# A (very) Brief Introduction to the Cyber Observables eXpression (CybOX)

IT Security Automation Conference

October 31st 2011

## What is a “Cyber Observable”?

***a measurable event or stateful property in the cyber domain***

## What is the “Cyber Observable eXpression”?

***a standardized means for representing cyber observables, i.e. a schema***

Some examples of CybOX *measureable events*

A registry key is created
A file is deleted
An HTTP Get Request is received
An IDS rule fired
…

**a few examples**

Some examples of CybOX *stateful properties*

MD5 hash of a file
Value of a registry key
Existence of a mutex
…

What are the indicators of a specific cyber incident?

What are the indicators of a family of malware?

What are the indicators of a specific pattern of cyber attack?

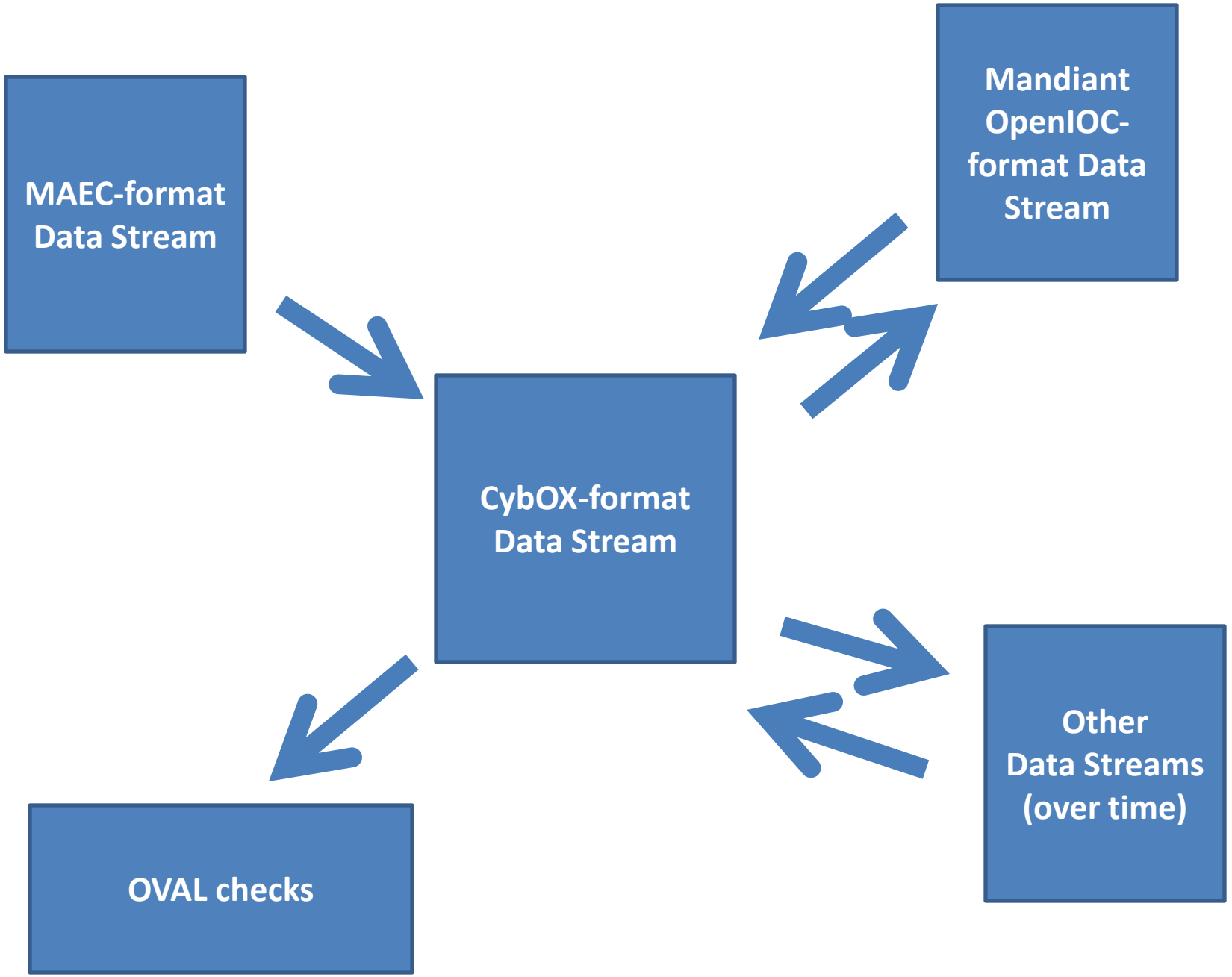
**CybOX is intended to help here…**

CybOX gives us a *standardized way* to represent this behavior

Storm Worm:
1. Drops a rootkit
2. Starts it as a service

Event:
Create file: **C:\Windows\spooldr.exe**
Event:
Create registry entry:
**Key=SYSTEM/CurrentControlSet\Services\spooldr**
**Hive=HKEY_LOCAL_MACHINE**

**Simple Example: Storm Worm**


MAEC-format Data Stream
Mandiant OpenIOC-format Data Stream
CybOX-format Data Stream
OVAL checks
Other Data Streams (over time)

# Cyber Observable eXpression (CybOX) Use Cases

## ITSAC 2011


http://cybox.mitre.org

Sean Barnum
Rich Struse

Oct 2011

Homeland Security

MITRE

NIST

# Cyber Observables Overview

- **The Cyber Observable eXpression (CybOX) is a standardized language for encoding and communicating high-fidelity information about cyber observables, whether dynamic events or stateful measures that are observable in the operational cyber domain.**
- **CybOX is not targeted at a single cyber security use case but rather is intended to be flexible enough to offer a common solution for all cyber security use cases requiring the ability to deal with cyber observables.**
- **It is also intended to be flexible enough to allow both the high-fidelity description of instances of cyber observables that have been measured in an operational context as well as more abstract patterns for potential observables that may be targets for observation and analysis apriori.**
- **By specifying a common structured schematic mechanism for these cyber observables, the intent is to enable the potential for detailed automatable sharing, mapping, detection and analysis heuristics.**

# Cyber Observables Apply to Numerous Domains

- Threat assessment & characterization (detailed attack patterns)
- Malware characterization
- Operational event management
- Logging
- Cyber situational awareness
- Incident response
- Forensics
- Etc.

- **Through utilization of the standardized CybOX language, relevant observable events or properties can be**
  - **captured and shared,**
  - **defined in rules**
  - **or used to adorn the appropriate portions of attack patterns and malware profiles in order to tie the logical pattern constructs to real-world evidence of their occurrence or presence for attack detection and characterization.**

- **Incident response and management can then take advantage of all of these capabilities to investigate occurring incidents, improve overall situational awareness and improve future attack detection, prevention and response.**

