Global Information Assurance Certification Paper

High Assurance File Filtering, It's Not Magic

GIAC GCIH Gold Certification

Author: Adam Gould, adamgould@outlook.com
Advisor: Johannes Ullrich
Accepted: January 20118

Template Version September 2014

Abstract

This paper examines file type identification techniques to inform further research to improve the security of cross domain solutions (CDS), which are regarded as the most reliable technologies of high-assurance file filtering solutions. Traditionally only used in highly classified government environments, CDS are slowly being adopted by other institutions in the financial, healthcare and mining sectors due to the increasing recognition of the value and importance of the protection of intellectual property (IP). The portable document format (PDF) is one of the primary document formats in which IP is shared and distributed. By using PDFs as a case study, this paper proposes recommendations specifically for software file format specification creators to develop file type sub-specifications that can be easily validated for the purposes of IP control and security. The recommendations herein will conceptually apply to all file types, although it should be noted that not all techniques and recommendations will be applicable to every file type due to unique properties that exist in different classes of file types.

1. Introduction

Commercial organisations have traditionally purchased independent technologies to combat data or network-based threats. Technical solutions that provide a high degree of assurance across both network and data attack threats are uncommon in the commercial sector. Within government, the use of high-assurance devices that seek to mitigate both data and network-based threats have been heavily implemented for decades in the form of Cross Domain Solutions (CDS). Among the numerous file filtering solutions available on the market today**,** CDS are considered the most robust and secure. A CDS is a form of controlled interface that allows the manual and/or automatic access and/or transfer information between different security domains. CDS form a boundary with a set of mechanisms that enforce security policies and control the flow of information between security domains (Instruction, 2015).

CDS are categorised in two logical types: Access CDS and Transfer CDS. Access CDS provide the user with a window to view and manipulate information residing in multiple security domains from a single device (Smith, 2015). The information is constrained to the security domain from where it originates and cannot be transferred to another security domain. Transfer CDS, on the other hand, facilitate the transfer of information between different security domains (Smith, 2015). Common examples include the transfer of email messages, documents, videos as well as the bulk synchronisation of database or information repositories. Transfer CDS conduct inspection, verification or redaction of information transferred between security domains to ensure that the transfer is undertaken in accordance with the specified security policy.

In the data-rich world in which government agencies, along with public and private organisations operate, the need to share information is critical. While Access CDS will always have their place, Transfer CDS have greater capacity to facilitate the controlled sharing of information. Their applicability to new and diverse use cases will continue to be identified. This is due to the fact that Transfer CDS technologies perform three key high-level security functions: they prevent leakage of information, they defend against both data and network-based threats, and they maintain separation between security domains (Caddick, 2016). Transfer CDS capabilities include but are not limited

to file filtering, keyword analysis, integrity checks, transliteration, sanitisation, cleansing, regrading of files, and malicious code detection.

CDS are becoming more widespread as private, public and government organisations are seeking highly reliable and robust file filtering technologies to respond to data and network-based threats. Information Technology (IT) security professionals are also increasingly being required to provide assurance that the technologies being implemented are effectively mitigating the associated risks. Information security technologies have traditionally countered either data or network-based threats. As technology and capability have evolved, it is increasingly common for products to attempt to mitigate both data and network-based threats, however, the extent to which each class of threats is mitigated against is proportionate to the capabilities of the specific technology considered. In the case of data based threats, government, private and public organisations have implemented network-based technologies in an effort to contain information to specific security domains. However, technological solutions to mitigate data based threats through the identification and verification of information within various file types is relatively immature. CDS technologies, which are the focus of the present paper, have and continue to be considered the most reliable and secure file filtering solution. In fact, Government bodies such as the Australian Signals Directorate (ASD) mandate the use of CDS when connecting TOP SECRET or SECRET security domains to lower classified security domains (Directorate, 2017). However, it should be noted that the core file type filters within CDS have likely not been updated in a number of years. As such, the improvement of the security capabilities of CDS must address the current deficiencies that likely exist in all file filtering technologies when presented with complex and ever-evolving file type formats, including PDF.

The necessity of such improvements is particularly evident when the harm caused by compromise of sensitive information is considered. Instances of organisations security controls being breached are occurring with greater frequency. One of the primary factors attributed to the severity of the breach is the scale of access to and loss of IP. It is estimated that the Group of Twenty (G20) economies have lost 2.5 million jobs to counterfeiting and piracy and that governments and consumers lose up to $125 billion to

cybercrime annually (Hathaway et al., 2015). In the United States (U.S) alone, the annual economic impact of intellectual property (IP) theft is estimated at $300 billion, which corresponds to 1 percent of the country's GDP (Hathaway et al., 2015). To respond to such threats, in 2017 alone, government, industry and defence spending in information security products and services will reach $86.4 billion worldwide, an increase of 7% over 2016, with spending expected to grow to $93 billion in 2018 (Gartner Inc., 2016). Gartner reports that in 2014 within the information security market space, the data loss prevention (DLP) segment recorded the fastest growth at 18.9 percent (Gartner, 2014). This trend has continued with organisations with some form of DLP already implemented which seek additional integration to enhance their DLP capabilities. Specific technologies being sought include integrated data classification, data masking and data discovery (Gartner Inc., 2016).

The demand of highly reliable and robust file filtering technologies has seen large multinational defence, security, and aerospace companies such as Forcepoint and BAE Systems diversify their existing product lines, currently restricted to Defence environments, and develop commercial-Off-The-Shelf (COTS) CDS not subject to Government control.

The necessity to guard and control the movement of IP is therefore critical, and the use of CDS in this sense is particularly necessary to monitor and secure the movement of information into environments of lesser security. As such, CDS play a crucial role in detecting threats through the inspection of both file type structure and content. In what follows, this paper will address the features and limitations of current file filtering CDS technologies and their impact on file filtering. It will do so by illustrating the security issues of CDS certification schemes and file identification methods. Secondly, it will provide recommendations for the design and adoption of standard file type specifications that can be easily validated for the purposes of IP control and security in CDS. In both the assessment of CDS technologies and in the case studies supporting the recommendations, the PDF file type specification will be adopted as the case study of reference.

Adam Gould, adamgould@outlook.com

2. PDF files and file type- related threats affecting CDS certification

The PDF format is one of the most used file type formats, particularly in the sharing of IP, as it provides a large array of features that can satisfy many uses cases across all sectors of business. Due to its flexibility, it allows organisations and creators to seamlessly store and integrate their sensitive content in text, vector, video and audio formats within one single file. Furthermore, from an end-user perspective, the PDF format offers a multitude of advantages over other popular document formats:

- They are universally displayable: PDF is a universal file format that preserves the fonts, images, graphics, and layout of any source document regardless of the application and platform used to create it.
- They are quick and easy to create: whether working with Microsoft Word, Excel or PowerPoint, documents are easily converted into the PDF format.
- There are many free PDF readers available: most PDF readers are free to the public. This ensures that anyone receiving a PDF file will be able to view it.

At the same time, however, the PDF file format consistently ranks as one of the most common file formats to be leveraged by cyber adversaries, with over 50% of all targeted phishing attacks recorded between 2009 and 2010 incorporating a PDF document in some way (Danchev, 2011). From a cyber adversarial and information security perspective, the PDF file format possesses many properties that make the file format favourable for the distribution of malware and software exploitation. These include:

- Recognisable and trusted file format: the ubiquitous use of PDF file format has resulted in greater acceptance of the PDF file format by the public.
- Well documented format: Due to the well-documented nature of PDF the file format, it is easier for malware authors to understand.
- Extensibility: The PDF file format is extensible and can use other technologies such as JavaScript, Flash, and different compression and encoding methods.

Adam Gould, adamgould@outlook.com

- Prevalence of PDF readers: Due to the popularity of the PDF file format, PDF viewers are built into most operating systems by default including all the most popular web browsers (Selvaraj & Gutierrez, 2010).

