PDF association

Towards a more secure future

Peter Wyatt
CTO, PDF Association & Principal Investigator, SafeDocs

SafeDocs

- *"The goal of the SafeDocs program is to dramatically improve software's ability to detect and reject invalid or maliciously crafted input data, without impacting the key functionality of new and existing electronic data formats.*
- *SafeDocs seeks to create technological assurance that an electronic document or message is automatically checked and safe to open, while also generating safer document formats that are subsets of current, untrustworthy versions."*

https://www.darpa.mil/program/safe-documents

Top 25 Most Dangerous Software Weaknesses

- **#4: Improper input validation**
  - #1: Out of bounds write
  - #3: Out of bounds read
- Score = prevalence and severity

Rank	ID	Name	Score	2020 Rank Change
[1]	CWE-787	Out-of-bounds Write	65.93	+1
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	CWE-125	Out-of-bounds Read	24.9	+1
[4]	CWE-20	Improper Input Validation	20.47	-1
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	+5
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	CWE-416	Use After Free	16.83	+1
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	+4
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type	8.45	+5
[11]	CWE-306	Missing Authentication for Critical Function	7.93	+13
[12]	CWE-190	Integer Overflow or Wraparound	7.12	-1
[13]	CWE-502	Deserialization of Untrusted Data	6.71	+8
[14]	CWE-287	Improper Authentication	6.58	0
[15]	CWE-476	NULL Pointer Dereference	6.54	-2
[16]	CWE-798	Use of Hard-coded Credentials	6.27	+4
[17]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	5.84	-12
[18]	CWE-862	Missing Authorization	5.47	+7
[19]	CWE-276	Incorrect Default Permissions	5.09	+22
[20]	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	4.74	-13
[21]	CWE-522	Insufficiently Protected Credentials	4.21	-3
[22]	CWE-732	Incorrect Permission Assignment for Critical Resource	4.2	-6
[23]	CWE-611	Improper Restriction of XML External Entity Reference	4.02	-4
[24]	CWE-918	Server-Side Request Forgery (SSRF)	3.78	+3
[25]	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	3.58	+6

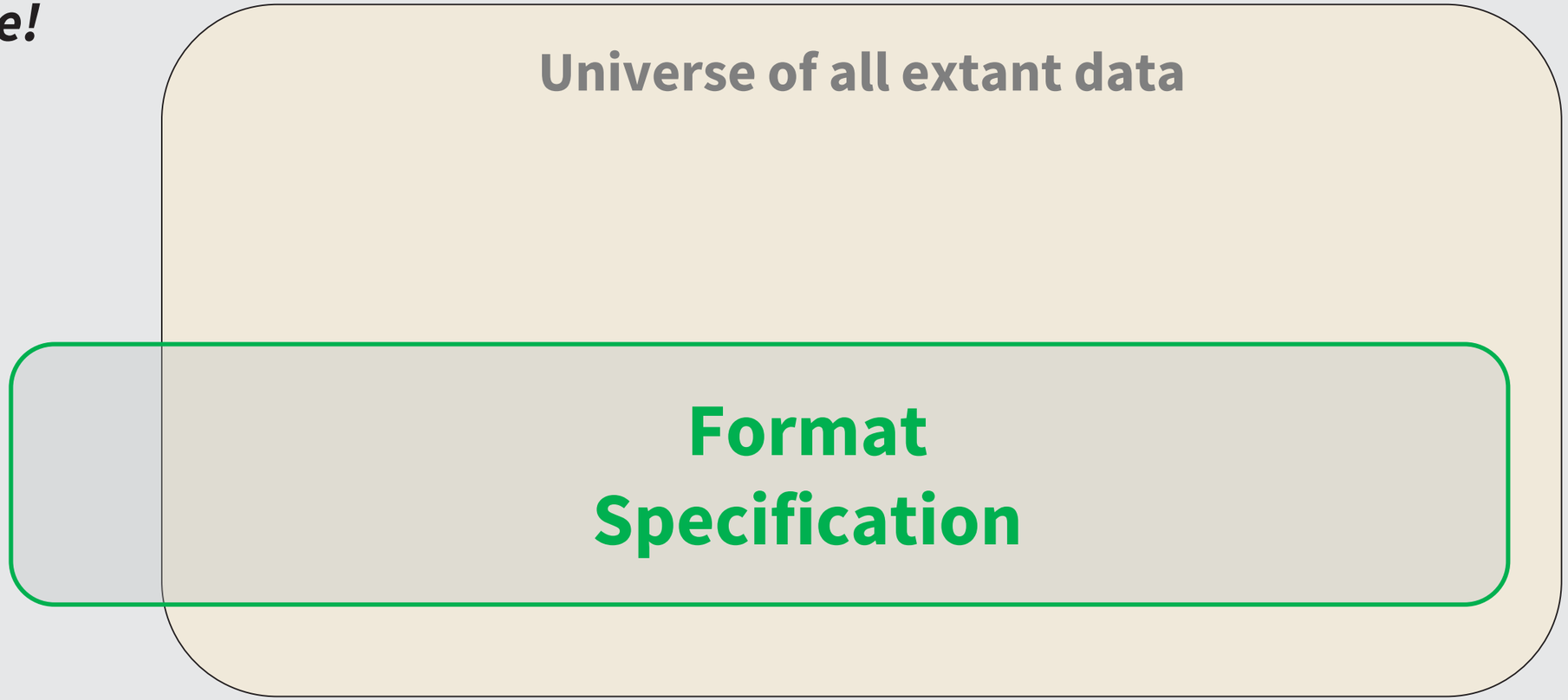
https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html