# The Role of Cyber Observables in Interoperability

- **Characterizing event records or streams as standardized observables**

  **[observable isComposedOf event]**

- **Mapping observables to detection rules**

  **[observable isModeledBy detectionRule]**

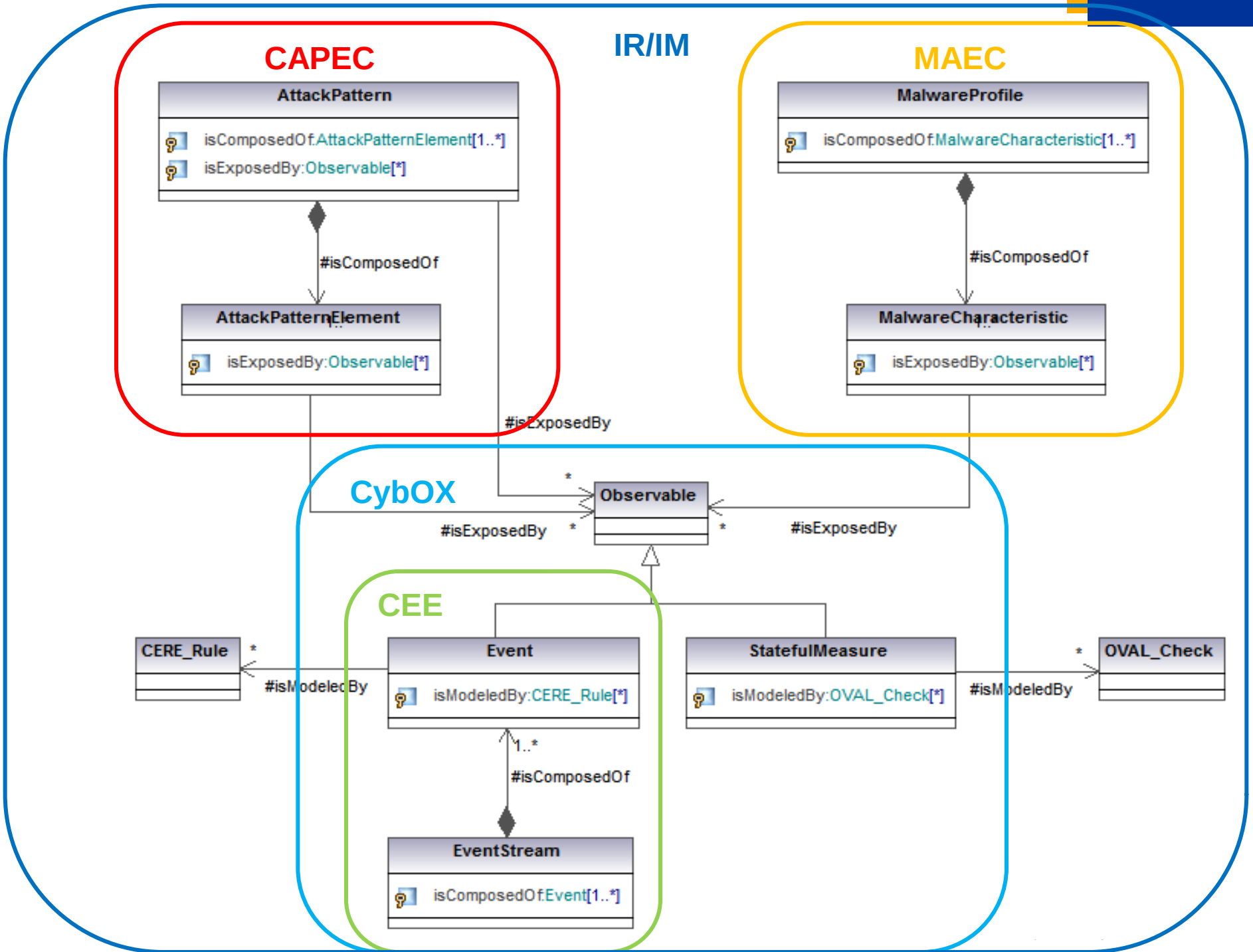
- **Mapping attack patterns to observables**

  **[attackPatternElement isExposedBy observable]**

- **Mapping malware characteristics to observables**

  **[malwareCharacteristic isExposedBy observable]**

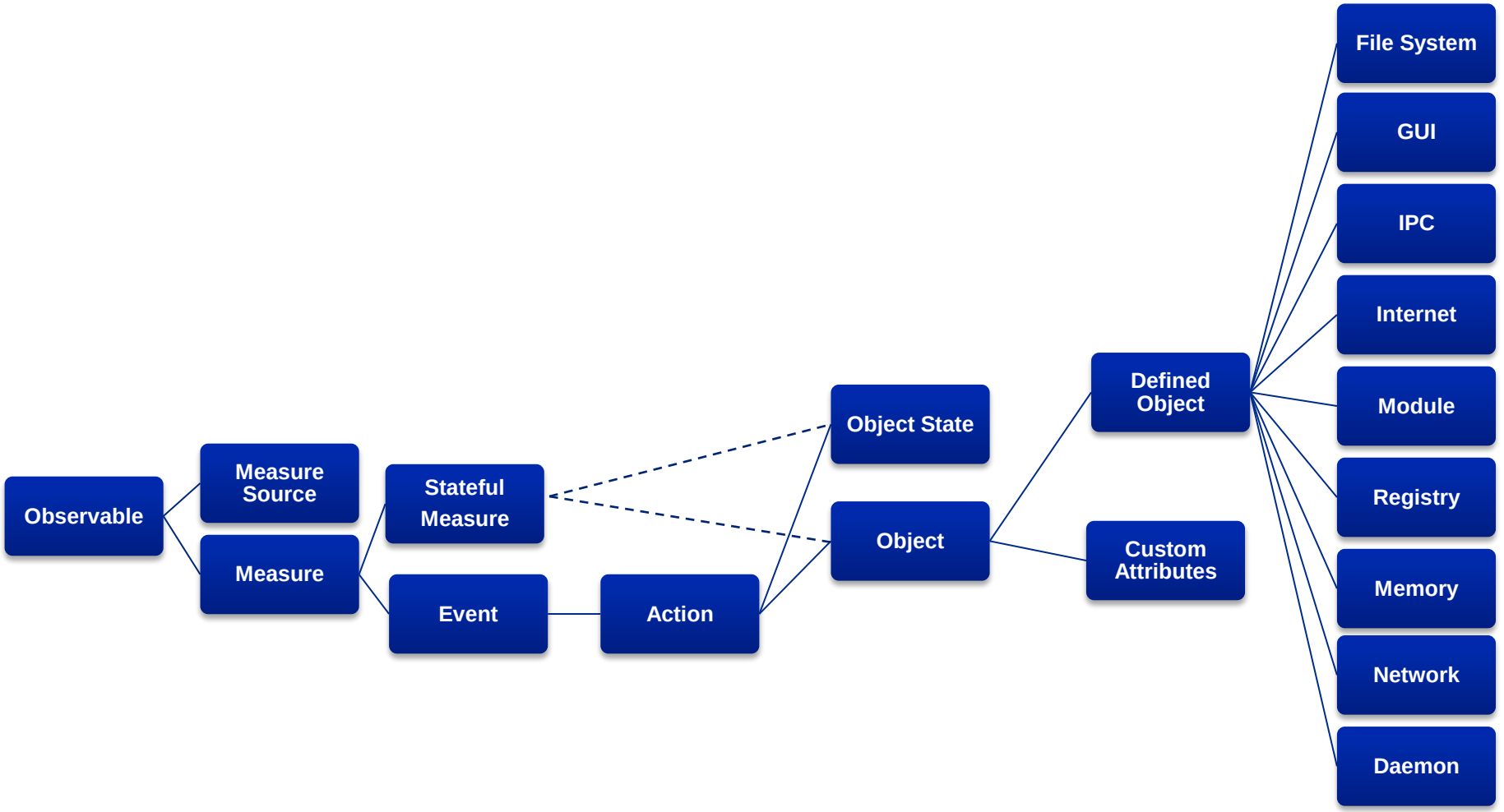
- **Incident Management automation through information exchange of standardized observables-based content**


IR/IM
CAPEC
AttackPattern
isComposedOf:AttackPatternElement[1..*]
isExposedBy:Observable[*]
#isComposedOf
AttackPatternElement
isExposedBy:Observable[*]
MAEC
MalwareProfile
isComposedOf:MalwareCharacteristic[1..*]
#isComposedOf
MalwareCharacteristic
isExposedBy:Observable[*]
#isExposedBy
CybOX
Observable
#isExposedBy
#isExposedBy
CEE
CERE_Rule
#isModeledBy
Event
isModeledBy:CERE_Rule[*]
1..*
#isComposedOf
EventStream
isComposedOf:Event[1..*]
StatefulMeasure
isModeledBy:OVAL_Check[*]
#isModeledBy
OVAL_Check