The broad feature set and extensibility of the PDF file format present unique challenges to the capacity of CDS in providing an adequate level of security. In particular, the testing and validation requirements make the certification process of these systems more complex, as well as time and resource intensive. As previously stated, CDS are intended to provide a high level of assurance as to their ability to interpret and verify files being filtered by the device. To do so, unlike other boundary control interfaces such as firewalls and proxies, firewall CDS are required to undergo a certification process prior to being connected to the interfacing security domains. The certification seeks to validate the CDS vendors' security claims to a defined degree of assurance which may include; the secure failure of componentry or the enforcement of dataflow directionality in addition to the people, processes and systems/applications involved (Welke, 2011). In Australia, Government agencies implementing CDS must reference applicable bodies such as the ASD, the Evaluated Products List (EPL) or the internationally recognised Common Criteria Scheme (CCS). Products evaluated under these schemes are assessed by competent and independent licensed laboratories. Supporting documents guide the Common Criteria certification process to define how the criteria and evaluation methods are applied when certifying specific technologies or components thereof ("Common Criteria," 2017). It should be noted, however, that certification schemes including the CCS do not provide an absolute guarantee of security. From a file filtering perspective, five key issues have an impact on the degree of assurance provided by the certification process. These include: environmental constraints, the inability to upgrade or enhance certified products without invalidating the certification, and the lack of clarity regarding which specific versions of each file type were validated through the certification process. These issues are expanded upon below and are particularly evident when the specificities of the PDF file type outlined above is considered.

1. The environmental conditions under which the file type has been assessed may not align with the consumers intended use case.

Firstly, the scope under which products are evaluated is tightly constrained (Aizuddin, 2001). The Protection Profile describes a set of requirements with the objective of countering specific threats within the context of the tightly constrained environment. For example, a product intended to operate within a highly secure and controlled environment would be required to mitigate a different set of threats than a product intended to operate within an untrusted environment such as the Internet.

These constraints define the scope, which in turn, impact the time and cost required to undertake certification. If the scope of the certification is too broad, there is a risk for the vendor to carry unnecessary expenses, which will not provide a sufficient return on investment. If, conversely, the scope is too narrow, the vendor risks failing to meet the requirements of their target customer base ("Budgeting for Common Criteria-Avoid Cost Creep," 2013). The environmental constraints under which each product is assessed may also not align with every organisation, and therefore the assurance benefits gained through the use a certified product may not be realised.

For example, when assessing the suitability of CDS specifically, it is not uncommon for a Transfer CDS to be certified to transfer data in a single direction. When transferring data from a trusted security domain to a security domain of lesser trust, the primary threats relate to the ability of the file type to be used to facilitate the exfiltration of information from the trusted security domain. Techniques may include using white text on white background or embedding an image outside of the printable page size of a PDF document. In the case of transferring content from security domain of lesser trusted to a security domain of higher trust, the primary threats relate to preventing the introduction of malicious content into the trusted security domain. Techniques may include the use of JavaScript or an embedded executable content within a PDF document. In each case, a CDS may only be designed to mitigate specific threats that relate to one of these broad objectives; therefore the use of CDS outside of the scope of certification may not only be ineffective but may also provide a false sense of security.

Adam Gould, adamgould@outlook.com

A potential solution to reduce the risk of using of products outside of the scope of certification could be the implementation of greater clarity in testing processes, specifically the file types, their version and the respective features that were evaluated in the certification process. This should be supported through the development of a commonly accepted dataset of file types that emulate the threats that a CDS would be expected to mitigate. As it will be discussed later in this document, the breadth of threats to be considered could be reduced through the adoption of tightly constrained file type sub-standards that provide only the mandatory features required.

2. The CCS discourages file filter enhancements that may be required to address changes introduced in new versions of supported file types, as deviation from the certified baseline invalidates the certification.

Secondly, the CCS certification process is known to typically require anywhere from 14 months to two years to complete. From the moment the product is submitted for assessment to the certification body, it is highly likely that new versions of existing file types may be developed, or that component software may require updating. Whilst it is commonly accepted within industry that the benefits gained through the application of security updates is greater than the potential risk posed by/from deviating from the certified baseline, this acceptance does not apply to changes or updates that impact fundamental elements of the product. Fundamental changes and updates could include new filtering methods to address changes introduced in newer versions of an existing file types or the introduction of a new capabilities such as remote administration. As a result, the invalidation of the certification may prevent vendors from enhancing their certified products as rapidly as their commercial product offerings. Furthermore, certification gained under the CCS has an indefinite validity period unless the product is altered, or the certification is withdrawn. The implication of this is that there may be a false level of assurance that a certified product is adequate to address the threats that exist in the current day, which are also bound to increase as the threat landscape changes and evolves in subsequent years after the certification was awarded. The PDF file type specification has evolved considerably since its release in 1992. Through several iterations, the PDF

file format has evolved from a basic file format to incorporate complex features such as interactive elements, support for embedded content including video, imagery and binary file type formats as well as support for JavaScript and Extensible Markup Language (XML) to name a few.

In recognition of the importance of fostering an environment where evolution and enhancements are embraced, the Common Criteria Recognition Arrangement Management Committee passed a motion to limit the validity period of CCS certification to five years, however these measures will not become effective until June 1st, 2019 (Criteria, Arrangement, Committee, & Statement, 2017). A more effective way to reduce the overhead associated with re-certification could be the support of a limited subset of file type versions or features. As it will be illustrated in this research, these could be combined with a modular approach to the software development of file type filters enabling change/updates to be constrained to a subset of code within the product.

3. Vendors do not articulate the specific threats that their CDS file filters mitigate based on the unique risks associated with each file type and the evolution of filetype versions.

Thirdly, the effectiveness of a CDS is dependent upon its ability to parse, inspect and understand the file type and content being transferred. In practical terms, this is addressed through the application of discrete file filter mitigations. File filtering technologies including CDS claim to facilitate the secure transfer of numerous file types; however, not all file filter mitigations can be applied to all file types. In the case of PDF files, for example, a file filter may have the ability to search text with the PDF document to identify the presence of sensitive information using a keyword search; however, a keyword search would ineffective when applied directly against a video file type format. As a result, there will be differing degrees of assurance that the file is safe or appropriate for transfer based on file type and associated filter mitigations that can be applied to the file. In the case of PDF files which may contain embedded content that cannot be reliably be programmatically verified using a file filter, the current mitigation method is to introduce a human element to manually inspect and authorise each file prior to transfer to

the destination security domain. However, human verification is primarily limited to evaluating the content of the files, such as images and text within a document. The assessment of a file structure or content not displayed through a traditional file viewer is not undertaken as the tools that are required to do so are not incorporated into CDS (Knott, 2002). Furthermore, the accuracy and the degree of assurance that can be gained through the human review process are directly proportional to the reviewer's ability to understand and contextualise the subject matter within the document or image. As it will be discussed below, a potential solution to this problem could be the use of tightly constrained file type formats to increase the filter's effectiveness. Increasing file filter effectiveness would reduce the quantity of files required to be manually verified, therefore increasing the accuracy of the human review process.

The issue of filetype testing is made more complex when different versions of the same filetype are considered. Although CDS are generally designed to support specific file types, it is not clear which specific versions of each file type are tested and evaluated when assessing the effectiveness of the file filters. As file specifications evolve, it is possible that changes in the file structure or the content of features may impact the ability of a file filtering component to correctly parse and therefore, inspect and understand the file being transferred. For example, as mentioned above, the PDF specification has evolved considerably from the initial PDF version 1.0 through to PDF version 1.7 (this is the current version at the time of writing). As the PDF specification has evolved, new features that may have a security impact have been added. These include, but are not limited to, the ability to embed external links in PDF 1.1 and support for extended Unicode character sets and digital signatures in PDF 1.3. As a result, it is possible that the file filter may no longer remain effective if it is not kept up to date with logic required to parse newer versions of a file type. A potential solution to prevent variance in file structure and content is the development of file types that are designed with the intent of providing long-term compatibility combined with a static well-defined feature set. This would require the file type to incorporate all the features that would reasonably be required by the user base for the desired period for which compatibility is desired.