Specifications and essential truths

Format
Specification

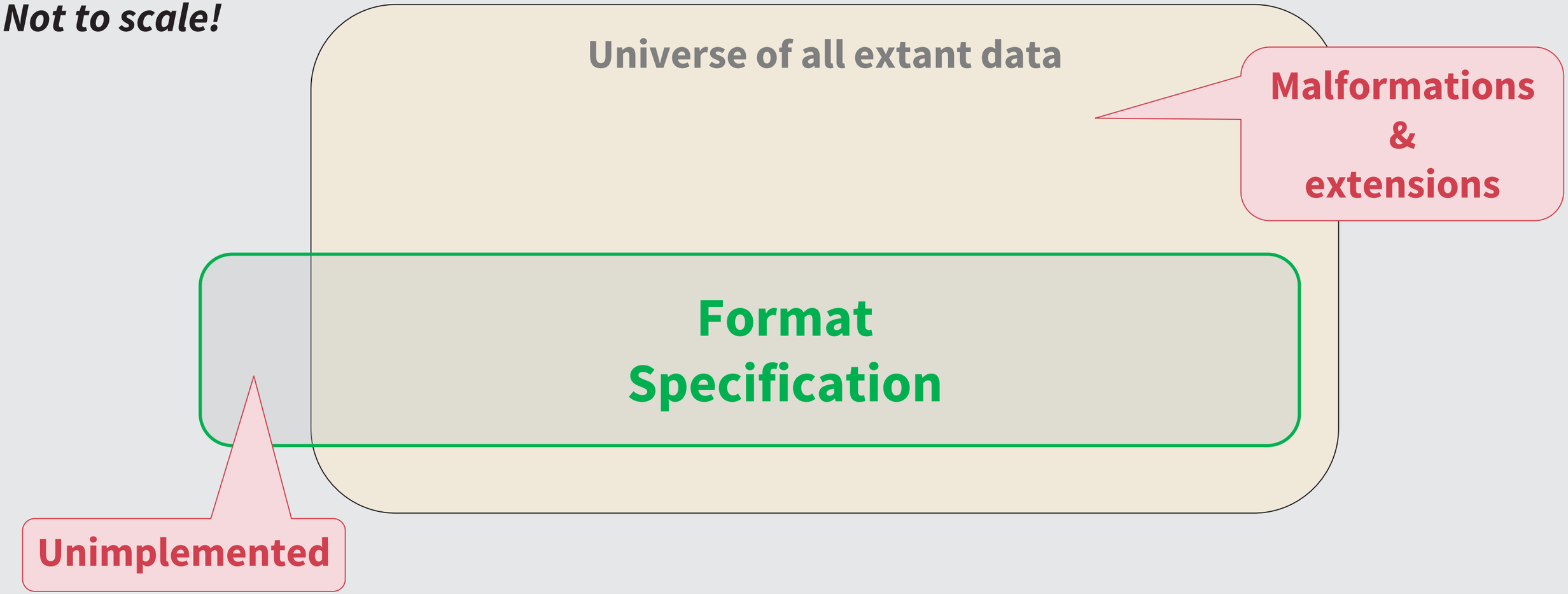
Specifications and essential truths

Not to scale!



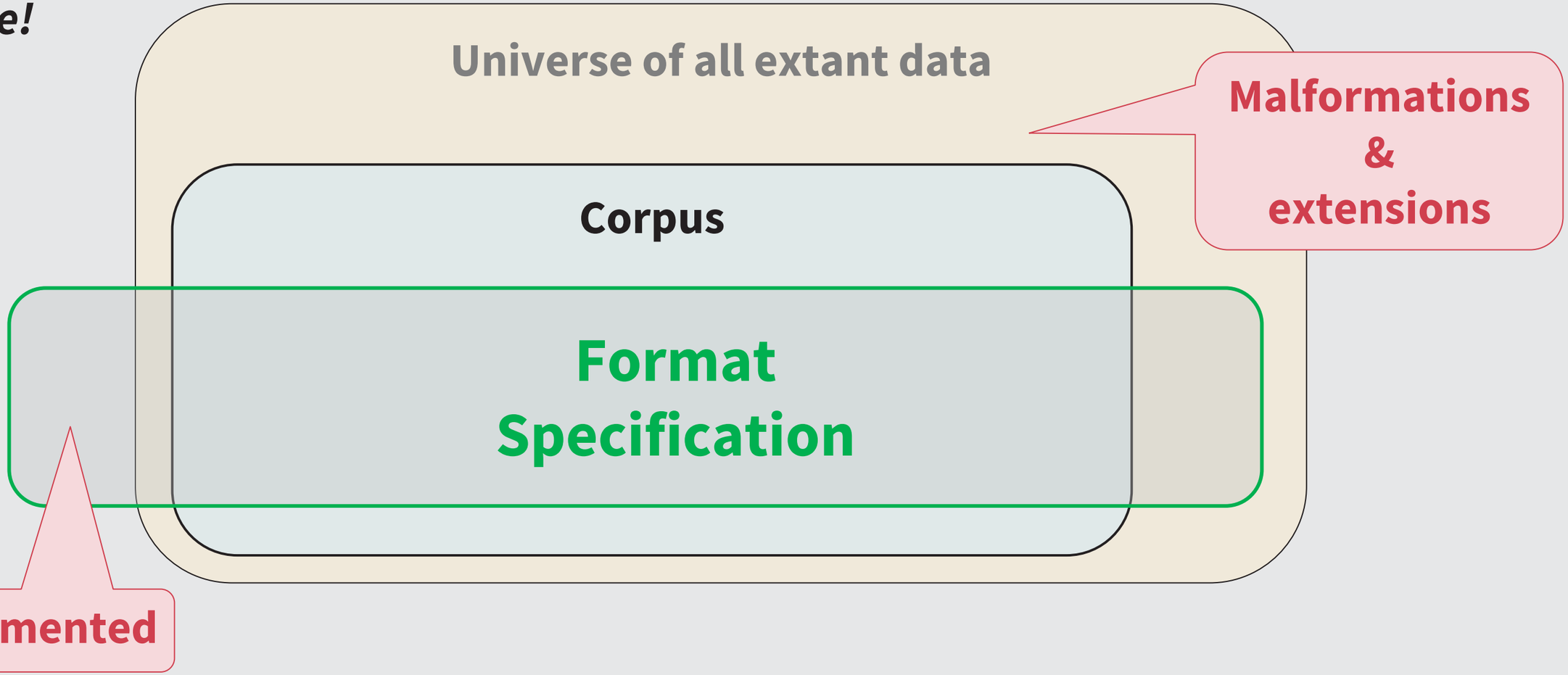
Specifications and essential truths

Not to scale!



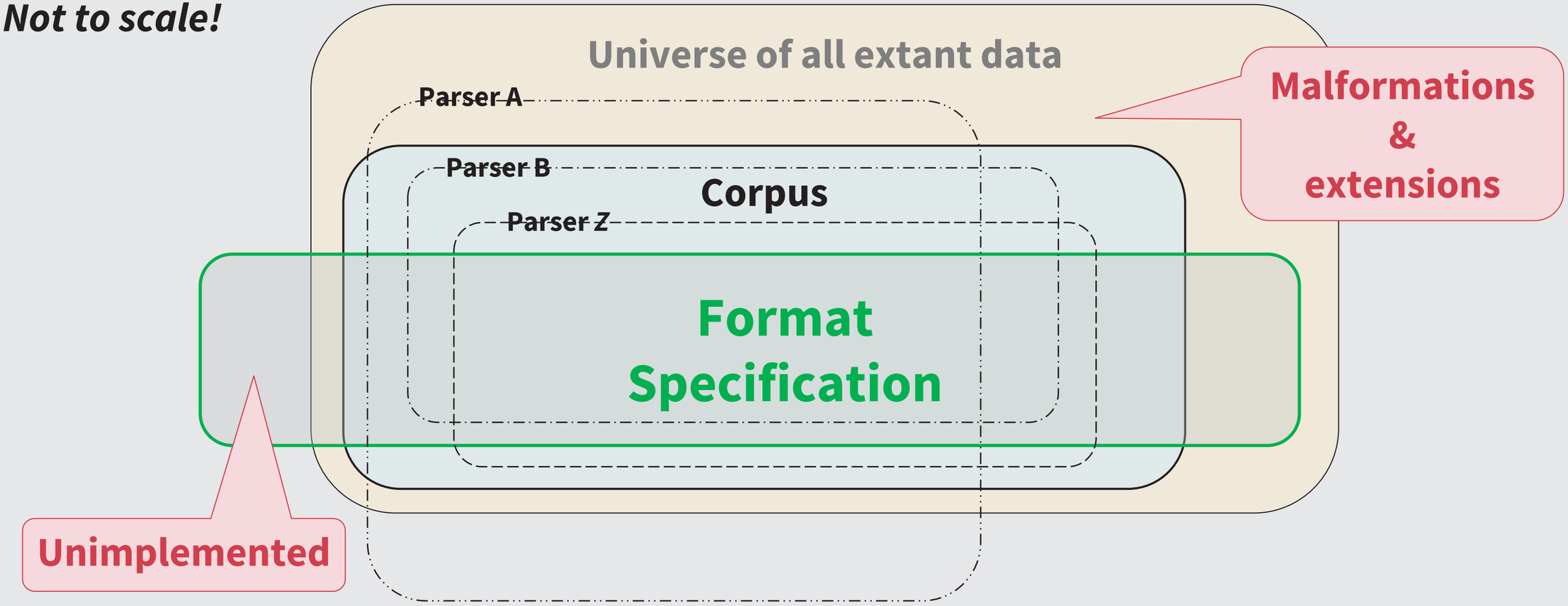
Specifications and essential truths

Not to scale!