# A Brief History of Cyber Observables

- **September 2009: Originally introduced as part of CAPEC adornment to the structured Attack Execution Flow**
- **June 2010: Broader relevance to MSM recognized leading to CAPEC, MAEC & CEE teams collaborating to define one common structure to serve the common needs**
- **August 2010: Discussed with US-CERT at GFIRST 2010**
- **December 2010: Cyber Observables schema draft v0.4 completed**
- **December 2010: Discussions with Mandiant for collaboration and alignment between Cyber Observables and Mandiant OpenIOC**
- **January 2011: Discussed & briefed with MITRE CSOC**
- **February 2011: Discussed & briefed with NIST – EMAP and US-CERT who also have a need for this construct and had begun to work on parallel solutions**
- **May 2011: Schematic alignment and integration with CEE**
- **May 2011: Spun off as independent effort called the Cyber Observable eXpression (CybOX)**
- **Oct 2011: CybOX website launched with initial schema Version 0.6**

# Common Cyber Observables (CybOX) Schema






MITRE

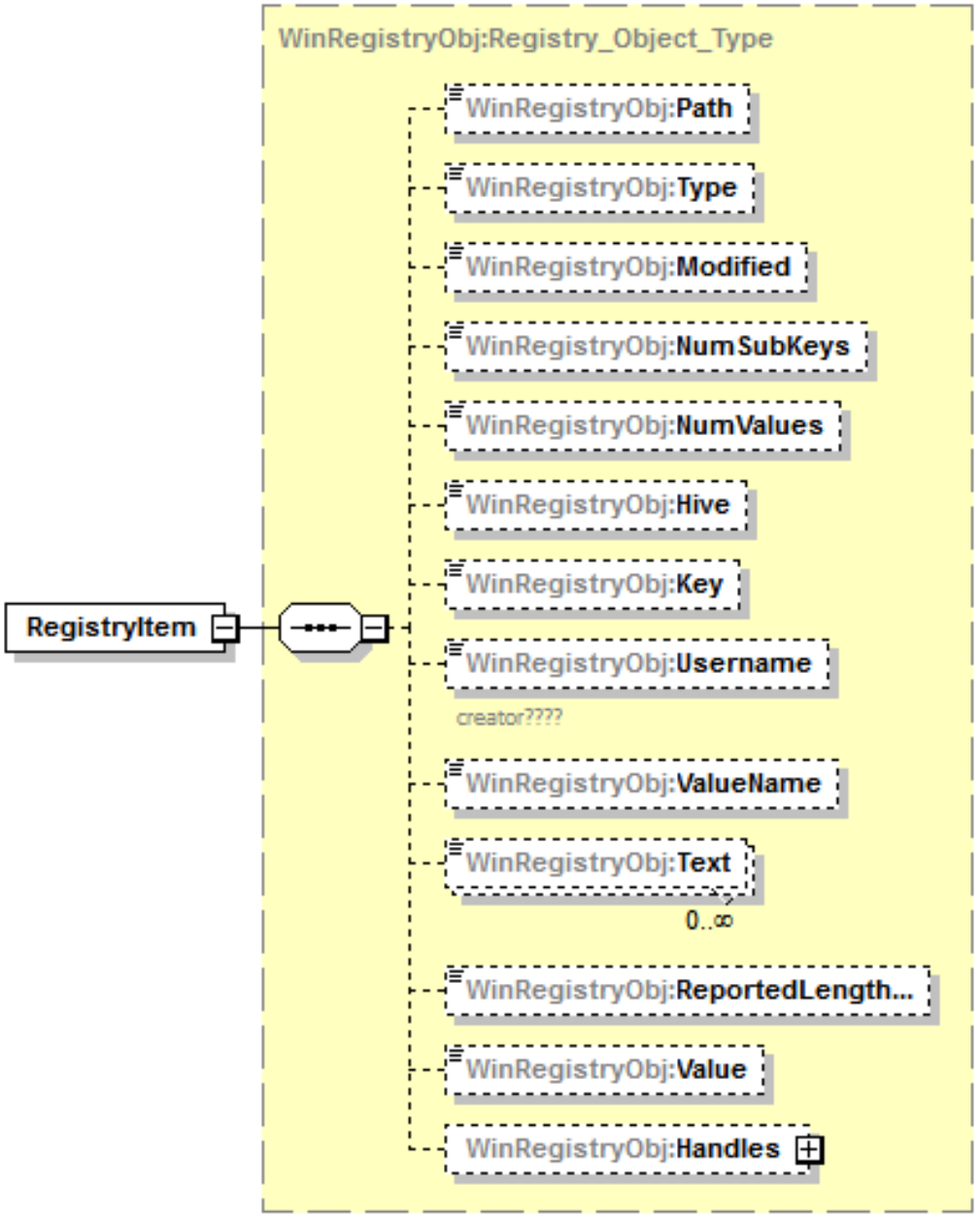
UNCLASSIFIED

# Various Defined Object Schemas

- Account
- Disk
- Disk Partition
- DNS Cache
- Email Message
- File
- GUI
- Library
- Package
- Memory
- Network Connection
- Network Route
- Linux Package
- Product
- Service
- Socket
- System
- User Session
- Volume
- Win Critical Section
- Win Driver
- Win Event
- Win Event Log
- Win Kernel
- Win Kernel Hook
- Win Handle
- Win Mailslot
- Win Mutex
- Win Named Pipe
- Win Network Route
- Win Prefetch
- Win Registry
- Win Semaphore
- Win System Restore
- Win Task
- Win Thread
- Win Waitable Timer
- X509 Certificate

…

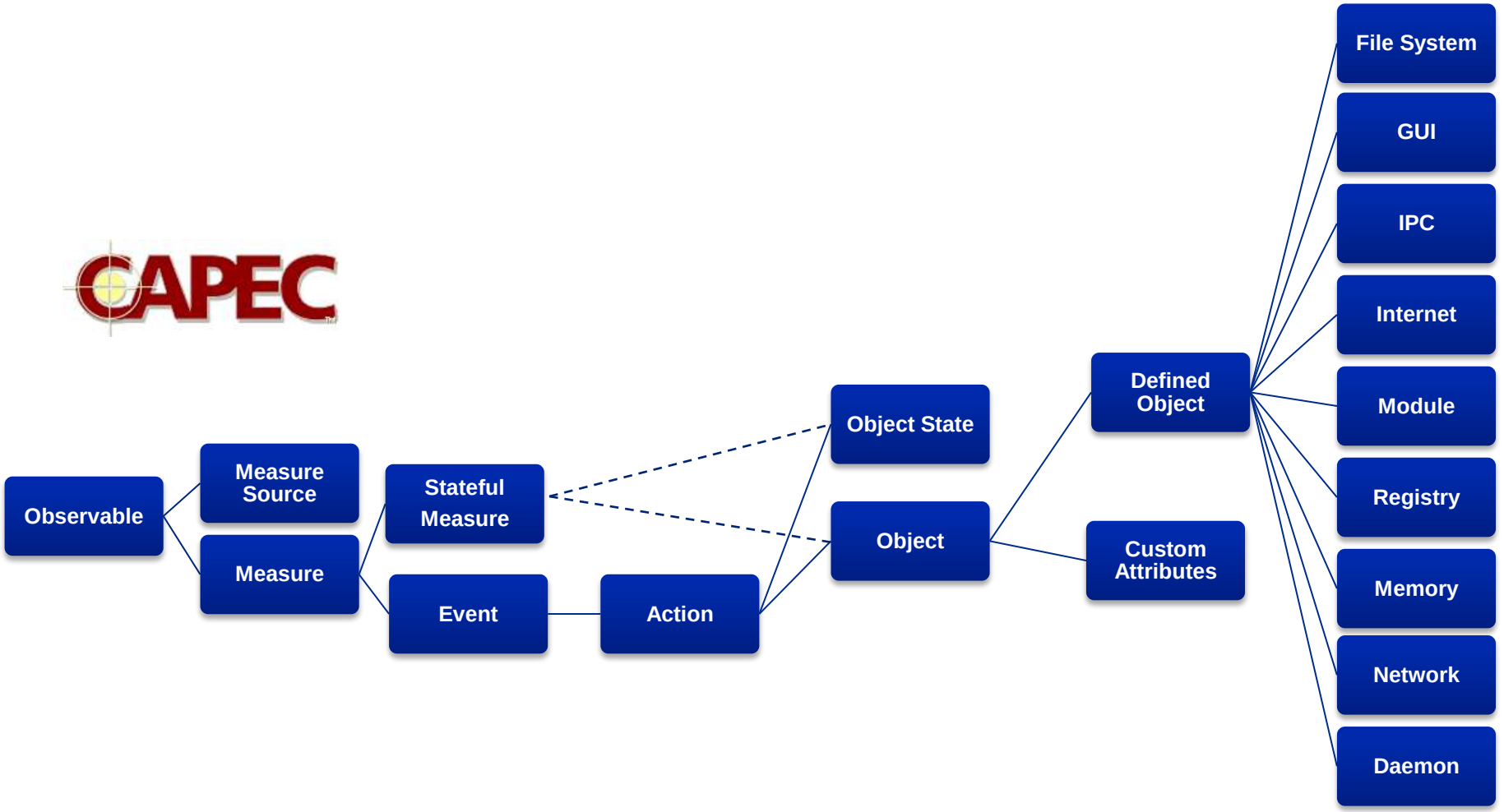
(more on the way)