Adam Gould, adamgould@outlook.com

4. Tolerances that exist within file type interpreters may prevent the identification of malformed files that do not conform to the file type specification.

For a file filter to correctly parse a file, the file filter is required to interpret and understand the file structure as defined in the associated file type specification. Not only is writing file type parses complicated and time intensive, but it is also error-prone. The PDF 1.7 file type specification is over 1300 pages alone, and it specifically states that the specification does not clearly identify methods for validating the conformance of PDF files or interpreters. In the absence of clear and well-defined boundaries, specifications are open to misinterpretation. To assist in reducing risk associated with misinterpretation, in addition to software development costs, it is common for software developers to obtain existing software libraries which can be integrated into their software applications that provide support to interpret and understand a file type out of the box. For example, the Adobe PDF Library offers complete functionality for generating, manipulating, rendering, and printing Adobe PDF documents. The Library also provides Adobe-supported implementation of the latest PDF specification ("Adobe PDF Library SDK," n.d.). While these software libraries provide savings in cost, as well as time and risk, they may not always be suitable for all use cases, such as security, where a system may need to deal with malicious or malformed content.

To cater to backward as well as forward compatibility, the PDF specification encourages software developers to build tolerances into their PDF interpreters in order to fail silently when unsupported content is detected. A common use case may be opening a PDF document that conforms to a newer version of the PDF specification in an older PDF viewer. While the intent of incorporating tolerances is positive in that it improves the end user experience, it may negatively impact the ability to perform strict file type validation. Failure to apply robust logic or insufficient validation and verification within software applications has been identified as a common cause of vulnerabilities in information systems (Anton, Anderson et al., 2004). It is common for a file to render successfully even when the file has been altered outside of the parameters of the associated file type specification. The degree of tolerance also varies considerably between different filetype interpreters, as file tolerance is often proportionate to filetype

complexity. As will be discussed in greater detail in the Recommendations section of this paper, file complexity, and therefore, tolerance can be both reduced by using less complex file types.

5. CDS may fail to mitigate known threats due to the absence of a commonly adopted file type dataset intended for the purpose of developing highly robust and secure file filtering solutions.

A final issue is the lack of consistency with testing practices and guidelines. Typically, each vendor will conduct their development and testing processes against a self-sourced file type dataset. There is no evidence which determines whether this dataset is representative of the file type and that the features within are being used in the target consumer environments. Furthermore, in the case for security enforcing products such as file filters, there are no commonly adopted file type datasets that emulate the threats against which software developers can assess the robustness and security of their products. While the malware research community maintains repositories of malware for the purposes of research and analysis, these repositories are not suitable as an appropriate file type filter dataset due to their malicious nature and unknown composition. Malicious files should serve as a source to inform the development of an appropriate dataset with clearly defined test cases that are accurate, measurable, safe and can be undertaken in repeatable manner. As previously stated, PDF files are commonly used by cyber adversaries in targeted attacks. Many of the features incorporated into PDF file type specification such as JavaScript that are commonly abused by cyber adversaries may not be required in most organisational use cases. A possible solution to this issue may be the development of constrained file type sub-standards that provide only the mandatory features required in exchange for an increased level of security and control of IP. The constraint of the version and features reduces the attack-surface and subsequent file type test cases that must be developed and catered for.

Security issues related to file type identification therefore, must regard both the file type, version, feature-set and threat environment in which such files are used. Improvements to CDS should consequently seek to develop systems that enable

independent assessment of system sub-components in order to reduce certification timeframes and promote modular sub-component enhancements in order to maintain currency with changes in file type specifications. Solutions must also take into consideration the specific threats related to distinct file types being verified by these systems. Going forward, a holistic approach is required that takes into consideration not only technical considerations related to each file type, but also the trade-off between filetype feature functionality and security.

3. Recommendations and case studies

Current responses to these issues are rather limited, as research on file format complexities and on the implications on file filtering and analysis technologies is scarce. Recent contributions such as research conducted by Ange Albertini have focussed heavily on file format abuse techniques, however the context of the assessment is limited to end-user applications such as Adobe PDF Reader and WinZip (Albertini, 2014). Little information is also available on the practical solutions to these issues and how these challenges can be addressed by file filtering technologies. At the time of writing, file filtering solutions that undertake strict validation of file types do not exist. However, it is recognised that the accuracy in determining a files type and its version is only possible when parsing the entire file to determine that all of its content is compliant with the specification (Fletcher, 2016). To reduce the complexity of parsing and validating an entire file, some file filtering solutions seek to reduce the complexity by rewriting the contents of the file to neuter or remove aspects of the file that may pose a security risk. This process may take two forms: it may rewrite the file by extracting and reusing only those components that can be satisfactorily verified; or, it may transform the file into an alternate format, whereby the risks introduced by the original format are not present in the transformed format.

Taking a blunt force approach by rewriting or transforming every file without consideration of its compliance with security policy has its drawbacks. These may include the loss of content and the inability to obtain information to inform the risk equation or identifying those that seek to intentionally or unintentionally circumvent

security policy. This lack of insight in CDS technology may be related to the fact that specific details pertaining to how CDS undertake file type identification is not publicly available. More research on the theoretical and practical aspects of file type identification should be undertaken, particularly in cases where the accuracy of the detection method is critical to informing a security-based decision. The author of this paper estimates however that it is highly likely that these products use one or more of the common file type identification methods in use in all modern software applications today.

There are three main file identification methods: filename extension, Multipurpose Internet Mail Extensions (MIME) media type identification, and magic number identification. Filename extension identification and MIME media type identification are considered low assurance identification methods due to their inability to discriminate between different file type formats that may use the same filename extension. In addition to this, file extensions are user definable and can be altered at the user's discretion or not even used at all. The third method, magic (magic against galloping internet complexity) number identification is considered the most robust method of file type identification method. A magic number refers to invariant data in a file type format that can be used to form a signature to uniquely identify a file type. These data may consist of an individual or series of identifiers in the form of a series of bytes or a string of text that enables the unique identification of a file type. A small subset of file types has been fortunate to reserve meaningful identifiers; these usually exist in the first few bytes of a file. For example, the PDF file type format uses %PDF- in US-ASCII encoding within the first 1024 bytes of a PDF file whereas the GIF87a format uses the hexadecimal number 0x474946383761, which when converted into US-ASCII is GIF87a (Underwood, 2009). The file command which incorporates the magic number file identification method is probably the most widely used tool for file type identification. The tests contained in the associated magic file have been and remain the largest repository of file signatures in the world (Underwood, 2009).

In order for a file filter to be considered robust and secure, it must categorically be able to identify the file type to ensure that the correct file filter mitigations are applied. Furthermore, these mitigations must be consistently applied and sufficient to mitigate

against known threats. The difficulty of satisfying these requirements is proportionate to the complexity and variance in the file type structure and content required to be verified by the file filter. This complexity is generally quantified through assessment of the file type specification in addition to a real-world analysis to determine how the specification has been implemented in practice. As previously stated, the PDF file type specification specifically states that the specification does not clearly identify methods for validating the conformance of PDF files or interpreters. Therefore, in the absence of clear and well-defined boundaries, specifications are open to misinterpretation. This results in different implementations of the specification, which in turn provide cyber adversaries opportunity manipulate tolerances within the PDF file type structure. Furthermore, the ability of file filters to accurately inspect and verify PDF documents is also negatively impacted.

Improvements to the technology should thus address the leverage of tolerances that exist within PDF file type interpreters, in order to prevent cyber adversaries from circumventing file filtering mitigations by influencing both the identification and content of the PDF file type. The proposed solutions intend to address these issues by; creating tightly constrained file type formats or sub-formats, consistently enforcing the specification of file signature at offset zero for all file type specifications and rewriting known good components of the existing file types into a file type capable of being verified for conformance with security policy. Each of these solutions is designed to address the recognition of diversity in PDF file type versions that may be in use within the consumer environment. Such solutions should also be particularly relevant to the PDF file type due to the PDF file type specification requiring both backwards and forwards compatibility with all PDF versions at a given major version number.

