proof of concept feature extraction and visualization. However, PDF files often contain several different file types, including image files for inline rendering, font files and/or regular attachments (including other PDF files!). In Fig. 5, we show the distribution of the containers (PDFs) and all embedded files.

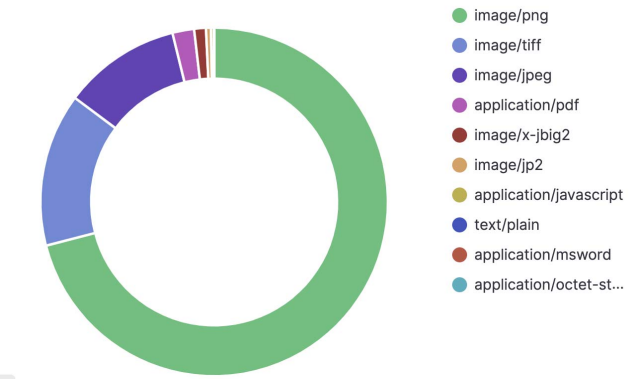

Fig. 5. Top 10 File Types for Container and Embedded Files

LangSec developers need to understand the temporal distributions of files in a corpus. This will allow them to ensure a broad range of tests of file formats through the years, as the syntax of file formats evolve through time. From an industry perspective, it can also be useful to visualize the adoption of new versions. For example, in Fig. 6, we show the distribution of PDF versions by year.

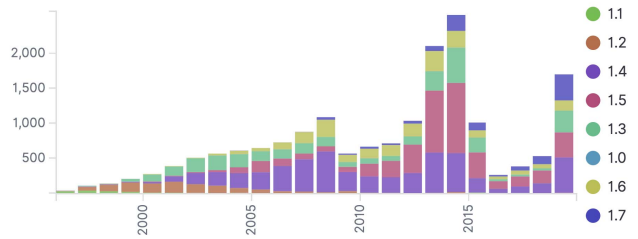

Fig. 6. PDF Version by Year

Developers and researchers also need to ensure coverage by creation software. Different software packages may implement standards differently, and it can be useful to ensure coverage of files created by the major software vendors – see, e.g. Figure 7. For forensics researchers, it can also be useful to find these characteristics to help confirm file authenticity.

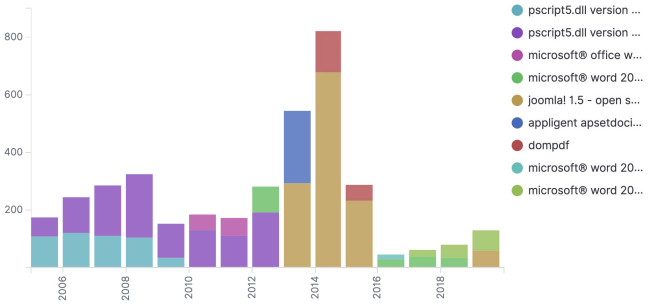

Fig. 7. Creator Tools by Year (Top 2 per Year)

Given the complexity of character set encodings and language directionality (left-to-right languages vs right-to-left languages), developers and researchers need a diversity of languages in their corpora. One of the recognized limitations of GovDocs1 is its high concentration of English language documents (see Fig. 8). CommonCrawl's PDFs offer a slightly broader distribution of languages (see Fig. 9) when compared to GovDocs1, getting closer to the prevalence of the English language's representation within content on the web (at approximately 59%) [39].

VI. FUTURE WORK

Our next steps include three primary areas:

1) increase the breadth of feature extraction – as we work with researchers and parser developers we continue to identify new features that need to be extracted and made searchable.
2) scaling – once we determine a minimum viable product in terms of feature extraction and search, we plan to scale out the indexing and search via Elasticsearch in AWS.

