OASIS CTI F2F – CybOX Session 1

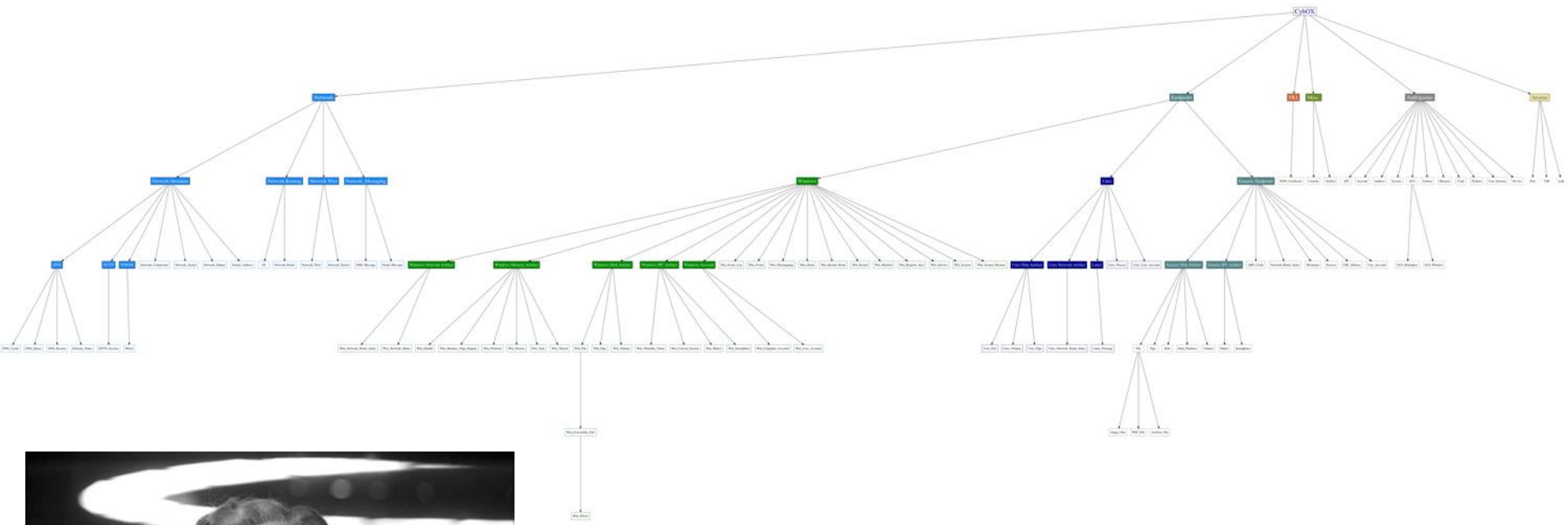
January 14, 2016

Ch-ch-ch-ch-changes

CybOX overview: history

- Initial driver: need for a standard to characterize cyber observables
- Standard was developed in an ad-hoc fashion to address evolving use cases
  - Informed by MAEC, STIX, DFAX, and others
- Organic (resource-constrained) development led to divergent architecture
- With the benefit of hindsight and the shared experience of implementers, numerous deficits have come to light

CybOX overview: challenges



”We have a taxonomy gap!”

CybOX 3.0 Design Philosophy

- Focus: simplification, reducing duplicate mechanisms
- Prioritizing refactoring of most-commonly used CybOX Objects
- Primary design consideration: backward-compatible-breaking changes now, stability for the foreseeable future:
  - New CybOX Objects will be added in sequential point releases targeting specific use cases, in service to community demand, building on a cohesive foundational rethinking of CybOX base constructs
  - CybOX 3.0 base constructs will be architected with care to forestall impacting backward-compatibility in additive point releases