Recommendation One: The creation of tightly constrained file type formats or sub-formats that incorporate only the mandatory features required.

Through the creation of tightly constrained file type formats that incorporate only the mandatory features required, specification developers will be able to eliminate tolerances within the specification, therefore, removing and reducing risk of misinterpretation. This will in turn, reduce complexity and variance in the

implementation of the specification enabling software developers to create robust and secure file filters for the purposes of security policy enforcement and control of IP. In the case of widely- used file formats, such as PDF, this can be achieved by identifying a sub-specification that already has several constraints that make the specification desirable from a security perspective. A suitable sub-specification of the PDF file format for this purpose is the PDF/A sub-specification, as PDF/A is a family of constrained forms of Adobe PDF intended to be suitable for long-term preservation of PDF documents and is already being used in practice today.

The PDF/A specification is based on PDF version 1.4 and is intended to strike a balance between currency and features. PDF/A was endorsed by the International Organization for Standardization in 2005 and later reviewed and in 2015. It attempts to maximise device independence, self-containment and self-documentation and has several constraints that make the standard desirable from a security perspective. These include the removal of support from audio, video, JavaScript, encryption and the launching of executable files (Morrissey, 2012). The reduced feature-set and complexity of the specification increases the likelihood of the development of highly reliable and tightly constrained file filter capabilities. The PDF/A standard is 29 pages in length, which is almost one-tenth of the initial release the PDF 1.0 file type specification, introduced in 1992, which was comprised of a modest feature set with the entire specification totalling only 230 pages. Its specification count is also a fraction of the current PDF 1.7 file type, which is over 1300 pages and is a far more complex and feature-rich file format with support for text and binary elements, JavaScript, interactive content, embedding rich media and audio formats, to name a few. Table 2 provides a chronological map of the PDF file format evolution over the 25 years since its development, including the PDF/A sub-specification recommended in this paper. In addition, the next major release, PDF file type specification version 2.0, was released in July of 2017. As this research was conducted only some short months later, the most current PDF version in use is PDF file type specification. Additionally, widely supported outside of Adobe's own paid commercial product offerings, is PDF version 1.7.

Table 2 PDF Features Overview

PDF Specification	Year of Release	Specification Page Count	Pertinent Features Introduced in PDF Version
PDF 1.0	1992	230	• Initial release of PDF file type format
PDF 1.1	1994	302	• Ability to link to external document or universal resource locators (URL's) • Password protection support • Support for notes within PDF document • Device independent colour support
PDF 1.2	1996	394	• Support for active forms (AcroForm) • Support for Unicode extended character sets • Support for interactive page elements
PDF 1.3	1999	518	• Support for annotations within PDF document • Additional colour space support • Support for digital signatures • Support for JavaScript • Support for 40-bit and 56-bit RC4 encryption
PDF 1.4 **(basis for PDF/A)**	2001	978	• Ability to selectively hide content in a PDF using inbuilt options • Improved support for JavaScript and enhanced integration with databases • Support for JBIG2 compression • Support for 128-bit RC4 encryption
PDF 1.5	2003	1172	• Improved compression techniques including object streams and JPEG 2000 compression • Support for layers within PDF documents • Support for XML Forms Architecture
PDF/A-1 **(based on PDF 1.4)**	2005	29	• Audio and video content are forbidden • Javascript and executable file launches are prohibited • All fonts must be embedded • Encryption is disallowed • Use of standards-based metadata is mandated
PDF 1.6	2005	1236	• Support for AES encryption • Support to selectively encrypt object • Support to directly imbed OpenType fonts • Ability to specify size object should be printed regardless of page size • Additional support for embedded file attachments including cross-document linking to and from embedded objects
PDF 1.7	2006	1310	• Enhancement to enable support for up to two passwords to decrypt and display contents of document based on permissions • Support for additional constraints regarding the certificate to be used when signing. These include digest methods and revocation checking

**(It should be noted that there are over twenty-two subversions of the PDF file type specification which include a subset of features supported within each minor version of the specification. These were not included in this table.)

Adam Gould, adamgould@outlook.com

Broad compatibility has many advantages, however, in cases where a product utilising the PDF file type format only requires features at a given PDF file type version, there is little incentive for product vendors to upgrade their PDF file type software libraries to the most current version.

To quantify the typical usage of PDF file type by version, a sample set of 2000 PDF documents was obtained from a representative Australian government department (Table 3). The sample set was further refined to include only PDF files with a file creation date between May 2001 and the current day. May 2001 coincides with the release of version 1.4 of PDF file type specification. This limitation was imposed since PDF version 1.4 incorporates all the core features of the PDF file type specification and therefore should satisfy most consumer use cases to the current day. It is expected that PDF version 1.4 would be a suitable baseline to which the average consumer could conceivably update their files to achieve a sufficient balance between currency and features. The results infer that many software applications in use today generate PDF files using various versions of the PDF filetype specification.

Table 3 PDF Count by Version

PDF Specification	Year of Release	Document Count
PDF 1.0	1992	Nil
PDF 1.1	1994	10
PDF 1.2	1996	20
PDF 1.3	1999	200
PDF 1.4	2001	243
PDF 1.5	2003	168
PDF 1.6	2005	281
PDF 1.7	2006	110

As previously discussed, the CCS discourages product enhancement due to the invalidation of the certification. As a result, it is possible that PDF software libraries utilised within certified products may only be as current as the time that the product was certified. The implications of this may be that a file filter utilising a PDF software library

prior to the version of PDF file type being validated by the file filter may fail to identify features or content associated with a later version of the specification. For example, a file filter using a PDF software library compliant with the PDF 1.2 specification may fail to identify the presence of JavaScript introduced in version 1.3 of the PDF specification. The risk of failure to identify features or content associated with a given version of the PDF specification not only relates to the currency of the file filter PDF software library, but also to the specification against which the file is assessed. The PDF file type specification requires that PDF files are rendered in accordance with the PDF file type specification declared within the PDF file header. The PDF file type specification requires that the PDF version number be declared within the first 127 bytes of the PDF file header. However, version 1.4 of the PDF specification introduced a feature that enables the PDF version declared within the header to be overridden through the declaration of an alternate PDF version within the PDF document catalogue. This amendment to the specification is intended to provide the ability to incrementally update a PDF document at a later point in time without rewriting the PDF file in its entirety.

From a file filtering perspective of developing highly reliable and secure file filtering solutions, the loose tolerances and broad compatibility required of the PDF file type specification may introduce a risk of security policy being bypassed or the inability of security policy to be consistently enforced. At a high level, these may include:

- Failures to apply security policy due to the inability to identify features associated with later versions of the PDF specification that is not supported and therefore silently ignored by a legacy PDF interpreter within the file filter;
- In continuation of the previous example, compromise of a destination system through the abuse of features silently ignored by the file filter but interpreted correctly by the destination system due to the use of a current PDF file type interpreter;
- The ability to alter the rendering of the PDF file through the declaration of the PDF file type specification in both the PDF file type format header and the document catalog, therefore altering the content that is inspected by the file filter in cases where the file filter utilising a PDF software library prior to version 1.4;

Adam Gould, adamgould@outlook.com

- In continuation of the previous example, compromise of a destination system through the specification of a later PDF version within the PDF document catalog resulting in the PDF rendering content that was not visible to the file filter.