# Wide Range of Cyber Observable Use Cases

- Potential ability to analyze data from all types of tools and all vendors
- Improved sharing among all cyber observable stakeholders
- Detect malicious activity from attack patterns
- Empower & guide incident management
- Identify new attack patterns
- Prioritize existing attack patterns based on tactical reality
- Ability to metatag cyber observables for implicit sharing controls
- Enable automated signature rule generation
- Enable new levels of meta-analysis on operational cyber observables
- Potential ability to automatically apply mitigations specified in attack patterns
- Etc….

# Common Cyber Observables (CybOX) Schema

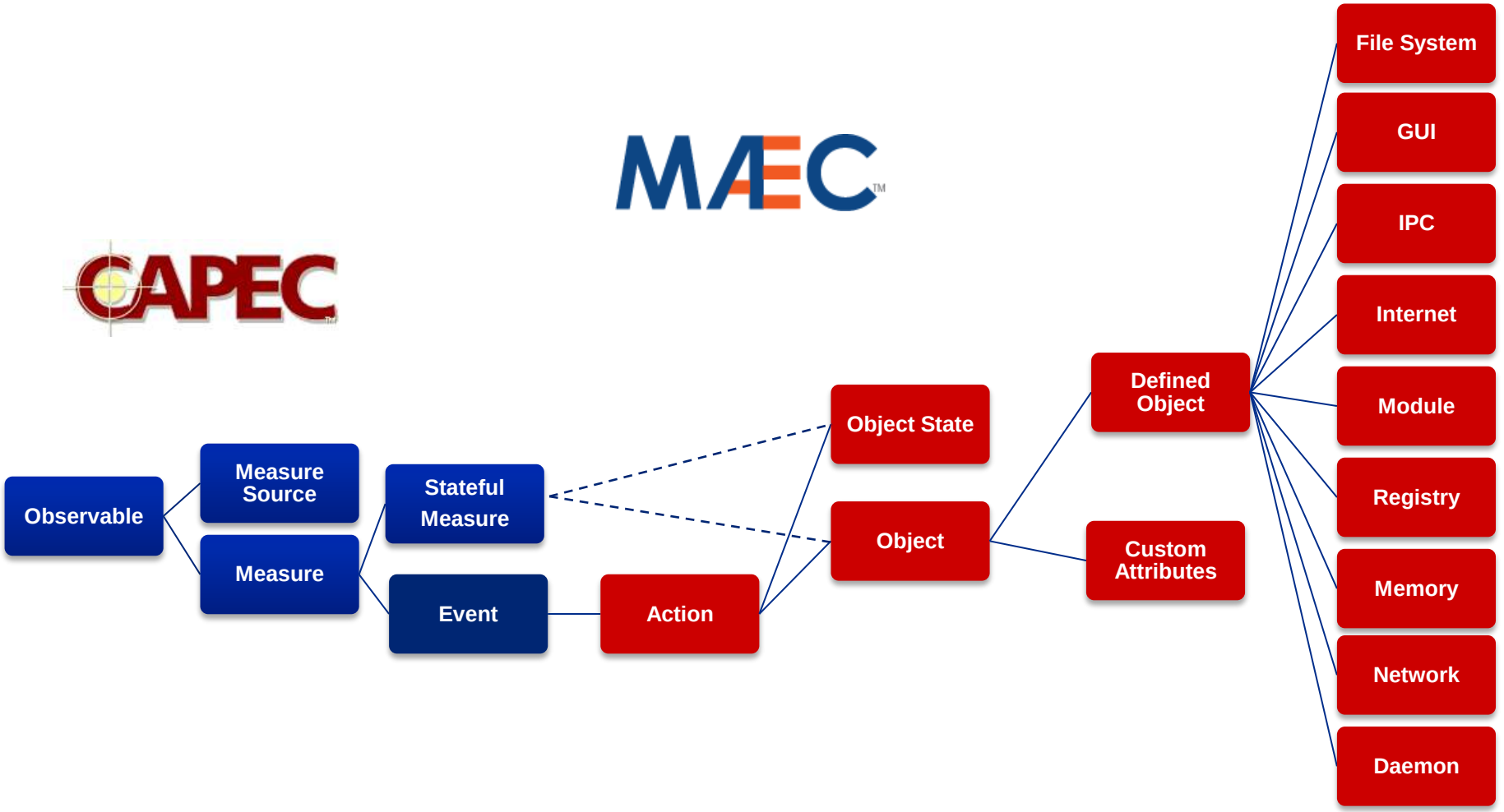





MITRE

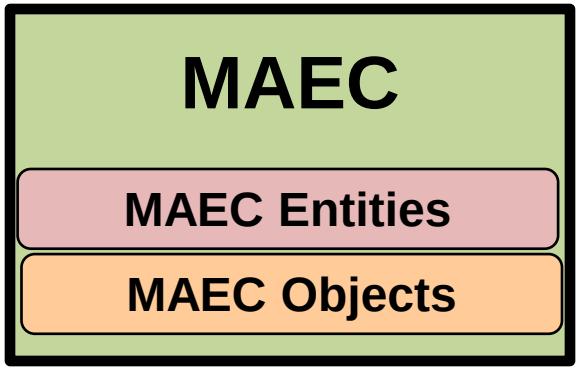
UNCLASSIFIED

# Common Cyber Observables (CybOX) Schema

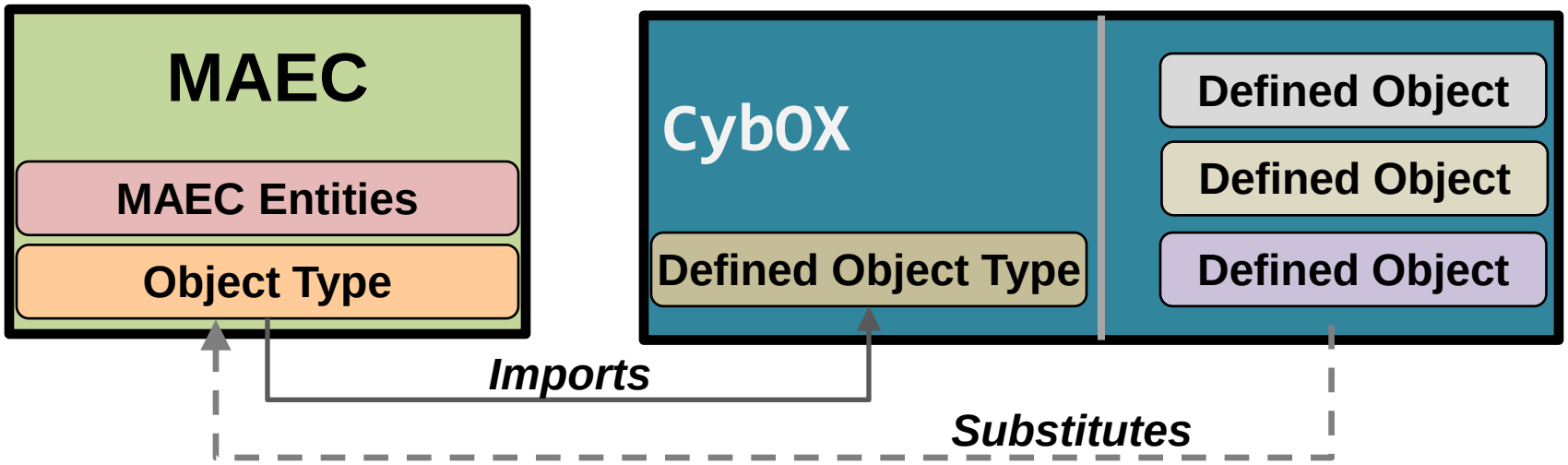


# MAEC & CybOX

- Before (MAEC 1.x)




- After (MAEC 2.0 and up)