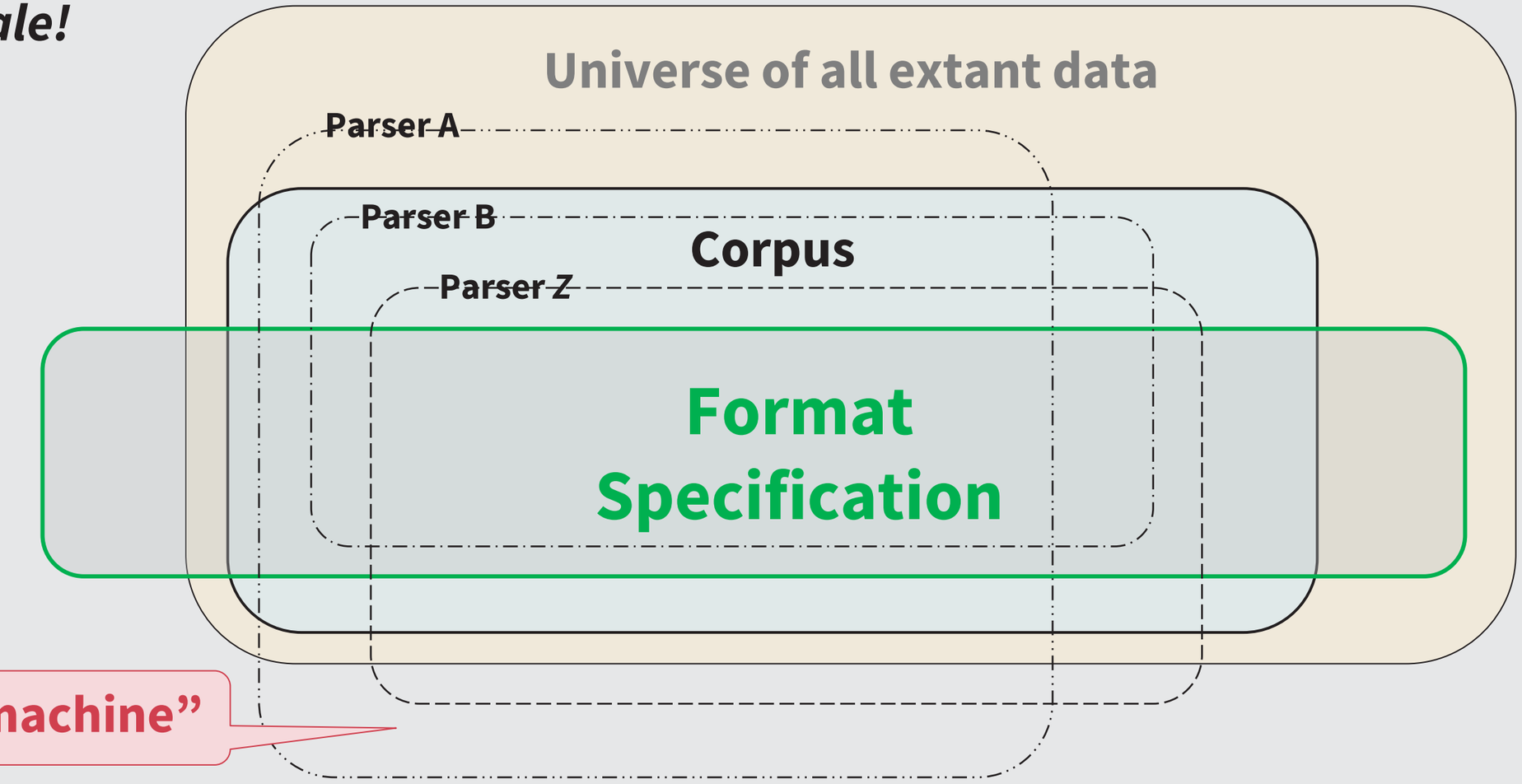
Specifications and essential truths

Not to scale!



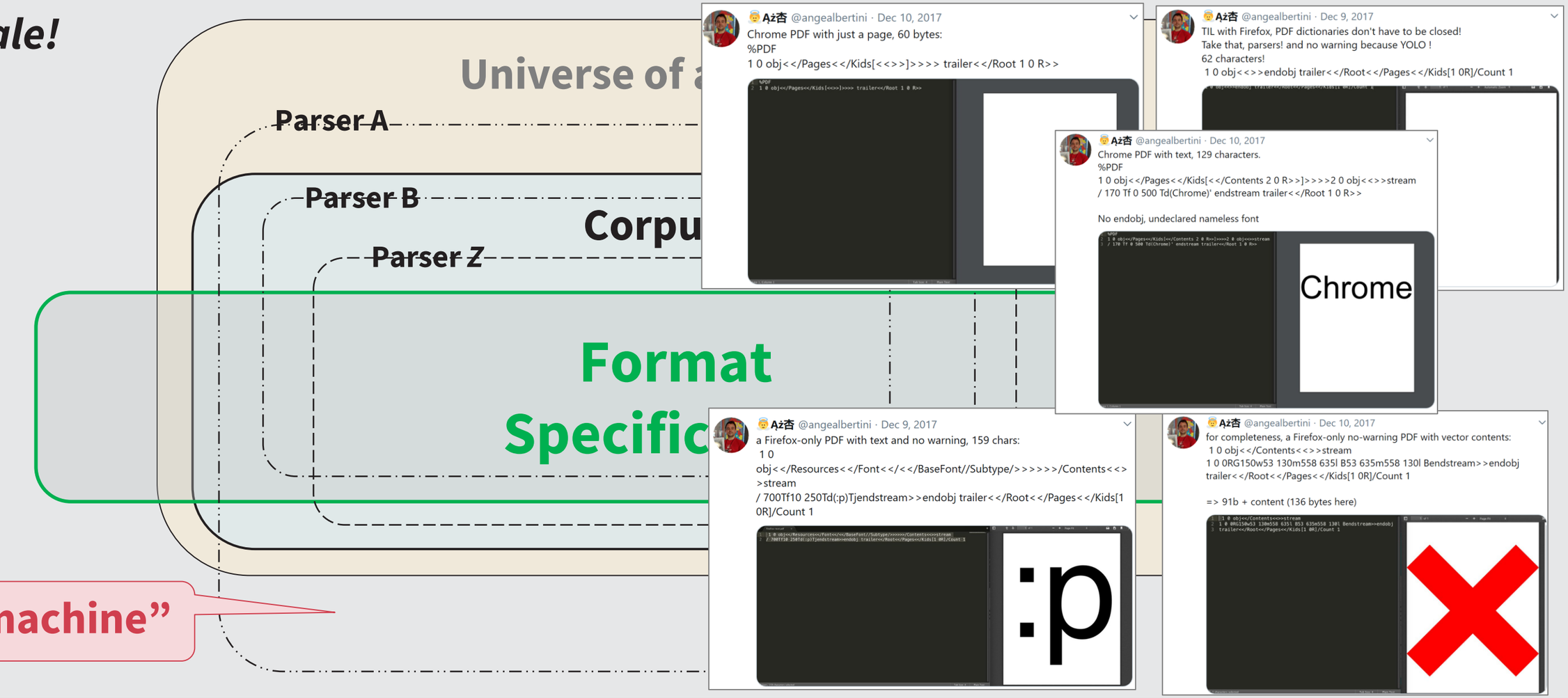
Specifications and essential truths

Not to scale!