The correct inspection of a file is only possible with a robust understanding of syntactic structure and semantic meaning of the file (Fletcher, 2016). As said above, the PDF file type format incorporates a broad array of features that may be abused to circumvent the application of security policy. A cursory review of Mitre's Corporation's Common Vulnerabilities and Exposures (CVE) list identified 568 vulnerabilities related to various features that exist within the PDF file type format that have been catalogues since the CVE Index was established in 1999 (CVE Board, 2017). Through the development of tightly constrained file type formats or sub-formats suitable for the purpose of security policy enforcement and control of IP, there is a higher likelihood that the complexity of the problem can be reduced in order to develop robust and secure file type interpreters and therefore file filters. A suitable sub-format of the PDF file type is the PDF/A sub-format, as PDF/A is a family of constrained forms of Adobe PDF intended to be suitable for long-term preservation of PDF documents and is already being used in practice today. PDF/A attempts to maximise device independence, self-containment and self-documentation and, has several constraints that make the standard desirable from a security perspective. These include the removal of support from audio, video, JavaScript, encryption and the launching of executable files (Morrissey, 2012). The PDF/A standard is 29 pages in length. The reduced feature-set and complexity of the specification increases the likelihood of the development of highly reliable and tightly constrained file filter capabilities. The following case study will illustrate this point further by demonstrating how tolerances within file type specifications, specifically the magic number can be abused to facilitate the miscatergorisation of a file type in order to bypass the application of security policy. Delving deeper into the PDF file type format, it examines the PDF file type structure itself and the implications that the complexity of the standard poses not only to file filtering capabilitie,s but also file type identification as well.

Adam Gould, adamgould@outlook.com

Recommendation Two: Consistently enforce the specification of file signature at offset zero for all file types

Magic number verification is currently considered the most robust method of a file type identification. In practice, however, the specification of magic number at multiple offsets across differing file type specifications introduces tolerances that can be abused to evade or alter the categorisation of a filetype. The enforcement of the magic number at offset zero, if implemented consistently, will eliminate the capability of a cyber adversary to abuse the file signatures for the purposes of miscatergorisation in order to bypass the application of security policy. In the first case study illustrates how the broad array of features available within the PDF specification may enable a cyber adversary to bypass or reduce the reliability of the application of security policy. This threat leverages the tolerances that exist in the respective file type specification combined with inconsistencies in the implementation of the specification. The following case study will demonstrate a basic manipulation that can be made to the PDF file type structure, specifically the header, to alter the categorisation of the file type in order to reduce or bypass the application of security policy. In order to mitigate against this type of manipulation being successfully used, it is recommended that file type specification developers consistently specify the file signature as offset zero within the file.

For the purposes of creating a PDF file with only the mandatory components required to meet the specification, the PDF filetype structure may be split into three parts: the header, body, and trailer. The header is the first line of a PDF file type structure; it identifies the version of the PDF specification with which the PDF document should conform. Typically, the first test applied by the file utility would be to inspect the file header to identify invariant data referred to as the magic number. The syntax of the PDF header comprises of a percentage symbol, the letters P,D,F followed by a hyphen and the PDF specification version number. Valid PDF versions at the time of writing are PDF 1.0 through PDF 1.7 (Figure 1).

```
1 %PDF-1.1
```

Adam Gould, adamgould@outlook.com

Figure 1 - PDF File Type Header

The body begins at line 2. A PDF document can be regarded as a hierarchy of objects contained in the body of the PDF file type structure. The PDF specification supports eight basic objects, which are defined in the PDF file type structure using the obj keyword, the object number and, the generation number respectively. Basic objects include; Boolean values, integer and real numbers, strings, names, arrays, dictionaries, streams and null objects. The object number uniquely identifies the object within the PDF file format structure, and the generation number provides a reference to track changes to an object throughout the life of the object. Changes to objects may include modifying the PDF content such as adding text or images or removing content. All objects should begin their life with a generation number of zero.

Object 1, beginning at line 2 of the example PDF, specifies the document catalog (Figure 2). The document catalog contains references to other objects within the PDF file type format structure. The document catalog may contain a variety of objects that specify items such as PDF reader preferences, such as defaulting to full-screen view or a complete reference to all other objects within the PDF file type format structure. In its most simplistic form as used in this case study, the document catalog only contains a single reference to a page within the PDF file type format structure as denoted by Pages keyword. The number following the Pages keyword refers to another object 2 within the PDF file type structure where further details that relate to the page are located. The number following the Pages keyword indirectly identifies the object that contains the information followed by the generation number. The R following the generation number is used to denote an indirect reference.

```
2  1 0 obj
3  <<
4  %   /Type /Catalog
5      /Pages 2 0 R
6  >>
7  endobj
```

Figure 2 - PDF Document Catalog

Adam Gould, adamgould@outlook.com

Object 2, beginning at line 9, specifies the pages within the PDF document (Figure 3). The PDF contains a single page as denoted by the Count keyword. The Kids keyword is used to reference further objects that the resources, specifically the font, form the page. Object 3, beginning at line 16, specifies the resources within the page that includes a reference to object 4 that contains the page content as denoted by the Contents keyword. In addition to the content, object 3 also specifies the font applied to resources in object 4 as denoted by the Font keyword. Object 4, beginning at line 33 contains a stream object containing text. The BT and ET keywords demote where the text object begins (Begin Text) and ends (End Text). Within the text object, there are further keywords to specify the font size (100) in addition to the text (Hello SANS!).

```
 9  2 0 obj
10  <<
11      /Type /Pages
12      /Count 1
13      /Kids [ 3 0 R ]
14  >>
15  endobj
16  3 0 obj
17  <<
18      /Type /Page
19      /Contents 4 0 R
20      /Parent 2 0 R
21      /Resources <<
22          /Font <<
23              /F1 <<
24                  /Type /Font
25                  /Subtype /Type1
26                  /BaseFont /Arial
27              >>
28          >>
29      >>
30  >>
31  endobj
32
33  4 0 obj
34  << >>
35  stream
36  BT
37  /F1 100
38  Tf 1 1 1 1 1 0
39  Tr(Hello SANS!)Tj
40  ET
41  endstream
42  endobj
```

Figure 3 - PDF Object Body

Adam Gould, adamgould@outlook.com

The file trailer, beginning at line 46, provides the PDF interpreter with a known location to obtain a pointer to the document catalog. In most cases, the trailer will include reference to the cross-reference table denoted by the xref keyword. The cross-reference table contains a listing of all objects within the PDF file type structure and supports the random access of objects within the PDF document without the interpreter having to scan through the entire PDF file structure. As the cross-reference table is not mandatory, the trailer within this example PDF file type structure refers to the document catalog at object 1.

Figure 4 File Trailer

```
43 trailer
44 <<
45     /Root 1 0 R
46 >>
```

The tests applied by the file utility are contained within a tests file. Each file type supported by the file utility will have a unique test file referred to as the magic file (Figure 5). Each test within the magic file is specified using the following four fields:

1. The offset from the beginning of the file where the test should begin. The initial location within the file is at byte 0;
2. The type of data to match which include byte, short, long quad, float, double, regular expression or string values;
3. The value or pattern to be matched;
4. The message to be output if the match is successful.

("magic(5) - Linux Man Page," 2008)

```
 1 # $File: pdf,v 1.6 2009/09/19 16:28:11
 2 # pdf:  file(1) magic for Portable Document Format
 3 #
 4
 5 0       string          %PDF-           PDF document
 6 !:mime  application/pdf
 7 >5      byte            x               \b, version %c
 8 >7      byte            x               \b.%c
 9
10 # Forms Data Format
11 0       string          %FDF-           FDF document
12 >5      byte            x               \b, version %c
13 >7      byte            x               \b.%c
```

Figure 5 - PDF Magic File. Note, the PDF magic file above was obtained from the current stable release of the file utility as available in CentOS Linux at the time of writing. The last update to the tests contained within the PDF magic file was the 19th of September 2009.