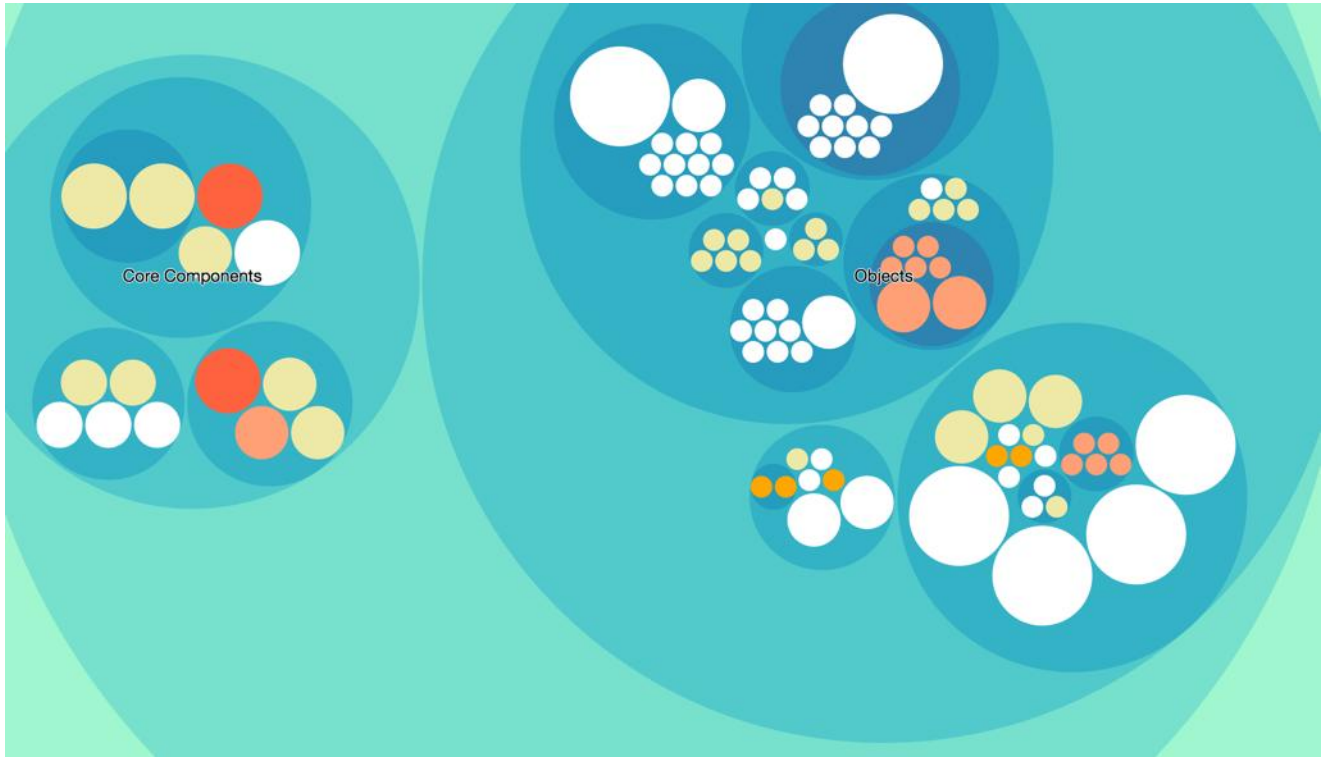
CybOX 3.0: Don't bite off more than you can chew!

- Refactoring and fixing issues with *all* existing CybOX objects in a single release is not achievable in a meaningful timeframe to address the pain currently experienced by the community