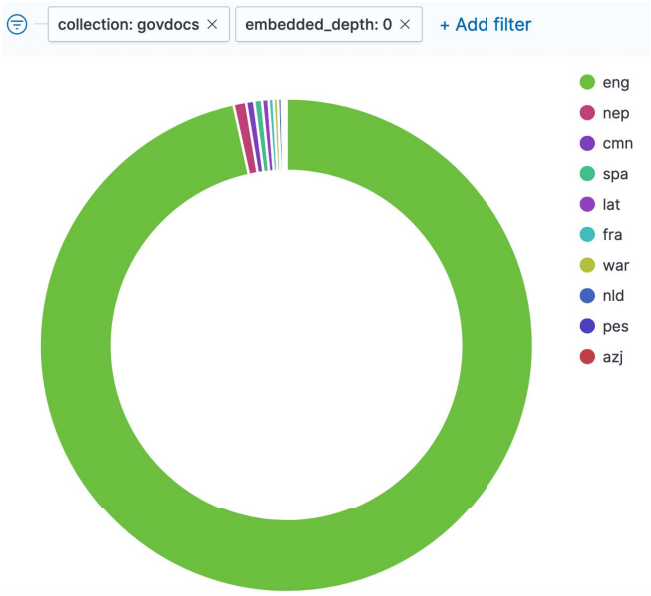

Fig. 8. Top 10 Languages in GovDocs1 PDFs

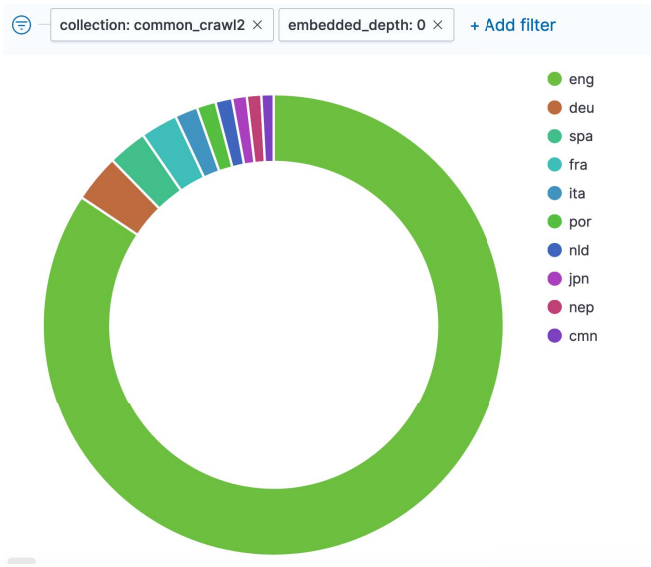

Fig. 9. Top 10 Languages in Common Crawl PDFs

3) making the data public – keeping in mind on cost, legal concerns, and other priorities, we would like to provide our toolsets and perhaps our corpus publicly, as has been the case with previous work [40] [6]. If we can work with Common Crawl or VirusTotal to include some of the features we are extracting, that could help the parser developer communities broadly.

Acknowledgments

This effort was supported in part by JPL, managed by the California Institute of Technology on behalf of NASA, and additionally in part by the DARPA Memex/XDATA/D3M/ASED/SafeDocs/LwLL/GCA programs and NSF award numbers ICER-1639753, PLR-1348450 and PLR-144562 funded a portion of the work. We acknowledge the XSEDE program and computing allocation provided by TACC on Maverick2 and Wrangler for contributing to this work. We would like to thank Peter Wyatt of the PDF Association and Dan Becker and his colleagues at Kudu Dynamics for their ongoing collegiality and collaboration on this task.

References

[1] D. Liu, H. Wang, and A. Stavrou, "Detecting malicious javascript in pdf through document instrumentation," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2014, pp. 100–111.
[2] H. V. Nath and B. M. Mehtre, "Ensemble learning for detection of malicious content embedded in pdf documents," in *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, Feb 2015, pp. 1–5.
[3] S. Ehteshamifar, A. Barresi, T. Gross, and M. Pradel, "Easy to fool? testing the anti-evasion capabilities of pdf malware scanners," 01 2019.
[4] "Announcement: Pdf attachment virus "peachy"," August 2001. [Online]. Available: https://forums.adobe.com/thread/302989
[5] J. Whitington, *PDF Explained*. O'Reilly, 2011.
[6] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," *Digital Investigation*, vol. 6, pp. S2 – S11, 2009, the Proceedings of the Ninth Annual DFRWS Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287609000346
[7] "Apache Tika," https://tika.apache.org.
[8] "Apache PDFBox," https://pdfbox.apache.org.
[9] "Apache POI," https://poi.apache.org.
[10] "Apache Tika's Regression Corpus (TIKA-1302)," https://openpreservation.org/blog/2016/10/04/apache-tikas-regression-corpus-tika-1302, 2016.
[11] "Datasets for Cyber For'ensics)," https://datasets.fbreitinger.de/datasets/, 2019.
[12] S. Garfinkel, "Lessons learned writing digital forensics tools and managing a 30tb digital evidence corpus," *Digital Investigation*, vol. 9, pp. S80 – S89, 2012, the Proceedings of the Twelfth Annual DFRWS Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287612000278
[13] V. Basile, J. Bos, K. Evang, and N. Venhuizen, "Developing a large semantically annotated corpus," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, pp. 3196–3200. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2012/pdf/534_Paper.pdf
[14] M. Wolfe, "Beyond "green buildings:" exploring the effects of jevons' paradox on the sustainability of archival practices," *Archival Science*, vol. 12, no. 1, pp. 35–50, Jul. 2011. [Online]. Available: https://doi.org/10.1007/s10502-011-9143-4
[15] "VirusTotal," https://www.virustotal.com/.
[16] "VirusTotal File Search Help," https://www.virustotal.com/intelligence/help/file-search/.
[17] C. G. R. Lavanya Pamulaparty and M. S. Rao, "A novel approach for avoiding overload in the web crawling." Odisha, India: High Performance Computing and Applications (ICHPCA), 2014.
[18] "Common Crawl," https://commoncrawl.org.
[19] "Navigating the WARC File Format," https://commoncrawl.org/2014/04/navigating-the-warc-file-format/, 2014.
[20] "Index to WARC Files and URLS in Columnar Format," https://commoncrawl.org/2018/03/index-to-warc-files-and-urls-in-columnar-format/, 2018.
[21] T. Gowda, S. Karanjeet, and C. A. Mattmann, "Sparkler - crawler on apache spark," 2017, spark Summit East. [Online]. Available: https://databricks.com/session/sparkler-crawler-on-apache-spark
[22] "Ghostscript's Bugzilla," https://bugs.ghostscript.com/.
[23] "JIRA's rest-apis," https://developer.atlassian.com/server/jira/platform/rest-apis/.
[24] "Apache PDFBox's JIRA," https://issues.apache.org/jira/projects/PDFBOX/summary.