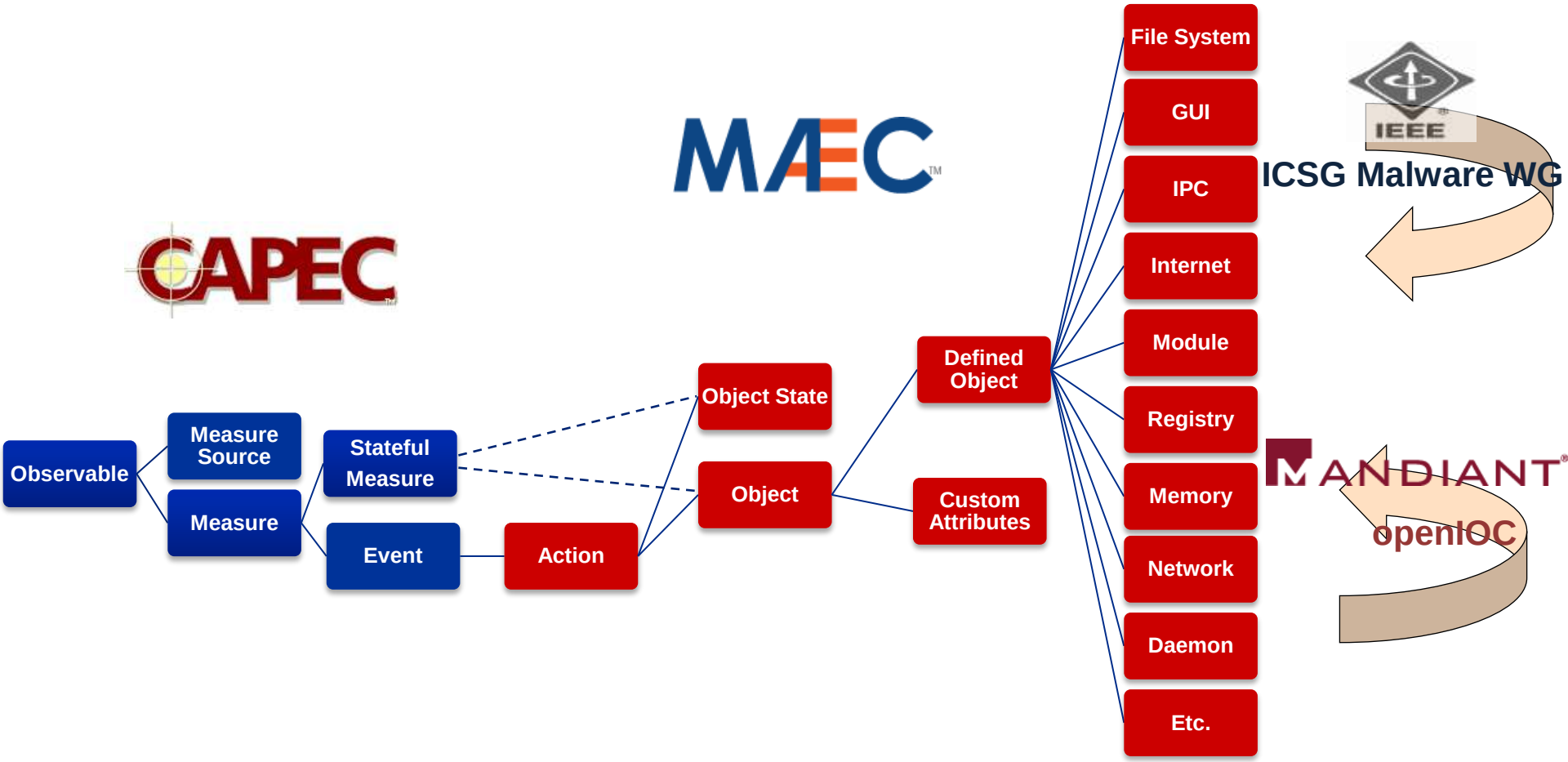



MITRE

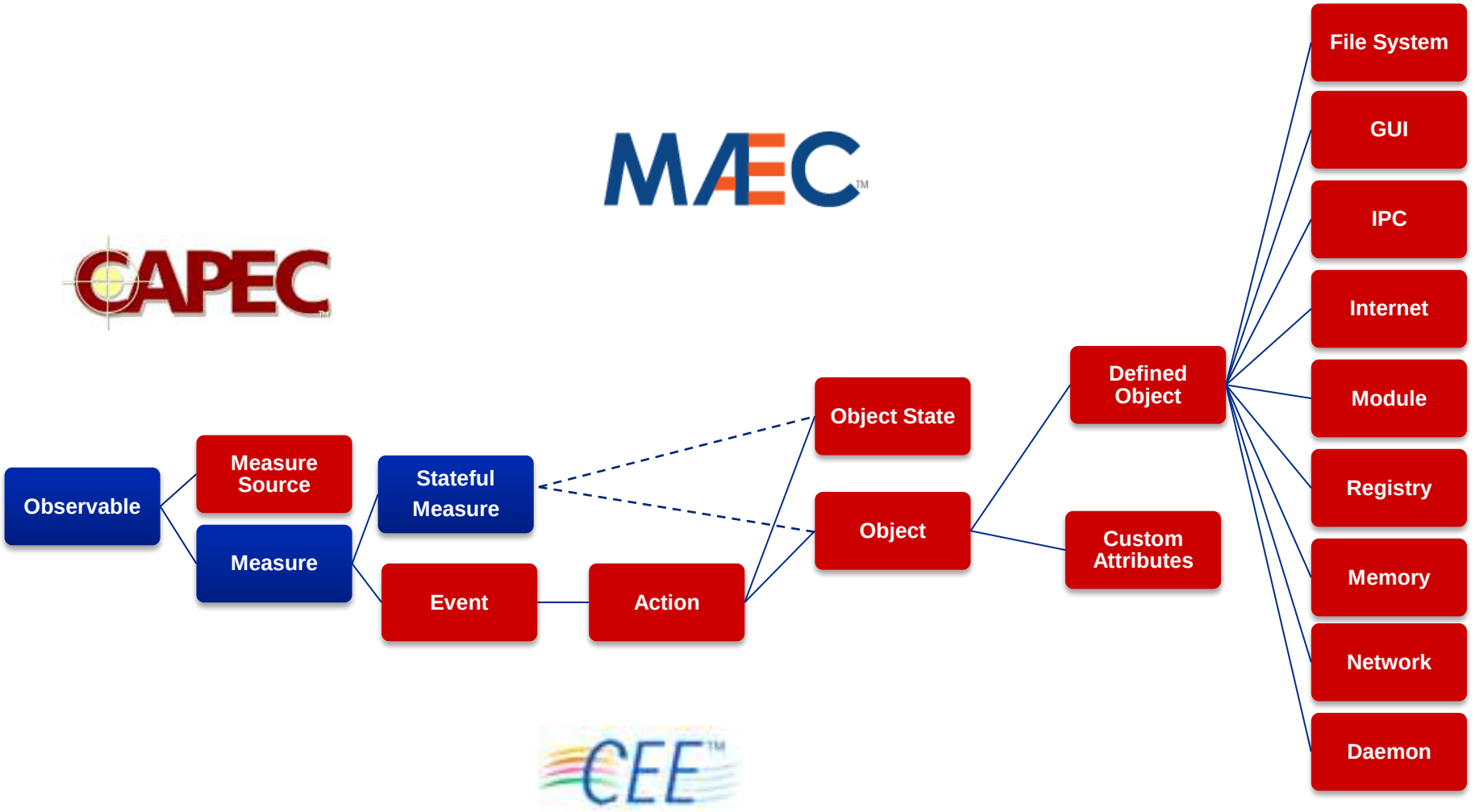
# Malware Community Engagement


- IEEE Industry Connections Security Group (ICSG)
  - Malware Working Group developed an exchange schema to facilitate the sharing of sample data between AV product vendors
    - MAEC currently imports the IEEE ICSG Malware Metadata exchange schema
  - Recently established Malware Metadata Exchange Format WG
    - Main Focus:
      - Adding capability to MMDEF schema for capturing blackbox behavioral metadata about malware
      - Will likely import MAEC/CybOX, especially MAEC Objects and Actions
    - Secondary Focus:
      - Adding capability to MMDEF schema for profiling clean (non-malicious) files, including software packages
      - Aimed at sharing information about clean files for reducing AV detection false positives
    - Potentially transition to a new IEEE standard

# Common Cyber Observables (CybOX) Schema



# Common Cyber Observables (CybOX) Schema



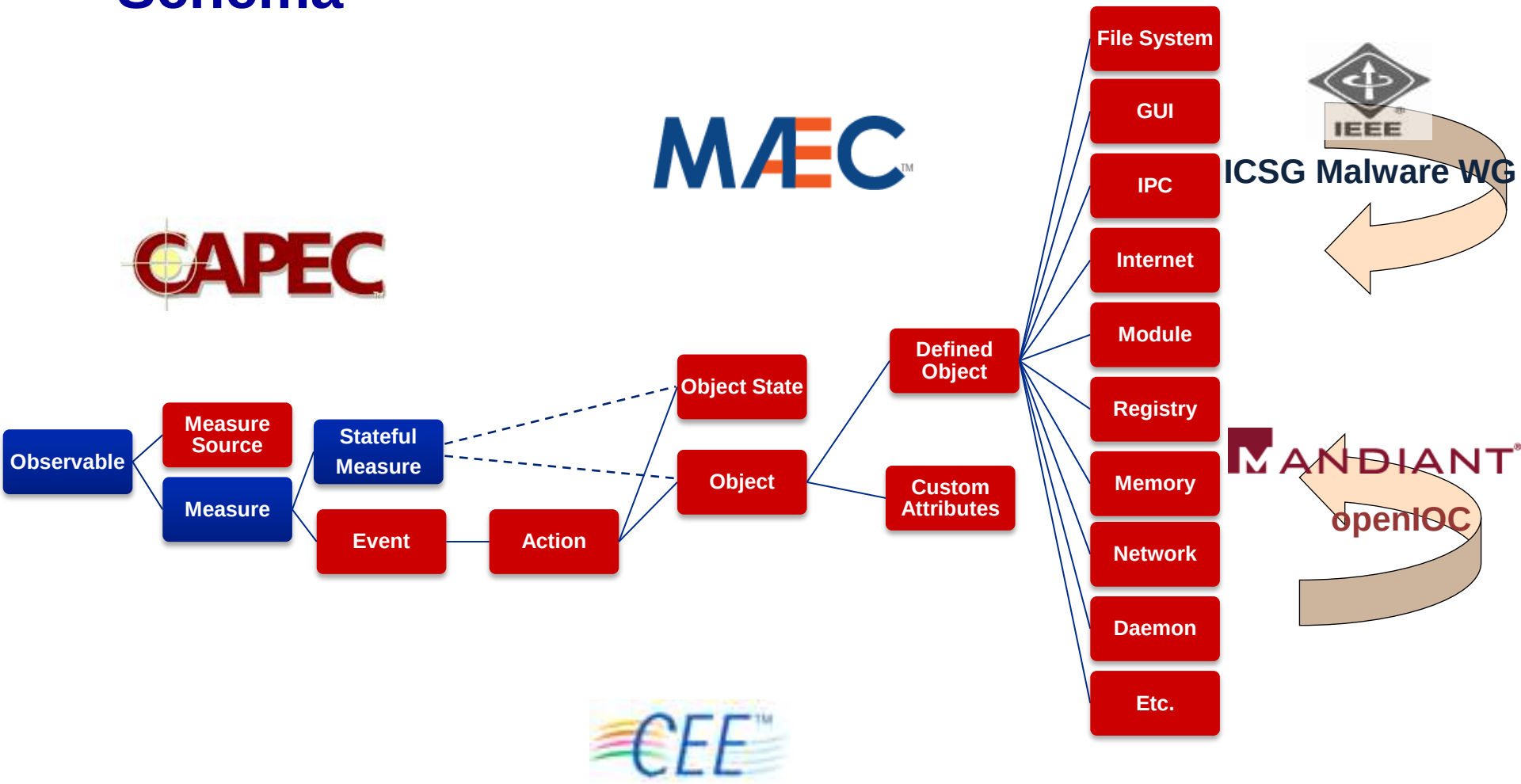
# Notional EMAP Components

- **Common Event Expression (CEE)**
  - **A suite of specifications to define taxonomy, syntax, transport, logging recommendations, and parsing information about event records**
- **Open Event Expression Language (OEEL)**
  - **A language to express parsing and normalization logic using CEE Profiles to convert event records into CEE**
- **Common Event Rule Expression (CERE)**
  - **A common format to express rules for pattern matching, filtering, and correlation**
- **Common Event Scoring System (CESS)**
  - **A specification that provides metrics of event severity and impact based on multiple factors**
- **Cyber Observable eXpression (CybOX)**
  - **A language to express cyber observable events or stateful measures that provides a common foundation for many of the other standards**

# The Role of CybOX within EMAP

- **Role of CybOX with CEE**
  - **Comprehensive structure of CybOX enables CEE to support full spectrum of event capture and sharing use cases that enterprise cyber security would require of an EMAP.**
  - **Common underlying structure would allow CEE events to be an integral and automatable part of holistic IR/IM.**
  - **CEE controlled vocabulary, taxonomy and object model benefits from a much broader and richer stakeholder community**
- **Role of CybOX with OEEL**
  - **CybOX offers architected structure for defining CEE Profiles and a strategically consistent basis for normalization and mapping from independent formats**
- **Role of CybOX with CERE**
  - **CybOX offers architected structure for defining automatable patterns and rules that are universally consistent and useful within EMAP and interchangeably with other automation protocols**