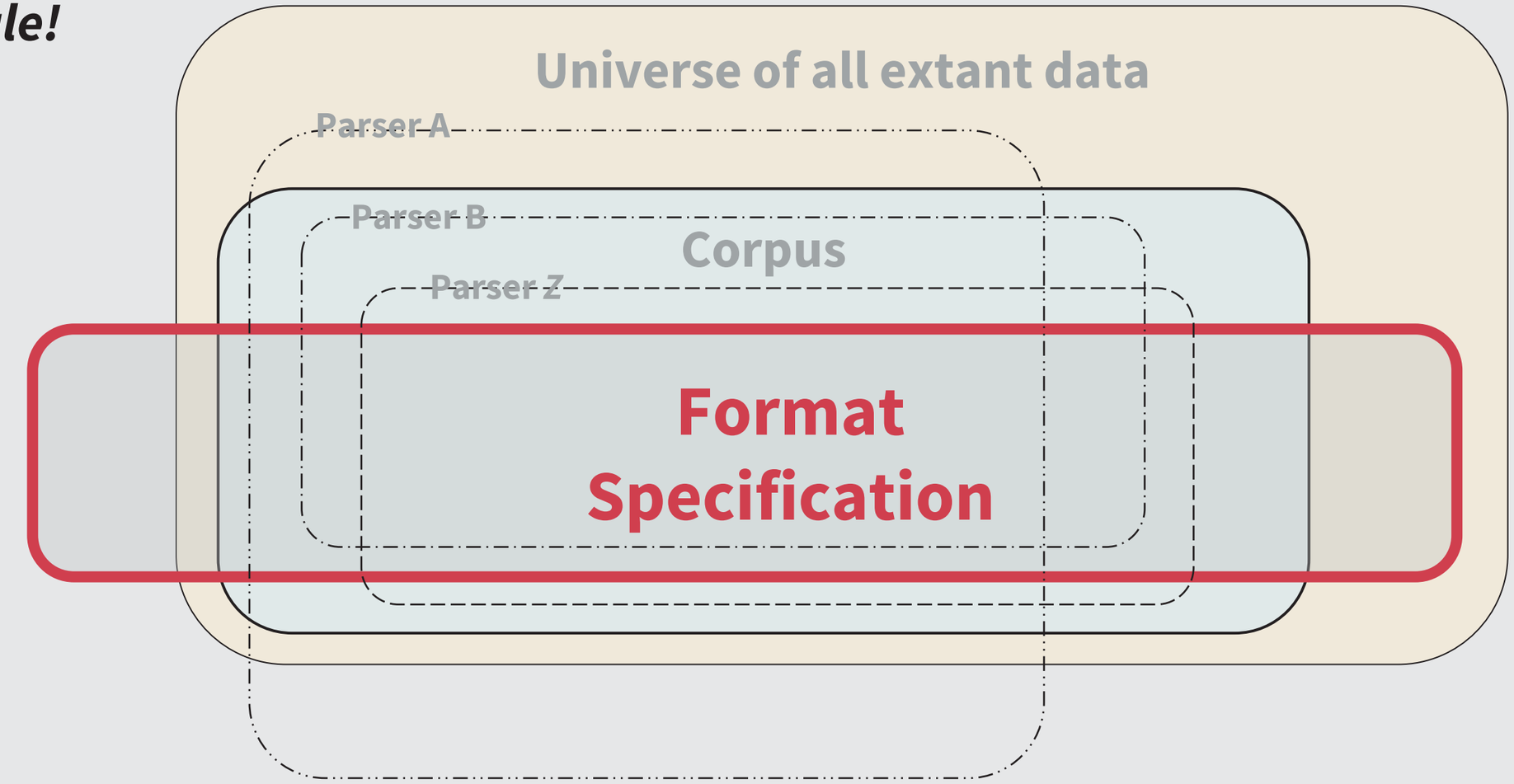
Specifications and essential truths

Not to scale!

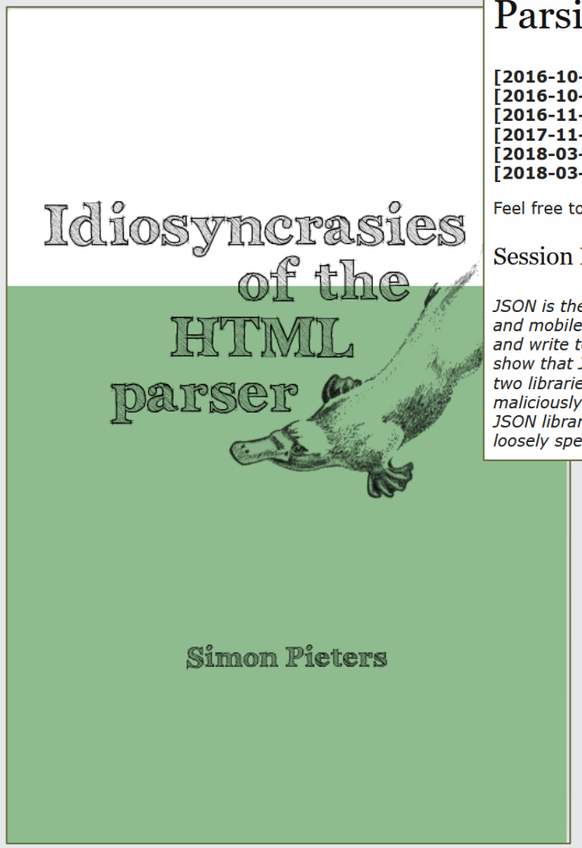


Specifications and essential truths

Not to scale!



Specifications



Parsing JSON is a Minefield 💣

[2016-10-26] First version of the article
[2016-10-28] Presentation at Soft-Shake Conference, Geneva (slides)
[2016-11-01] Article and comments in The Register
[2017-11-16] Presentation at Black Alps Security Conference, Yverdon (slides)
[2018-03-09] Presentation at Toulouse Hacking Conference (slides)
[2018-03-30] Updated this article considering RFC 8259

Feel free to comment on Hacker News (2016-10), Hacker News (2018-04) or reddit.