Reviewing the contents of the magic file associated with the PDF file type reveals that the file contains two independent series of tests. The first series of tests apply to the PDF file type and the second series of tests apply to the FDF (Form Data Format) file type, both of which are developed by Adobe Systems. The tests within the various magic files often contain a hierarchy of tests to deduce the correct file type. These tests may also solicit specific information relating to the file to be used in the message output of the file utility such as version information or comments within the file metadata. Additional tests are introduced by one or more right-angle brackets (>) preceding the offset within the file to indicate from where the match will be made (Figure 5). The number of right-angle brackets used on each line indicates the level of the test; a line without a right-angle bracket at the beginning is at level zero test. Tests are arranged in a tree-like hierarchy: If the test on a line at level n succeeds, all following tests at level n+1 are performed, and the messages printed if the tests succeed, until a line with level n (or less) appears ("magic(5) - Linux Man Page," 2008).

The following list explains the tests performed for PDF file type format as illustrated in Figure 5 above;

1. Attempt to match the string "%PDF-" beginning at offset zero within the file
2. Compare the mime media type against the value "application/pdf"
3. If the previous test succeeds, extract the values at byte 6 and
4. Extract the value of byte 8. These two values identify the stated PDF file type specification with which the file states it conforms. The output is subsequently printed to the screen.

Running the file utility against the example PDF, to detect the file type and PDF filetype version, and subsequently running the file utility a second time to display the mime media type, confirms the file has been successfully identified as a PDF (Figure 6).

```
[root@centos7 ~]# file helloworld-generic.pdf
helloworld-generic.pdf: PDF document, version 1.1

[root@centos7 ~]# file --mime helloworld-generic.pdf
helloworld-generic.pdf: application/pdf; charset=us-ascii
```

Figure 6 - File Conformant PDF

With an understanding of how the PDF file type is categorised by the file utility, the case study will now examine a basic manipulation that can be applied to a PDF file type structure to impact the accuracy and reliability of the tests applied by the file utility in a way that is advantageous for a cyber adversary looking to circumvent security policy.

To gain a full understanding of how these tests indeed verify the contents of the PDF file, it is helpful to view the example PDF file using a hexadecimal editor. When reviewing the first line of the output, the presence of the string, %PDF- and the version number 1.1 are clearly visible.

```
[root@centos7 ~]# hexdump -c helloworld-generic.pdf
0000000   %   P   D   F   -   1   .   1  \n  \n   1       0       o   b
0000010   j  \n   <   <  \n   %  \t   /   T   y   p   e       /   C   a
0000020   t   a   l   o   g  \n  \t   /   P   a   g   e   s       2
0000030   0       R  \n   >   >  \n   e   n   d   o   b   j  \n  \n   2
0000040       0       o   b   j  \n   <   <  \n  \t   /   T   y   p   e
0000050       /   P   a   g   e   s  \n  \t   /   C   o   u   n   t
0000060   1  \n  \t   /   K   i   d   s       [       3       0       R
0000070       ]  \n   >   >  \n   e   n   d   o   b   j  \n  \n   3
0000080   0       o   b   j  \n   <   <  \n  \t   /   T   y   p   e
0000090   /   P   a   g   e  \n  \t   /   C   o   n   t   e   n   t   s
00000a0       4       0       R  \n  \t   /   P   a   r   e   n   t
00000b0   2       0       R  \n  \t   /   R   e   s   o   u   r   c   e
00000c0   s       <   <  \n  \t  \t   /   F   o   n   t       <   <  \n
00000d0  \t  \t  \t   /   F   1       <   <  \n  \t  \t  \t  \t   /   T
00000e0   y   p   e       /   F   o   n   t  \n  \t  \t  \t  \t   /   S
00000f0   u   b   t   y   p   e       /   T   y   p   e   1  \n  \t  \t
0000100  \t  \t   /   B   a   s   e   F   o   n   t       /   A   r   i
0000110   a   l  \n  \t  \t  \t   >   >  \n  \t  \t   >   >  \n  \t   >
0000120   >  \n   >   >  \n   e   n   d   o   b   j  \n  \n   4       0
0000130       o   b   j  \n   <   <       >   >  \n   s   t   r   e   a
0000140   m  \n   B   T  \n   /   F   1       1   0   0  \n   T   f
0000150   1       1       1       1       1       0  \n   T   r   (   H
0000160   e   l   l   o       W   o   r   l   d   !   )   T   j  \n   E
0000170   T  \n   e   n   d   s   t   r   e   a   m  \n   e   n   d   o
0000180   b   j  \n  \n   t   r   a   i   l   e   r  \n   <   <  \n  \t
0000190   /   R   o   o   t       1       0       R  \n   >   >  \n
000019f
```

Figure 7 - Hexdump Conformant PDF

Of the four tests within the magic file, only first test references invariant data and is therefore considered reliable. A closer look at the test identifies that the match is being made beginning at offset zero within the file. The PDF specification, however, allows for

the PDF header to present itself anywhere within the first 127 bytes of the file. The following screenshot illustrates the impact on the file utilities' ability to categorise the PDF file by simply amending the PDF file type structure to incorporate a space prior to the string "%PDF-" resulting in the string beginning at offset one.

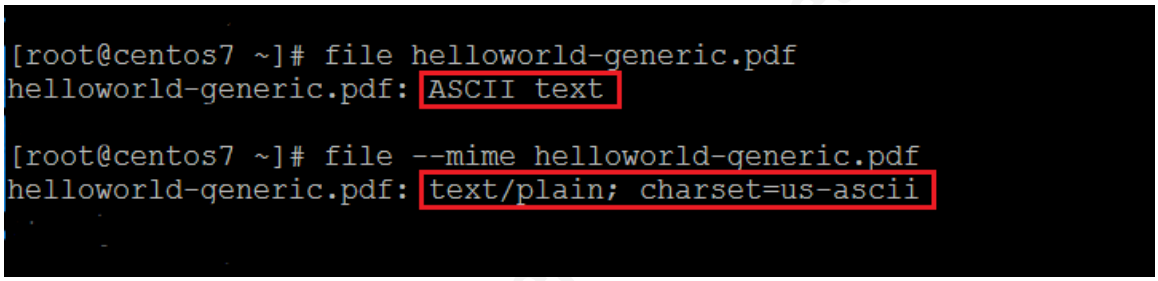

Figure 8 - File Manipulated PDF

As displayed above, the file is now incorrectly categorised as ASCII text. From the perspective of assessing risk related to each unique file type, the text file type is viewed as one of the safest file type formats. This is due to the absence of support for security-sensitive features such as interactive content or JavaScript. Therefore, it is not uncommon for the text files type to be exempt from security policy such as virus scanning. The text file type, however, is still commonly subject to tests to identify keywords within the text itself for the presence of IP that may be a contravention with security policy.

Magic number verification is currently considered the most robust method of a file type identification. In practice, however, as illustrated in case study two and three, the specification of magic number at multiple offsets between differing file type specifications introduces tolerances that can be abused to evade or alter the categorisation of a filetype. The miscatergoisation of a file can have catastrophic consequences relating to the failure to apply the appropriate mitigation to render the file safe for transfer to the destination security domain. The enforcement of the magic number at offset zero, if implemented consistently, will eliminate the capability to abuse the file signatures for the purposes of miscatergorisation as long as the content and the offset of the file type signatures is strictly enforced by the interpreting applications including file filters. The following case study examines a more complex and interesting example of file type

structure abuse. As such, the associated recommendation seeks to avoid this added complexity by applying an alternate mitigation of transliteration.

Recommendation Three: Rewriting known good components of the existing file types into a file type capable of being verified for conformance with security policy.

In cases where file types are too complex, evolve too rapidly or cannot be created natively in the desired file type format and version, the transliteration function is recommended. Transliteration extracts strictly defined components from the original file and writes the contents to new file in conformance with desired file type specification. This function is beneficial from a security perspective as only known good content is extracted and transferred to the destination security domain eliminating all other content, which may include malicious content that may exist within the original file. Building upon the previous case study on the manipulation of a PDF filetype to alter the file utilities categorisation, there is opportunity to look at the more extreme example of polyglot files. A Polyglot file is a file that is multiple different file types simultaneously (Morrissey, 2012). Polyglot files are the exemplar of the real-world implications of the implementation of file specifications without clear and well-defined boundaries. Because standards are often unclear, incomplete, or difficult to understand, a variety of file type abuse methods are possible (Morrissey, 2012).