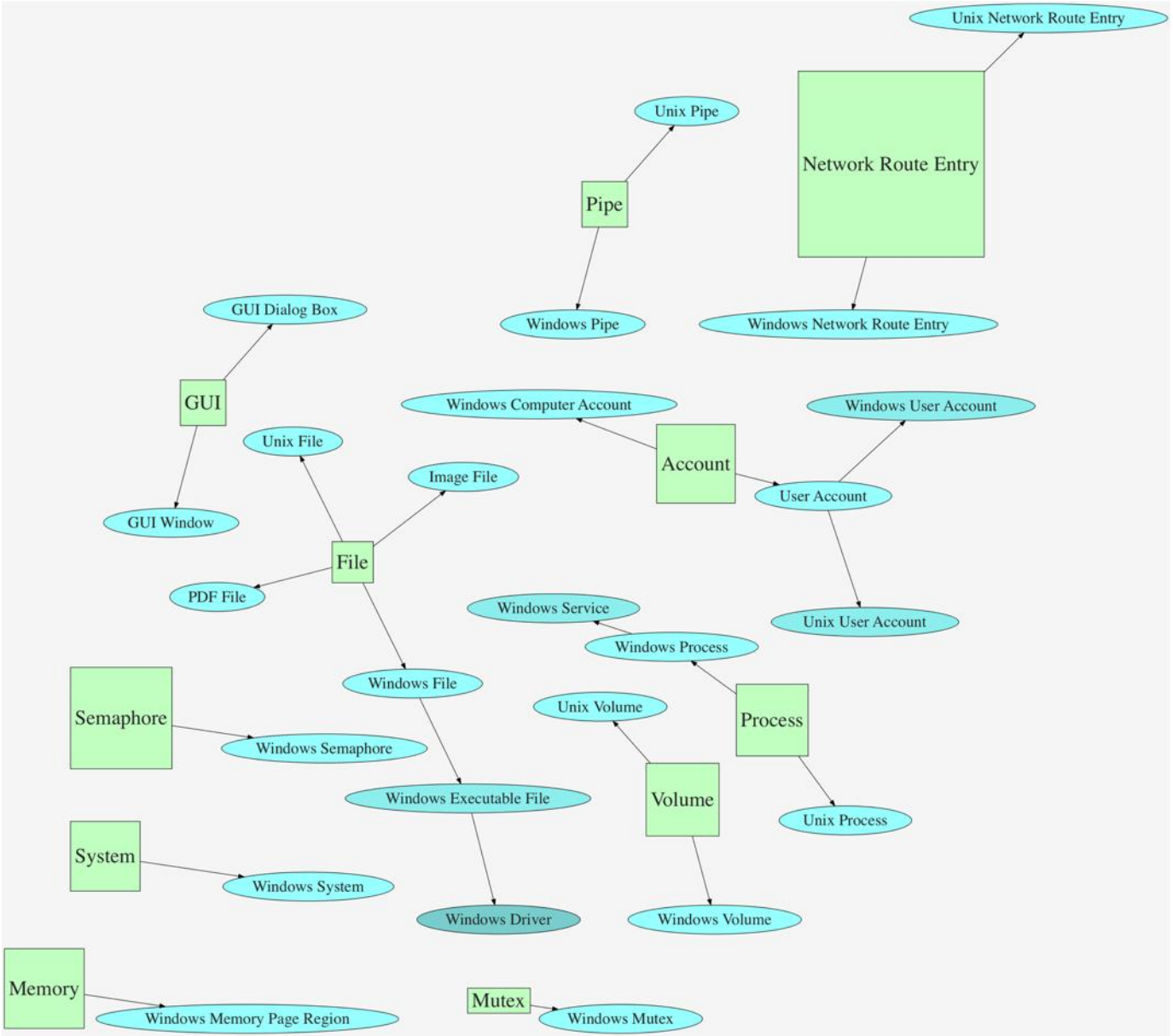
- Pragmatic approach: focus on a core set of CybOX Objects for the 3.0 release
  - Meet the ≈80% need ASAP, iterate quickly from there…

CybOX overview: ongoing refactoring approach

- Quantitative analysis: cti-stats
- A picture's worth a thousand pull-requests



CybOX Design: Object Hierarchy



CybOX Design: Object Hierarchy

https://github.com/CybOXProject/schemas/wiki/CybOX-Design:-Object-Hierarchy-Structuring

- Key Issue: how should the CybOX Object hierarchy be designed?
  - Current parent/child (sub-class) based approach has several issues
    - Excessive subclassing
    - Inconsistency
    - Instance content restriction

- **Open Questions**
  - Does it make sense to organize Objects around a separate Extension structure?
  - Or should we follow the more granular (non sub-class) philosophy of extensions, but instead define them as separate Objects using the existing structure?
    - E.g., an NTFS File *Object* vs. an NTFS File *Extension*

Address Object Refactoring

https://github.com/CybOXProject/schemas/wiki/CybOX-3.0:-Address-Object-Refactoring