Session Description

JSON is the de facto standard when it comes to (un)serialising and exchanging data in web and mobile programming. But how well do you really know JSON? We'll read the specificati and write test cases together. We'll test common JSON libraries against our test cases. I'll show that JSON is no find two libraries that exh nd maliciously crafted p caus JSON libraries rely o ails loosely specified or n

An Exploration of JSON Interoperability Vulnerabilities

Jake Miller on Feb 25, 2021 5:00:00 AM

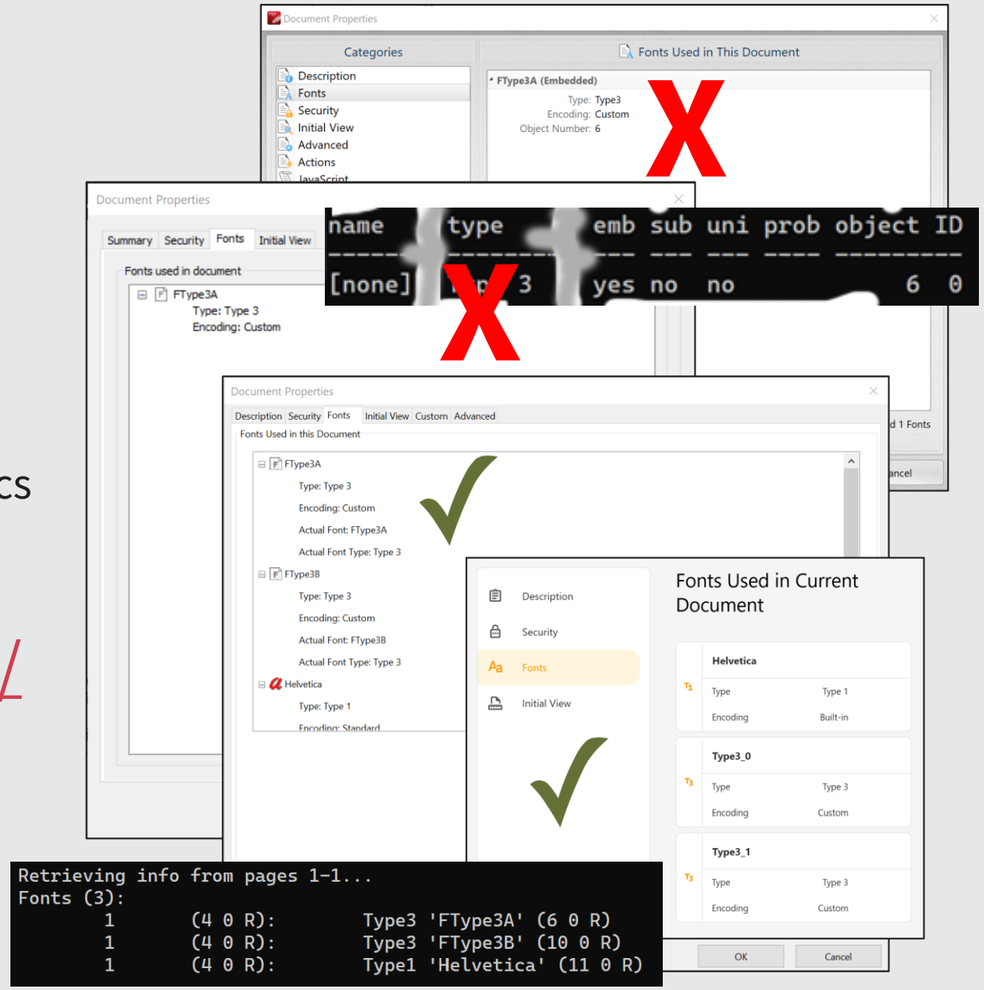
```
o = {
    "qty": 1,
    "qty": -1
}
o["qty"] // ???
```

TL;DR The same JSON document can be parsed with different values across microservices, leading to a variety of potential security risks. If you prefer a hands-on approach, try the labs and when they scare you, come back and read on.

https://htmlparser.info/
https://seriot.ch/projects/parsing_json.html
https://labs.bishopfox.com/tech-blog/an-exploration-of-json-interoperability-vulnerabilities

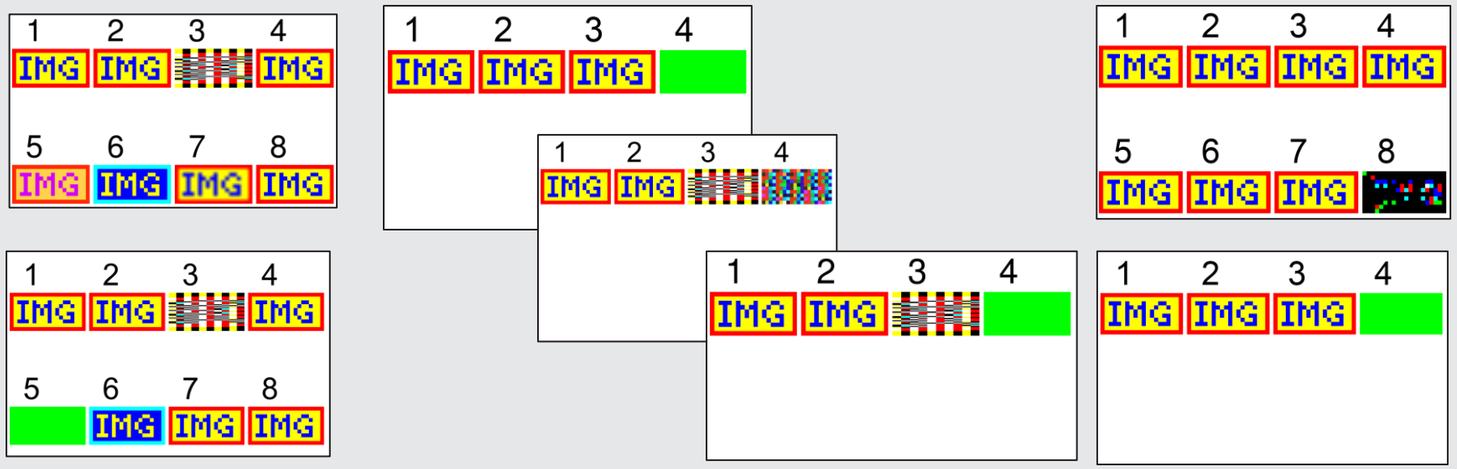
Something as simple as fonts…

- **Methodical**:
  - All text requires fonts
  - Fonts are a type of Resource
  - Any content stream can have Resources
    - Page, annotations, Form XObjects, Type 1 Patterns, Type3 CharProcs
- GitHub https://github.com/pdf-association/safedocs/
  - “Miscellaneous Targeted Test PDFs” folder



Inline image with both full & abbreviated keys

- PDF TWG industry correction:
  - "*In the situation where both an abbreviated key name and the corresponding full key name from Table 91 are present, the abbreviated key name shall take precedence.*"
- GitHub https://github.com/pdf-association/safedocs/
  - "Inline Image Abbreviations" folder



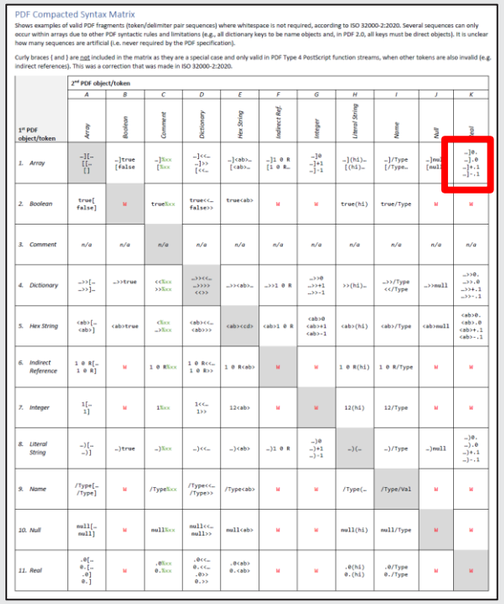
Full key name	Abbreviation
BitsPerComponent	BPC
ColorSpace	CS
Decode	D
DecodeParms	DP
Filter	F
Height	H
ImageMask	IM
Interpolate	I (*uppercase i*)
Length (*PDF 2.0*)	L
Width	W