Polyglot files are created through the merging of file types that can work in conjunction with one another to be combined into a single file structure with each file type capable of being interpreted independently. In some cases, it is possible to combine multiple file types whilst remaining conformant with each respective specification, while in other cases tolerances that exist within interpreting applications will enable the files to be rendered successfully despite not achieving compliance with respective file type specification. This case study examines a PDF, Joint Photographic Experts Group (JPEG), ZIP polyglot file. The example polyglot file, zipjpg.pdf used in this case study has been developed by Ange Albertini, it is available at his GitHub repository, Corkami. In order to understand how a file can assume different file types simultaneously, it is helpful to realise that a file has no intrinsic meaning. The meaning of a file – its type, its

validity and its contents can be different for each parser or interpreter (Albertini, 2014). The screenshots below (Figure 9) illustrate zipjpg.pdf being successfully interpreted as a valid PDF, JPEG and ZIP file respectively.

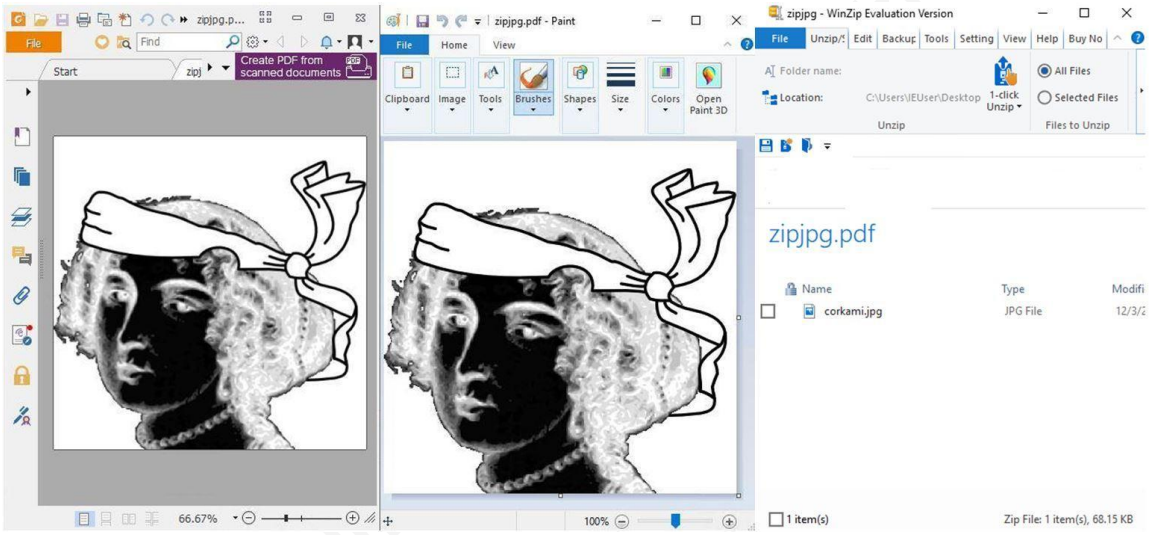

Figure 9 - PDF JPEG ZIP Polyglot File

Tolerances that can be exploited within the file header and the data offset for the respective file types provide the framework to undertake this type of abuse. These two structural components provide the flexibility to include multiple file headers or combine independent or common data portions to be made available to each file types. The successful merger of multiple file types also requires that the respective file type interpreters either ignore or are unaware of content relating to the other file types. Due to differences in the interpretation, implementation, and enforcement of file type specifications between different software products is not uncommon for polyglot files to be rendered successfully in one interpreter while failing to open correctly or at all in another (Albertini, 2014).

Before delving into the file structure is it is advantageous to determine the file type categorization of zipjpg.pdf using the file utility. This specific example uses Albertini's image corkami.jpg as a common data portion for each respective filetype. The screenshot below illustrates that the polyglot file, zipjpg.pdf is categorised as a JPEG image data file which is part of the JPEG File Interchange Format (JFIF) file type specification.

Adam Gould, adamgould@outlook.com

```
[root@centos7 ~]# file zipjpg.pdf
zipjpg.pdf: JPEG image data, JFIF standard 1.01, comment: "%PDF-1.4"

[root@centos7 ~]# file --mime zipjpg.pdf
zipjpg.pdf: image/jpeg; charset=binary
```

Figure 10 - File Polyglot PDF, JPEG, ZIP

To understand why this is the case, the following illustration by Albertini (Figure 11) provides a breakdown of the file structure illustrating the pertinent file components that have been combined and then manipulated to create a PDF, JPEG, ZIP polyglot file.

Offset	Content 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	JPEG	PDF	ZIP
00000:	FF D8	magic		
00002:	FF E0 00 10 .J .F .I .F 00 01 01 01 00 48 00 48 00 00	header		
00014:	FF FE 02 1F	comment segment start (length)		
00018:	%PDF-1.4 1 0 obj ...		PDF header & document	
00140:	20 0 obj «/Length 69786» stream		dummy object start	
00168:	.P .K 03 04			local file header start
00181:	00 9B			filename length
00186:	endstream endobj 5 0 obj «/Width 400 ... stream		dummy object end image object start	lfh's filename (abused)
00221:	FF D8 FF E0 00 10 .J .F .I .F 00 01 01 01 00 48 00 48 00 00	(end of comment)	image header	stored file data
00235:	FF DB 00 43 ...	image data (DQT)	—	—
112B5:	FF D9	end of image	—	—
112B7:	FF FE 00 E6	segment comment start (not strictly req.)		
112BC:	endstream endobj 24 0 obj stream ...		end of image object dummy object start	
112DE:	.P .K 01 00			central directory
1130C:	corkami.jpg			filename (correct)
11317:	.P .K 05 06			end of central directory
1132B:	75 00			length of comment
1132E:	endstream endobj xref ...		end of dummy object xref, trailer	archive comment
1139A:	%%EOF %		end of file line comment	
113A1:	FF D9	end of image marker	(end of line)	(end of comment)

Figure 11 - PDF, JPEG, ZIP File Structure (Albertini, 2014).

The first step in creating a polyglot file is to determine which file type can coexist within a single file structure. This is achieved by identifying the constraints placed on each file type by the respective specification. In some cases, these constraints can be bypassed due to tolerances that may exist within the respective file type interpreters. The PDF file type specification requires the PDF signature, "%PDF-" to be present within the first 1024 bytes of the file, the ZIP specification only requires that the ZIP signature, ".P

.K 3 4" be present anywhere within the file. This enables the JPEG signature, "FF D8" to be used at offset zero in accordance with the JFIF specification.

With the requirements of the file type headers addressed, the next step is to determine how the body of the file structure can be formed. Working with the JPEG specification there is capacity within the specification to incorporate comments. As these comments will not interfere with the rendering of the JPEG file, this is a suitable location within which to house the PDF header. Fortunately, the PDF specification dictates that the PDF file only be rendered from the point that PDF signature is identified, therefore the JPEG header is safely ignored by PDF interpreters (Albertini, 2014). The PDF, ZIP and JPEG headers can be referenced at offset 00018, 00168 and 00000 respectively in illustration contained within Figure 11.

To address the data portion of the respective file types, the PDF file type specification will tolerate the presence of incomplete objects. An incomplete object is created by declaring an object using the 'obj' keyword, as illustrated on offset 00140, and subsequently failing to close the object by failing to specify 'endobj' keyword. Incomplete objects will be recognised by the PDF interpreter but silently ignored, thus preventing an error due to a malformed file structure. This is a likely side effect of maintaining both backwards and forwards compatibility. The creation of an incomplete object is used to contain the common image datastream referenced by both the ZIP and JPEG files. The incomplete object ends at offset 00186, followed by an image object which is required to render the image for the PDF file type. This incomplete object is ignored. Finally, moving into the trailer components of the file structure, it is common-place for file interprets to also ignore all content beyond the respective files types end of file (EOF) marker. This enables each file to structure to end without interfering with one another. It should also be noted that consistently enforcing the specification of the file signature at offset zero for all file types would prevent the creation of polyglot files.