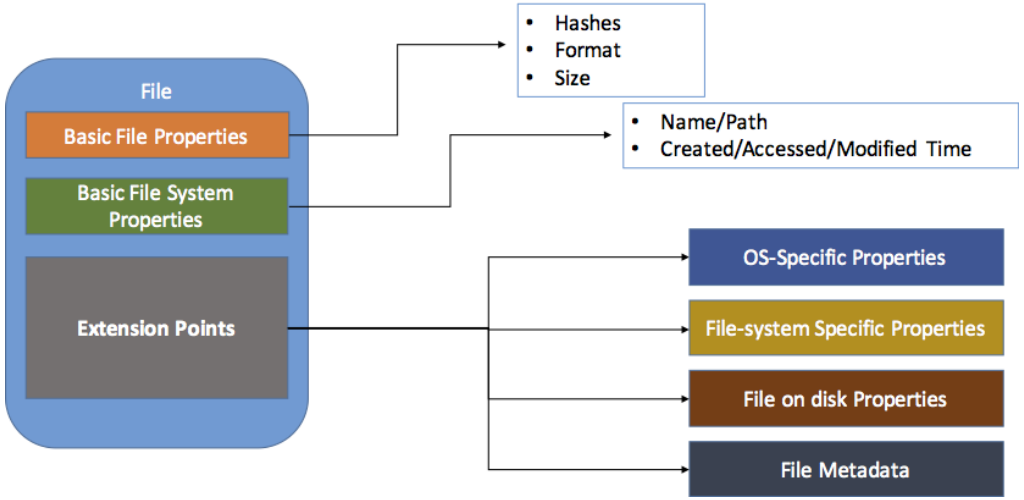
- CybOX 2.1
  - **Address**
- CybOX 3.0
  - IP Address
  - IPv4 Address
  - IPv6 Address
  - MAC Address
  - Email Address

```
{
    "ip_address" : "192.168.1.1/32",
    "version" : "ipv4"
}
```

- **Open Questions**
  - Should CIDR be the *only* supported notation for IP addresses?
  - How should IP addresses be modeled?
    - Should IPv4/IPv6 be an extension of IP Address?