Compacted syntax

- Test of PDF lexical analyzers for all 121 delimiter token pairings
  - In body of a PDF file
  - In PDF content streams
- GitHub: https://github.com/pdf-association/safedocs/
  - “Compacted Syntax” folder
  - ***Cannot*** be tested visually

```
<</SomeKey<</A/B>>>>
[[/A/B][/C/D]]
<0a><0d>
(cat)(mat)
```

```
…]1.23
…]-1.23
…]+1.23
…].23
```


PDF Compacted Syntax Matrix

Shows examples of valid PDF fragments (token/delimiter pair sequences) where whitespace is not required, according to ISO 32000-2:2020. Several sequences can only occur within arrays due to other PDF syntactic rules and limitations (e.g., all dictionary keys to be name objects and, in PDF 2.0, all keys must be direct objects). It is unclear how many sequences are artificial (i.e. never required by the PDF specification).

Curly braces { and } are not included in the matrix as they are a special case and only valid in PDF Type 4 PostScript function streams, when other tokens are also invalid (e.g. indirect references). This was a correction that was made in ISO 32000-2:2020.

1st PDF object/token \ 2nd PDF object/token	A Array	B Boolean	C Comment	D Dictionary	E Hex String	F Indirect Ref.	G Integer	H Literal String	I Name	J Null	K Real
1. Array	…][…[…[]…]…]	…]true[false	…]%xx[%xx	…]<<…>>[<<…	…]<ab>…[<ab>…	…]1 0 R[1 0 R…	…]0[…]+1[…]-1	…](hi)…[(hi)…	…]/Type[/Type…	…]null[null	…]0.[…].0[…]+.1[…]-.1
2. Boolean	true[false]	✖	true%xx	true<<…false>>	true<ab>	✖	✖	true(hi)	true/Type	✖	✖
3. Comment	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
4. Dictionary	…>>[<…>>]…	…>>true	<<%xx>>%xx	…>>>><<…>>>><<>>	…>><ab>…	…>>1 0 R	…>>0<<…>>+1<<…>>-1	>>(hi)…	…>>/Type<</Type	…>>null	…>>0.<<…>>+.1<<…>>-.1
5. Hex String	<ab>[…<ab>]	<ab>true	<%xx…>%xx	<ab>><<…<ab>>>	<ab><cd>	<ab>1 0 R	<ab>0<ab>+1<ab>-1	<ab>(hi)	<ab>/Type	<ab>null	<ab>0.<ab>+.1<ab>-.1
6. Indirect Reference	1 0 R[…1 0 R]	✖	1 0 R%xx	1 0 R<<…1 0 R>>	1 0 R<ab>	✖	✖	1 0 R(hi)	1 0 R/Type	✖	✖
7. Integer	1[…1]	✖	1%xx	1<<…1>>	12<ab>	✖	✖	12(hi)	12/Type	✖	✖
8. Literal String	…)[……)]	…)true	…)%xx	…)<<…	…)<ab>	…)1 0 R	…)0…)+1…)-1	…)(…	…)/Type	…)null	…)0.…).0…)+.1…)-.1
9. Name	/Type[…/Type]	✖	/Type%xx	/Type<<…/Type>>	/Type<ab>	✖	✖	/Type(…	/Type/Val	✖	✖
10. Null	null[…null]	✖	null%xx	null<<…null>>	null<ab>	✖	✖	null(hi)	null/Type	✖	✖
11. Real	.0[…0.[….0]0.]	✖	.0%xx0.%xx	.0<<…0.<<…0>>0.>>	.0<ab>0.<ab>	✖	✖	.0(hi)0.(hi)	.0/Type0./Type	✖	✖

Dialects

- Small variations in a grammar
- *Where can inline dictionaries (<< … >>) occur in PDF content streams?*

```
BI
  /Width  20
  /Height 10
  /BPC     8
  /CS    /DeviceRGB
  /ACME_Private << /SomeDict << /Type /FooBar >> >>
  /Filter [/ASCIIHexDecode]
  /Length 1240
ID
ff0000ff0000ff0000ff0000ff0000ff
0000ff0000ff0000ff0000ff0000ff00
```

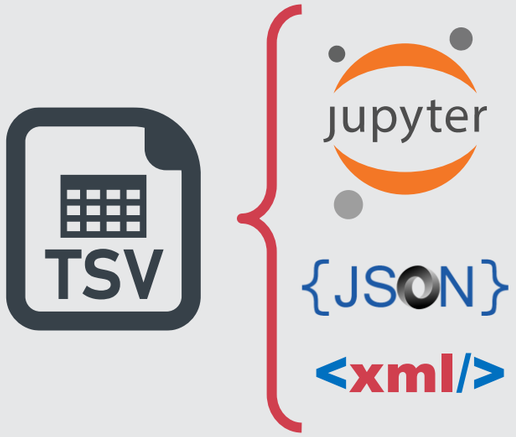
```
BI
  /Width  20
  /Height 10
  /BPC     8
  /CS    /DeviceRGB
  /ACME_Private << /SomeDict << /Type /FooBar /Self 7 0 R >> >>
  /Filter [/ASCIIHexDecode]
  /Length 1240
ID
ff0000ff0000ff0000ff0000ff0000ff
0000ff0000ff0000ff0000ff0000ff00
```

```
BX
        << /SomeKey /SomeValue /InnerDict << /Key1 /Value1 >> >> newoperator
        BT
          0 0 0 rg % Black text
          /F1 4 Tf
          10 0 0 10 45 260 Tm
          (Inside BX/EX, after unknown operator) Tj
        ET
EX
```

```
BX
        << /SomeKey /SomeValue /InnerDict << /Key1 /Value1 /Self 7 0 R>> >> newoperator
        BT
          0 0 0 rg % Black text
          /F1 4 Tf
          10 0 0 10 45 260 Tm
          (Inside BX/EX, after unknown operator) Tj
        ET
EX
```

Arlington PDF Model

- **First open access, vendor neutral, specification-derived, machine and human readable, comprehensive definition of all PDF objects**
- Easy to understand and use
  - Text-based TSV file sets
  - 12 fields with custom predicates
  - Platform and language agnostic
  - Easily transformable
  - No code / low code / code