Furthermore, in the case of file types that are too complex or burdensome to be reliably verified programatically, the alternate approach of transliteration is recommended. Transliteration extracts strictly defined known good components from the original file and writes the contents to new file in conformance with desired file type

specification. Transliteration is beneficial from a security perspective as unknown content, which may include malicious content that may exist within the original file, is eliminated and therefore not transferred into the destination security domain. This solution has the additional benefit of a wide array of differing file types that may be ingested and transformed into a single desired filetype generated as a consistent and safe output. At the same time, however, this approach may result in the loss of content and therefore may be undesirable in some use cases. Nevertheless, it may be a reasonable interim mitigation until more tightly constrained native file formats such as PDF/A are widely supported.

Conclusions

As illustrated in the first case study, the PDF file type specification has evolved considerably since its initial release in 1992. As such, there is a large variance in PDF specification versions in use. Taking into consideration the backwards and forwards compatibility requirement of the PDF file format it is likely that this variance will be a longstanding issue that needs to be addressed. The PDF/A specification has been endorsed for over twelve years; however, tools that facilitate the creation, conversion, and validation of PDF/A documents are not widely implemented. The absence of appropriate tools may be addressed by incorporating a transformation capability with file filtering solutions including CDS. In cases where the tools to create native PDF/A files exist, capability should be established to extract known good components from the original PDF file and write the contents to new PDF document in conformance with desired PDF/A specification. This approach has the additional benefit of enabling alternate non-PDF document formats to be ingested and transformed, resulting in a single file type which can then be verified in accordance with security policy. Conversely, this approach may result in the loss of content and therefore may be undesirable from the end user perspective however it may be a reasonable interim mitigation until more tightly constrained native file formats and, the associated tools are commonplace.

Adam Gould, adamgould@outlook.com

In light of this research, information security professionals assisting organisations in better understanding and reducing their data-attack risk must address many complementary factors. These include, but are not limited to:

- The accurate identification of PDF file type versions and associated features in use to inform the threats that may apply, given the use of a specific feature within a given version of the PDF file type.
- The environment in which the file filter will operate. The likelihood of organisations being exposed to malicious PDF files is highly dependent on the environment from which the PDF files originate. For example, a PDF file sourced from an unvetted public forum on the Internet is far more likely to contain malicious code than a PDF file sourced from a government partner in a well-controlled and maintained network environment.
- The version of the PDF file type interpreter used within the file filter For the file filter to correctly interpret and verify the contents of the PDF file type there is a requirement that the PDF file be rendered using a PDF interpreter at the same or later version than the PDF file itself.
- The tolerances that exist within a specified PDF file interpreter. As shown above, to retain both backwards and forwards compatibility, the PDF file type speciation requires that features introduced in newer versions be silently ignore by interpreters conforming with a prior PDF specification.

Currently, it is not commonplace for product vendors to address explicitly which versions of the PDF file type specification are supported by their products. Also, not addressed is the clear articulation of the threats and associated environmental constraints that must be considered when determining whether a file filter product adequately addresses the potential threats that may exist within the consumers environment. As a result, the complexity of addressing all the possibilities that exist within the PDF file type holistically across all versions of the specification remains a complex and challenging task.

Adam Gould, adamgould@outlook.com

References

Adobe PDF Library SDK. (n.d.). Retrieved from http://www.adobe.com/devnet/pdf/library.html

Aizuddin, A. (2001). The Common Criteria ISO/IEC 15408 - The Insight, Some Thoughts, Questions and, Issues. SANS about Common Criteria, (Cc).

Albertini, A. (2014). Let 's Talk About Analogies Between Animals and File Formats. Retrieved from https://www.slideshare.net/ange4771/lets-talk-about-41659294

Anton, P. S., Anderson, R. H., Mesic, R., & Scheiern, M. (2004). Finding and Fixing Vulnerabilities in Information Systems: The Vulnerability Assessment and Mitigation Methodology.

Australian Government IP Australia. (2017). Australian Intellectual Property Report 2017. Retrieved from http://www.ipaustralia.gov.au/uploaded-files/reports/intellectual-property-report-2014-low-res.pdf

Australian System of National Accounts, 2014-15. (2015).

Budgeting for Common Criteria- Avoid Cost Creep. (2013).

Caddick, E. M. (2016). CDS Concepts and Definitions, (16).

Common Criteria. (2017). Retrieved from http://www.commoncriteriaportal.org/

Criteria, C., Arrangement, R., Committee, M., & Statement, V. (2017). Common Criteria Recognition Arrangement Common Criteria Management Committee Vision Statement.

Danchev, D. (2011). Malicious PDF files becoming the attack vector of choice. Retrieved from http://www.zdnet.com/article/report-malicious-pdf-files-becoming-the-attack-vector-of-choice/

Fletcher, B. (2016). Preventing Victim Zero with Advanced Content Filtering (pp. 1–41). Retrieved from http://www.gitec.org/sites/default/files/7-BoydFletcher.pdf

Gartner. (2014). Gartner Says Worldwide Information Security Spending Will Grow Almost 8 Percent in 2014 as Organizations Become More Threat-Aware. Retrieved from https://www.gartner.com/newsroom/id/2828722

Adam Gould, adamgould@outlook.com

Gartner Inc. (2016). Gartner Says Worldwide Information Security Spending Will Grow 7.9 Percent to Reach $81.6 Billion in 2016. Retrieved from http://www.gartner.com/newsroom/id/3404817

Hathaway, M., Demchak, C., Kerben, J., Mcardle, J., & Spidalieri, F. (2015). Cyber Readiness Index 2.0, (November), 48. Retrieved from http://www.potomacinstitute.org/images/CRIndex2.0.pdf

Instruction, C. (2015). Committee on National Security Systems Glossary. Retrieved from https://www.cnss.gov/CNSS/openDoc.cfm?rgZKeNrt5/AjJwThVadsiQ==

magic(5) - Linux Man Page. (2008). Retrieved from https://linux.die.net/man/5/magic

Morrissey, S. M. (2012). PDF/A, PDF for Long-term Preservation. Retrieved from http://www.loc.gov/preservation/digital/formats/fdd/fdd000318.shtml

Selvaraj, N., & Gutierrez, N. (2010). The rise of pdf malware. Symantec Security Response. Retrieved from http://www.govdefenders.com/sites/default/files/The Rise of PDF Malware.pdf

Smith, S. D. (2015). Shedding Light on Cross Domain Solutions. SANS Reading Room, (November). Retrieved from https://www.sans.org/reading-room/whitepapers/dlp/shedding-light-cross-domain-solutions-36492

Statistics, Australian B. of. (2015). International Trade in Goods and Services, Australia, Jun 2016 - 5368. Retrieved from http://www.abs.gov.au/AUSSTATS/abs@.nsf/Previousproducts/5368.0Main Features1Jun 2016?opendocument&tabname=Summary&prodno=5368.0&issue=Jun 2016&num=&view=

Underwood, W. (2009). Extensions of the UNIX File Command and Magic File for File Type Identification, (September). Retrieved from http://perpos.gtri.gatech.edu/publications/TR 09-02.pdf

Welke, S. (2011). What Federal Agency Customers Don't Know about CDS But Should. Retrieved from https://raytheontrustedcomputersolutionsblog.wordpress.com/2011/08/16/what-federal-agency-customers-don't-know-about-cds-ca…but-should/

Adam Gould, adamgould@outlook.com

CVE Board. (2017). Retrieved from Common Vulnerabilities and Exposures: https://cve.mitre.org/community/board/

Adam Gould, adamgould@outlook.com