# Incident Response/Management and the Common Cyber Observables (CybOX) Schema



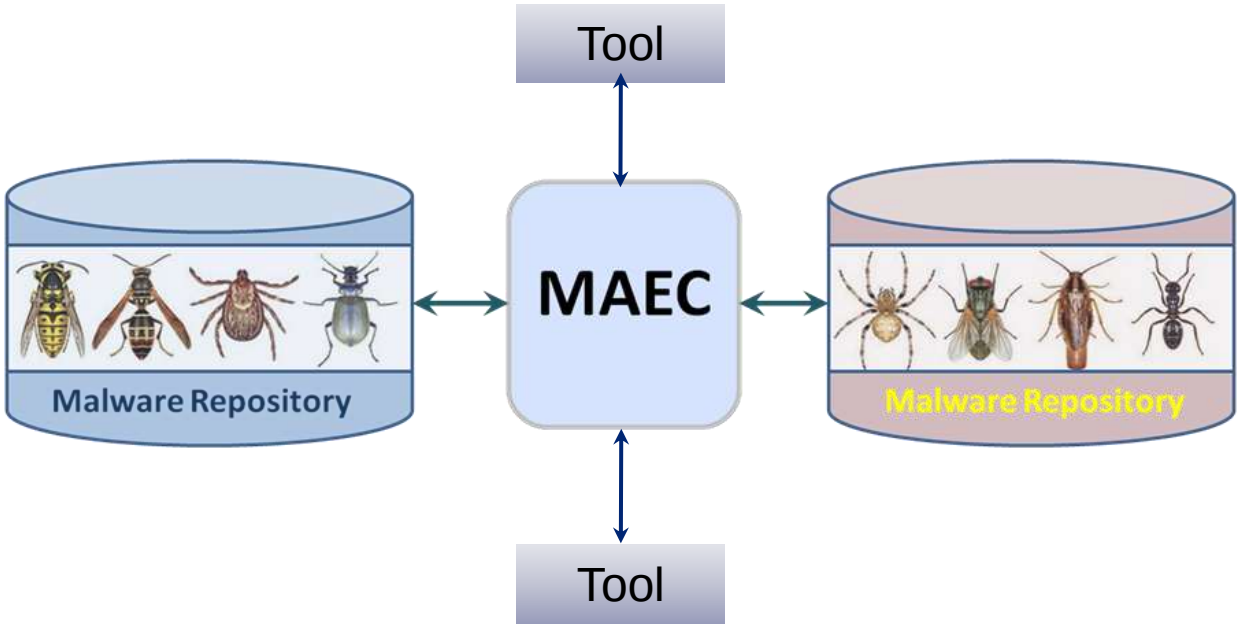
# Use Case: Malware Analysis

- **Current:**
  - **Difficult to combine different analysis perspectives or tools**
  - **Difficult to share info**
  - **Difficult to recognize if malware has been seen before**
  - **Does not scale well**

- **CybOX(MAEC)-enabled:**
  - **Easier to integrate different forms of analysis, different tools and even information from different sources**
  - **Easier to share information**
  - **Easier to recognize malware (including variants and pertubations)**
  - **Enables automated interaction among the various dimensions of malware analysis**

# MAEC(CybOX) Use Cases

- **Analysis**
  - **Help Guide Analysis Process**
  - **Standardized Tool Output**
  - **Malware Repositories**



```
<maec:MAEC_Bundle schema_version="2.0">
<maec:Behaviors>
 <maec:Behavior id="maec:storm:bhv:1" ordinal_position="1" successful="true">
  <maec:Description>A Storm Worm behavior that instantiates itself as a rootkit-enabled binary on a
system</maec:Description>
  <maec:Actions>
   <maec:Action id="maec:storm:act:1" ordinal_position="1" type="Create">
    <maec:Description>Storm first drops the malicious binary on the system</maec:Description>
    <maec:Affected_Objects>
     <maec:Affected_Object effect_type="Created" target_object_id="maec:storm:obj:1"/>
    </maec:Affected_Objects>
   </maec:Action>
   <maec:Action id="maec:storm:act:2" ordinal_position="2" type="Create">
    <maec:Description>Storm then instantiates the binary as a service</maec:Description>
    <maec:Affected_Objects>
     <maec:Affected_Object effect_type="Created" target_object_id="maec:storm:obj:2"/>
    </maec:Affected_Objects>
   </maec:Action>
  </maec:Actions>
 </maec:Behavior>
</maec:Behaviors>

 <maec:Objects>
  <maec:Object id="maec:storm:obj:1" Type="File">
   <observables:Defined_Object xsi:type="WinFileObj:Windows_File_Object_Type">
    <FileObj:File_Name>spooldr.exe</FileObj:File_Name>
    <FileObj:Full_Path>C:\Windows</FileObj:Full_Path>
   </observables:Defined_Object>
  </maec:Object>
  <maec:Object id="maec:storm:obj:2" Type="Key/Key Group">
   <observables:Defined_Object xsi:type="WinRegistryObj:Windows_Registry_Object_Type">
    <WinRegistryObj:Hive>HKEY_LOCAL_MACHINE</WinRegistryObj:Hive>
    <WinRegistryObj:Key>SYSTEM/CurrentControlSet\Services\spooldr</WinRegistryObj:Key>
   </observables:Defined_Object>
  </maec:Object>
 </maec:Objects>
</maec:MAEC_Bundle>
```

# Portion of MAEC Analysis of Storm Worm

# Use Case: Detect Malicious Activity

- **Current:**
  - **Manual effort to pull together data across many sensors**
    - **Results in limited situational awareness**
  - **Attack patterns and rules are typically too detailed (physical signatures) or ambiguous prose**
  - **High level of effort**
  - **High false negatives & positives**

- **CybOX-enabled:**
  - **Diverse set of sensors output data in common format**
  - **Attack patterns and rules can be defined in a uniform fashion**
  - **Pattern matching and analysis heuristics can be easily automated**

```
<Observables>
 <Observable>
    <Measure>
      <Pattern>
        <Event Type="File Ops (CRUD)">
          <Description>Storm drops a rootkit enabled binary on the system</Description>
          <Action Name="Create">
            <Object Type="File" Action_Role="Target">
              <Defined_Object xsi:type="WinFileObj:Windows_File_Object_Type">
                <FileObj:File_Name>spooldr.exe</FileObj:File_Name>
                <FileObj:Full_Path>C:\Windows</FileObj:Full_Path>
              </Defined_Object>
            </Object>
          </Action>
        </Event>
      </Pattern>
    </Measure>
 </Observable>
 <Observable>
   <Measure>
     <Pattern>
       <Event Type="Registry Ops">
         <Description>Storm creates a registry key to load itself as a service</Description>
         <Action Name="Create">
           <Object Type="Key/Key Group" Action_Role="Target">
             <Defined_Object xsi:type="WinRegistryObj:Windows_Registry_Object_Type">
               <WinRegistryObj:Hive>HKEY_LOCAL_MACHINE</WinRegistryObj:Hive>
               <WinRegistryObj:Key>SYSTEM/CurrentControlSet\Services\spooldr</WinRegistryObj:Key>
             </Defined_Object>
           </Object>
         </Action>
       </Event>
     </Pattern>
   </Measure>
 </Observable>
</Observables>
```

# CybOX Content Transformed from Portion of MAEC Analysis of Storm Worm

# Use Case: Malware Artifact Hunting

- **Current:**
  - **Very manual**
  - **Often imprecise and inconsistent**
  - **Localized**

- **CybOX(MAEC)-enabled:**
  - **Very automated**
  - **Consistent**
  - **Enables broad, non-localized sharing and hunting**

# Use Case: Host Based Detection

**Dynamic Analysis Engine**

- Anubis
- CWSandbox
- ThreatExpert
- Etc.