File Object Refactoring

https://github.com/CybOXProject/schemas/wiki/CybOX-3.0:-File-Object-Refactoring

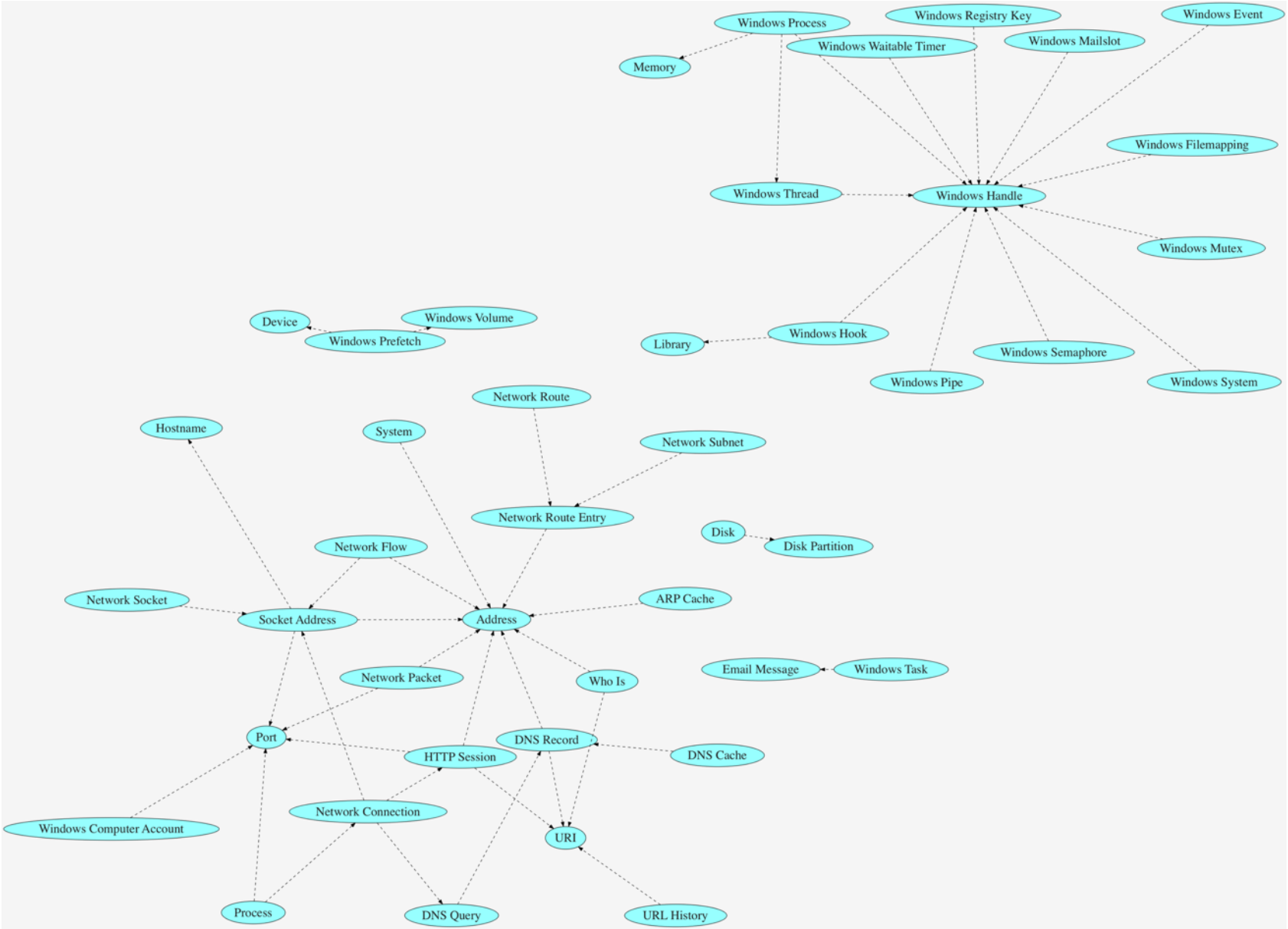


- **Default Extensions**
  - File Metadata
  - EXT3 File
  - NTFS File
  - Image File
  - PDF File
  - Archive File
  - PE Binary File
  - …

- **Open Questions**
  - What default extensions make sense?
  - Should File encompass other file-like Objects?
    - Pipe
    - Socket
    - …


```
{
  "hashes" : [{"type":"md5",
               "hash-value":"3773a88f65a5e780c8dff9cdc3a056f3"}],
  "size" : 25537,
  "file-system-properties":{"is-directory":false,
                            "file-name": "foo.exe",
                            "file-path": {"delimiter":"/",
                                          "components":["usr","tmp"]}},
  "extended-properties": [{"type":"FileMetadataExtension",
                           "mime-type":"vnd.microsoft.portable-executable"},
                          {"type":"EXT3FileExtension",
                           "inode":"34483923"},
                          {"type":"PEBinaryFileExtension",
                           "exports":[{"name":"foo_app"}]}]
}
```

CybOX Design: Obj. Relationships



CybOX Design: Object Relationships?

https://github.com/CybOXProject/schemas/wiki/CybOX-Design:-Relationships-vs.-Embedded-Objects

- Key Issue
  - Relationships between Objects can be expressed in two different ways
    - Directly embedded via Fields
    - Indirectly via an Object Relationship
  - For the sake of simplicity ("one way of doing things") we should use a single approach

- **Open Questions**
  - Does it make sense to express Object <-- --> Object relationships ONLY via an explicit Relationship construct?
    - How should the corner case of relationships between Object *Types* and other Objects be handled?
    - How should we deal with Objects such as the Socket Address that are exclusively defined based on other Objects?

Just the facts, ma'am…

Ongoing debate about where patterning should live…

There's a prevalent notion that CybOX should be the alphabet and that other related standards should define the semantic context.

Just food for thought…

Iterating: point releases

- Mobile Objects
  - Android
  - iOS
- SCADA Objects
- Additional Layer 7 Objects
  - FTP
  - SMTP
  - IRC
  - TAXII over TOR
- Document File Objects
  - OLE
    - DOC
    - PPT
  - DOCX
- Comprehensive network forensics
- Comprehensive endpoint forensics
- …

Going deeper down the rabbit hole…

Proposal: a week-long F2F focused on CybOX extension, in collaboration with DFAX, MAEC, and other non-OASIS stakeholders sometime Q3 2016?