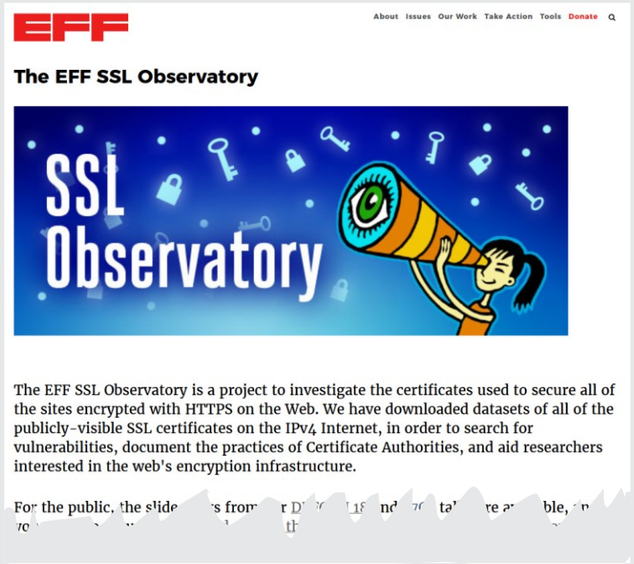


Arlington PDF Model

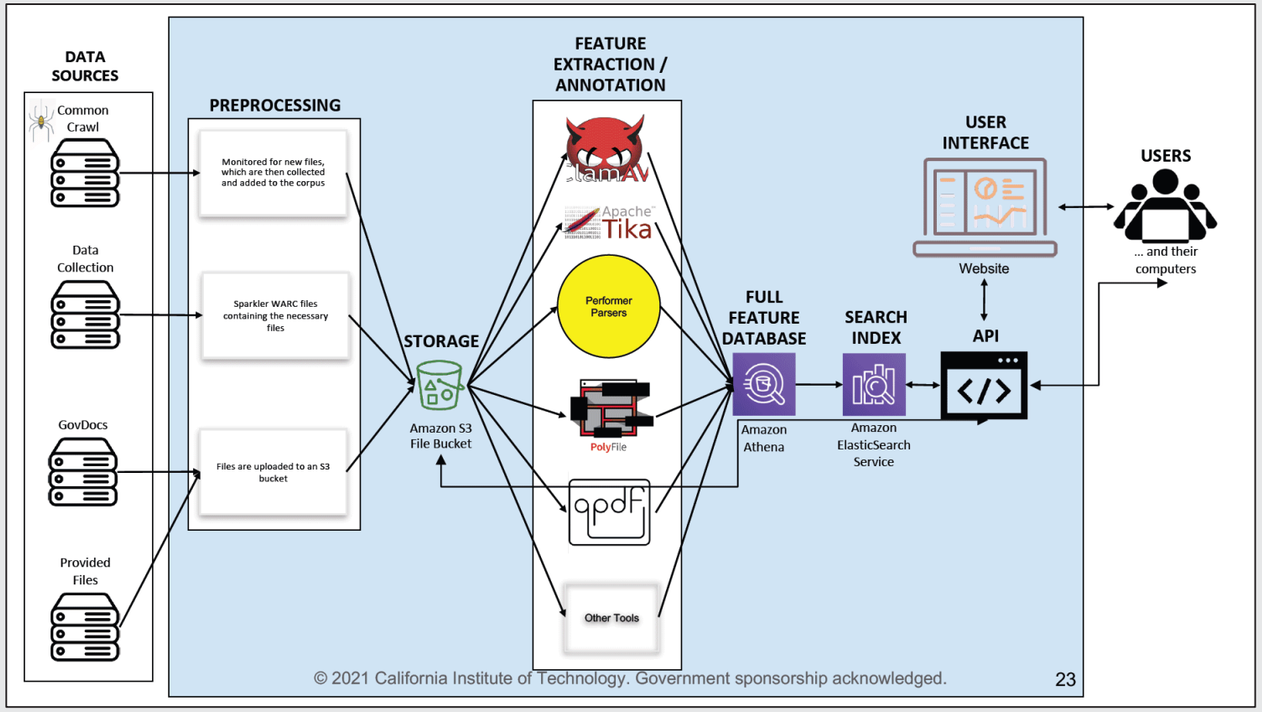
https://github.com/pdf-association/arlington-pdf-model