[25] "Apache Tika's JIRA," https://issues.apache.org/jira/projects/TIKA/summary.
[26] G. P. Rodrigo, M. Henderson, G. H. Weber, C. Ophus, K. Antypas, and L. Ramakrishnan, "ScienceSearch: Enabling search through automatic metadata generation," in *2018 IEEE 14th International Conference on e-Science (e-Science)*. IEEE, Oct. 2018. [Online]. Available: https://doi.org/10.1109/escience.2018.00025
[27] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010
[28] K. Hundman and C. A. Mattmann, "Measurement context extraction from text: Discovering opportunities and gaps in earth science," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017.
[29] "About." [Online]. Available: https://www.clamav.net/about
[30] G. van Rossum, "Python tutorial, technical report cs-r9526," Tech. Rep., May 1995.
[31] C. Mattmann and J. Zitting, *Tika in Action*. USA: Manning Publications Co., 2011.
[32] O. Alhabashneh, R. Iqbal, N. Shah, S. Amin, and A. James, "Towards the development of an integrated framework for enhancing enterprise search using latent semantic indexing," in *Conceptual Structures for Discovering Knowledge*. Springer Berlin Heidelberg, 2011, pp. 346–352. [Online]. Available: https://doi.org/10.1007/978-3-642-22688-5_29
[33] A. B. Burgess and C. A. Mattmann, "Automatically classifying and interpreting polar datasets with apache tika," in *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)*, Aug 2014, pp. 863–867.
[34] A. N. Jackson, "Formats over time: Exploring uk web history," 2012.
[35] J. Tiedemann, "Improved text extraction from pdf documents for large-scale natural language processing," in *Computational Linguistics and Intelligent Text Processing*, A. Gelbukh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 102–112.
[36] T. Gowda, K. Hundman, and C. A. Mattmann, "An approach for automatic and large scale image forensics," in *Proceedings of the 2nd International Workshop on Multimedia Forensics and Security*, ser. MFSec '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 16–20. [Online]. Available: https://doi.org/10.1145/3078897.3080536
[37] "Elasticsearch," https://www.elastic.co.
[38] "Kibana," https://www.elastic.co/kibana.
[39] "Usage statistics of content languages for websites." [Online]. Available: https://w3techs.com/technologies/overview/content_language
[40] B. Klimt and Y. Yang, "Introducing the enron corpus." in *CEAS*, 2004. [Online]. Available: http://dblp.uni-trier.de/db/conf/ceas/ceas2004.html#KlimtY04

APPENDIX

A. JPL Abuse Data Malware Categories

- **Credential Phishing**: Attempts to trick the victim into providing sensitive username/password information. Usually contains a link which directs the victim to a site requesting they enter in the username/password.
- **False Positive**: A legitimate email that is mistakenly labeled as a malicious email.
- **Malware**: An email which contains either a link, hidden pixel, or attachment which downloads or installs malware used to infect the user's system or provide Trojan backdoor access.
- **Phishing Training**: Test phishing emails sent to users to measure their susceptibility for falling victim to phishing attacks.
- **Propaganda**: Email containing political, religious, or other debatable information use to spread the attackers ideas and world views.
- **Recon**: Email used to gather more information, usually to prepare for another future attack. For example, if you ever see an email from an unknown Gmail or Yahoo account with a blank subject-line or content, the email is likely a test email to determine if your email account is active and may be used for a future attack.
- **Social Engineering**: An email attempted to trick the user into responding to the email or provide sensitive information (Example: Nigerian lottery).
- **Spam**: A non-malicious marketing email.
- **Unknown**: An email that was not well categorized into any of the above categories.