PDF Observatory: PDF at scale

- "*Making sense of PDF structures in the wild at scale*", Tim Allison, JPL

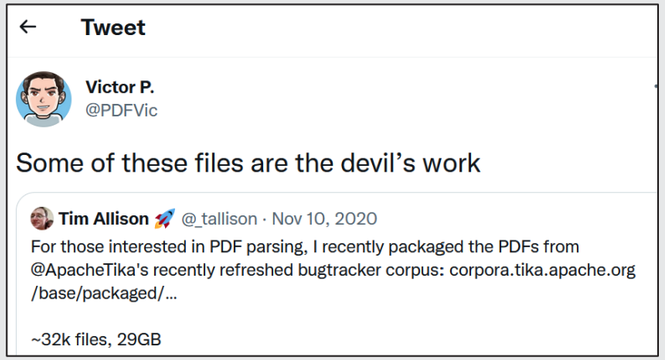


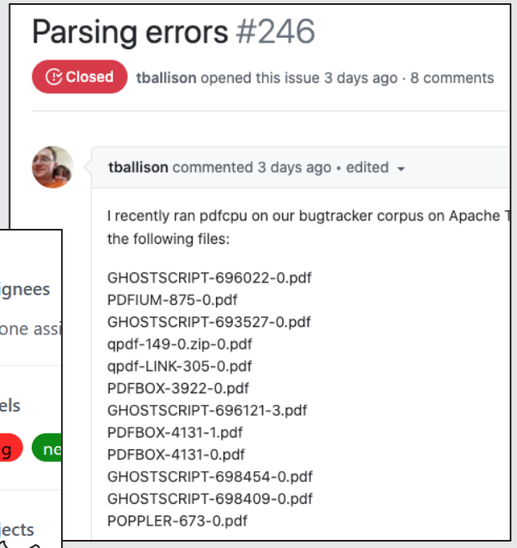
https://www.eff.org/observatory

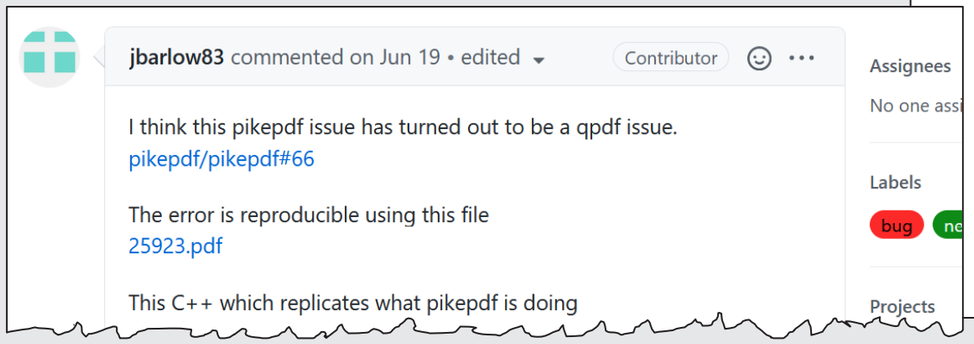


Issue Tracker stressful corpus

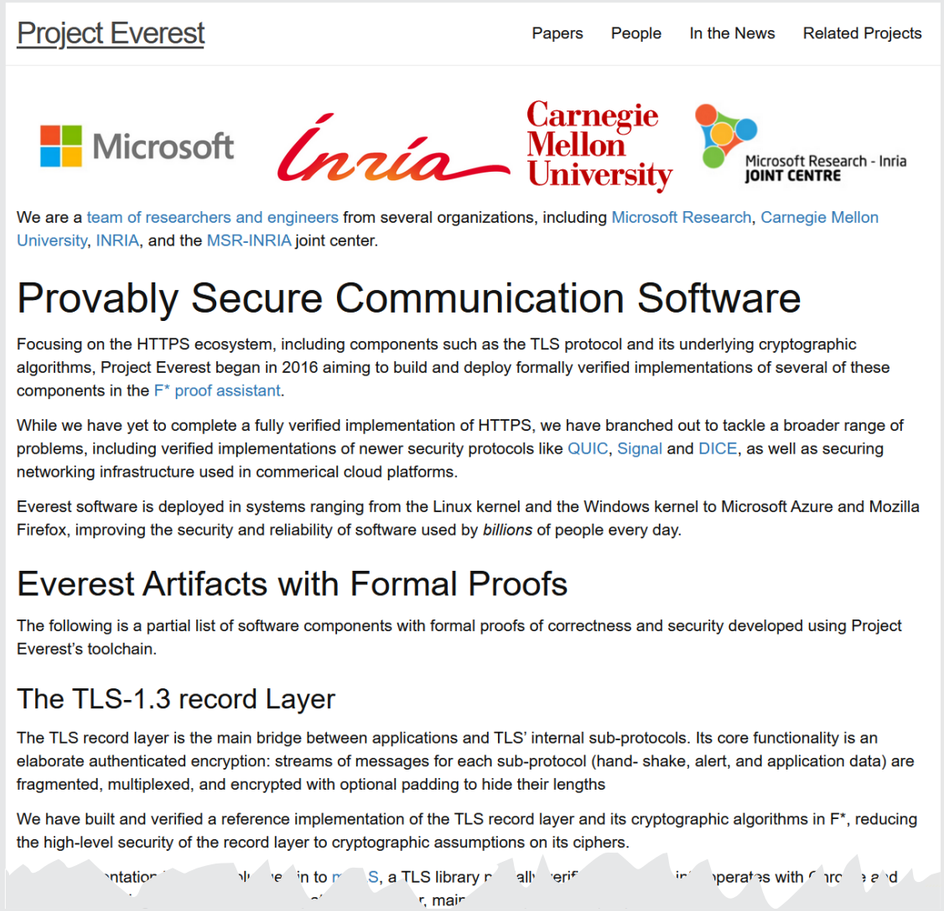
- https://www.pdfa.org/stressful-pdf-corpus/
- https://corpora.tika.apache.org/base/
  - Pre-packaged PDFs: https://corpora.tika.apache.org/base/packaged/pdfs/
- Other corpora: https://github.com/pdf-association/pdf-corpora







Microsoft EverParse / Project Everest



Project Everest

Papers People In the News Related Projects

Microsoft Inria Carnegie Mellon University Microsoft Research - Inria JOINT CENTRE

We are a team of researchers and engineers from several organizations, including Microsoft Research, Carnegie Mellon University, INRIA, and the MSR-INRIA joint center.

Provably Secure Communication Software

Focusing on the HTTPS ecosystem, including components such as the TLS protocol and its underlying cryptographic algorithms, Project Everest began in 2016 aiming to build and deploy formally verified implementations of several of these components in the F* proof assistant.

While we have yet to complete a fully verified implementation of HTTPS, we have branched out to tackle a broader range of problems, including verified implementations of newer security protocols like QUIC, Signal and DICE, as well as securing networking infrastructure used in commerical cloud platforms.

Everest software is deployed in systems ranging from the Linux kernel and the Windows kernel to Microsoft Azure and Mozilla Firefox, improving the security and reliability of software used by *billions* of people every day.

Everest Artifacts with Formal Proofs

The following is a partial list of software components with formal proofs of correctness and security developed using Project Everest's toolchain.

The TLS-1.3 record Layer

The TLS record layer is the main bridge between applications and TLS' internal sub-protocols. Its core functionality is an elaborate authenticated encryption: streams of messages for each sub-protocol (hand- shake, alert, and application data) are fragmented, multiplexed, and encrypted with optional padding to hide their lengths

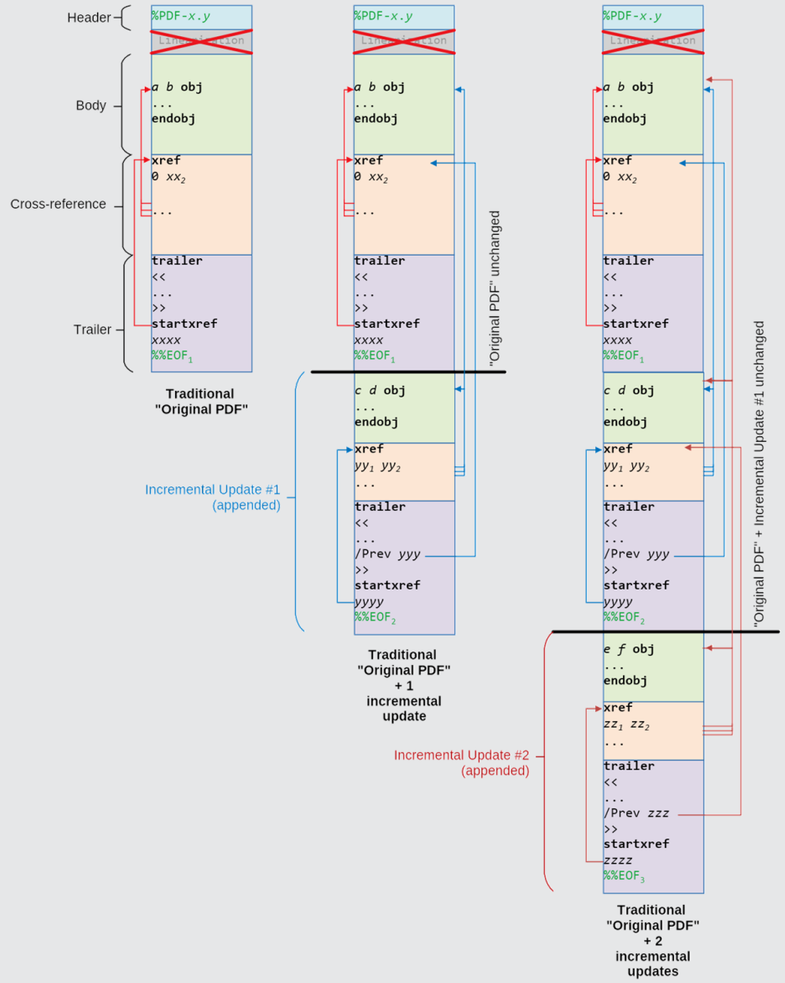
We have built and verified a reference implementation of the TLS record layer and its cryptographic algorithms in F*, reducing the high-level security of the record layer to cryptographic assumptions on its ciphers.

https://project-everest.github.io/

https://www.microsoft.com/en-us/research/blog/everparse-hardening-critical-attack-surfaces-with-formally-proven-message-parsers/

Where to next?

- **PDF “Chain of Trust”**
  - ISO TC 171 SC 2 WG 8 “Securing PDF” discussion group
- **DDLs and Parser Generation Toolkits**
  - Verifiably correct parsers
  - Parsley/PVS, DaeDaLus, GGG/Hammer
- **iccMAX (ICC.2 / V5)**
  - ISO 20677 “Calculator” element



Resources

- **Now**
  - PDF Association's SafeDocs repo: targeted PDFs
  - "Issue Tracker" stressful corpus: high ROI
  - The Arlington PDF Model: machine-readable model
- **Future**
  - "PDF Observatory": understanding PDF at scale
  - PDF Chain of Trust
  - iccMAX

GitHub

Arlington PDF Model

Thank you

Questions?

peter.wyatt@pdfa.org