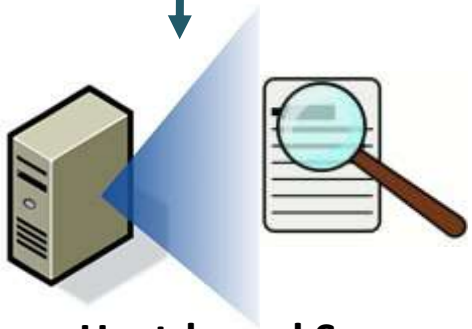
**Engine Output**

MAEC™

OVAL®

**Malware Binary**

**Sandbox -> MAEC Translator**



**Host-based Scanner**

# Real World Example: MAEC & Zeus Bot

Anubis Output*

Anubis → MAEC Translator Script

MAEC Output

```
MAEC_Bundle "1"
  Analyses
    Analysis "3"
      Subject
      Tools_Used
  Pools
    Action_Collection_Pool
      Action_Collection "2382"
      Action_Collection "2383"
      Action_Collection "2386"
      Action_Collection "2388"
      Action_Collection "2385"
    Object_Collection_Pool
      Object_Collection "2374"
      Object_Collection "2373"
      Object_Collection "2376"
      Object_Collection "2372"
      Object_Collection "2371"
```

OVAL Output

```
oval_definitions "http://oval.mitre.org/XMLSchema/oval-d
  generator MAEC XML to OVAL Script
    oval:product_name MAEC XML to OVAL Script
    oval:product_version 1.0
    oval:schema_version 5.7
    oval:timestamp 2010-11-08T09:26:33.347000
  definitions
  tests
    win-def:file_test "oval:maec_out:tst:1"
    win-def:file_test "oval:maec_out:tst:2"
    win-def:file_test "oval:maec_out:tst:3"
    win-def:file_test "oval:maec_out:tst:4"
    win-def:file_test "oval:maec_out:tst:5"
    win-def:file_test "oval:maec_out:tst:6"
    win-def:registry_test "oval:maec_out:tst:7"
    win-def:registry_test "oval:maec_out:tst:8"
    win-def:registry_test "oval:maec_out:tst:9"
    win-def:registry_test "oval:maec_out:tst:10"
  objects
    win-def:file_object "oval:maec_out:obj:1" C:\DO
    win-def:file_object "oval:maec_out:obj:2" C:\Doc
    win-def:file_object "oval:maec_out:obj:3" C:\Doc
    win-def:file_object "oval:maec_out:obj:4" C:\Doc
    win-def:file_object "oval:maec_out:obj:5" C:\Doc
    win-def:file_object "oval:maec_out:obj:6" C:\Doc
    win-def:registry_object "oval:maec_out:obj:7" H
    win-def:registry_object "oval:maec_out:obj:8" H
    win-def:registry_object "oval:maec_out:obj:9" H
    win-def:registry_object "oval:maec_out:obj:10" I
```

MAEC → OVAL Translator Script

Anubis Sandbox

Zeus Binary

*http://anubis.iseclab.org/?action=result&task_id=1167a57d1aa905e949df5d5478ab23bf9

```
<oval_definitions>
<definitions>
  <definition class="compliance" id="oval:storm:def:1" version="1">
    <metadata>
      <title>Storm Worm Test</title>
      <affected family="windows"/>
      <description>A test for a file and a registry key associated with the Storm Worm</description>
    </metadata>
    <criteria operator="AND">
      <criterion comment="File test" test_ref="oval:storm:tst:1"/>
      <criterion comment="Registry key test" test_ref="oval:storm:tst:2"/>
    </criteria>
  </definition>
 </definitions>
 <tests>
  <win-def:file_test id="oval:storm:tst:1" version="1" check="all" comment="test for a Storm file"
check_existence="all_exist">
    <win-def:object object_ref="oval:storm:obj:1"/>
  </win-def:file_test>
  <win-def:registry_test id="oval:storm:tst:2" version="1" check="all" comment="test for a Storm registry key"
check_existence="all_exist">
    <win-def:object object_ref="oval:storm:obj:2"/>
  </win-def:registry_test>
 </tests>
 <objects>
  <win-def:file_object id="oval:storm:obj:1" version="1">
    <win-def:path>C:\System</win-def:path>
    <win-def:filename>spooldr.exe</win-def:filename>
  </win-def:file_object>
  <win-def:registry_object id="oval:storm:obj:2" version="1">
    <win-def:hive>HKEY_LOCAL_MACHINE</win-def:hive>
    <win-def:key>SYSTEM/CurrentControlSet\Services\spooldr</win-def:key>
    <win-def:name xsi:nil="true"/>
  </win-def:registry_object>
 </objects>
</oval_definitions>
```

# OVAL Checks Transformed from Portion of MAEC Analysis of Storm Worm

# Use Case: Incident Response Data Capture

- **Current:**
  - **Very manual**
  - **Inconsistent between analysts & organizations**
  - **Prose-based and imprecise**
  - **Difficult to automate capture and actionable alerts**

- **CybOX-enabled:**
  - **Improved consistency**
  - **Ability to tie everything together**
  - **Simplified and automated data capture**
  - **Alerts become actionable for automation**

# Use Case: IR/IM Alerts

- **Current:**
  - **Typically unstructured prose**
  - **Labor intensive and slow**
  - **Limited actionable (in an automated fashion) data**

- **CybOX-enabled:**
  - **Structured and consistent**
  - **Alert generation can be much faster and less labor intensive**
  - **Potentially actionable in an automated context**

# Notional Flow of a Modern Security Incident

1. An attack on an information system occurs involving social engineering, vulnerability exploit, malware + command and control (C2).
2. CybOX-enabled operational sensors (IDS, host-based, etc.) pick up anomalous activity and report it in CEE/CybOX formats.
3. Automated analysis tools & rules attempt to match anomalous activity against CybOX-adorned CAPEC attack patterns but discover no matching patterns.
4. Incident is reported – Incident Response/Management process is initiated.
5. IR personnel capture discovered detail of incident in CybOX-compliant formats, including CEE.
6. IR personnel detect malware as part of the ongoing attack.

# Notional Flow of a Modern Security Incident (cont.)

7. Malware undergoes automated analysis (dynamic and/or static) and results are captured in MAEC (CybOX-integrated) language.
8. Malware analysts are able to correlate the current malware instance with a broad range of pre-existing malware samples and analysis data from MAEC-enabled repositories and zoos.
9. Malware analysts capture new discovered detail of the malware in MAEC format, including the CWE or CVE exploited .
10. Sample and analysis data from current malware instance are entered into appropriate malware repositories and zoos.
11. CybOX observables of malware effects on hosts are extracted from MAEC content to generate OVAL checks to determine if any given host has been infected/affected by the current malware instance.
12. OVAL checks are distributed and run against other areas of the domain or enterprise to determine breadth of compromise.

# Notional Flow of a Modern Security Incident (cont.)

13. IR/IM personnel apply appropriate mitigations/remediations to negate the effects of the attack.
14. A new CAPEC attack pattern is authored to describe this new observed attack behavior, and is adorned as appropriate with CybOX content observed for this pattern in the operational space.
15. IR/IM personnel issue relevant alerts for the observed incident including the new CAPEC pattern, MAEC bundle and related CEE/CybOX content.
16. Secure development takes advantage of this new CAPEC pattern to: define/refine appropriate security policy, training & requirements; guide security engineering (control selection), architectural risk analysis, secure code review and security testing; identify relevant CWE weaknesses, CVE vulnerabilities & CCE configuration issues; prioritize relevant CAPEC patterns based on real-world observed prevalence/frequency profiled through automated observation of CybOX patterns in the operational space .

# Where is CybOX today?

- Currently integrated into CAPEC
- Currently integrated into MAEC
- In process of being integrated into CEE
- Part of the strategic approach for EMAP
- Part of the strategic vision for IR/IM with US-CERT
- Continued integration discussions planned for Mandiant OpenIOC once initial drafts of Object schemas are complete
- Currently being evaluated for integration into multiple research projects
- Website should be up soon

# Timeline

- Initial CybOX Schemas released with CAPEC v1.6
- CEE v0.6 Released
  - Have some internal mappings to CybOX
  - Formalized, released in next update
- MAEC 2.0 update leveraging CybOX coming out within a few weeks
- Revised CybOX to be released EOFY11
  - Support limited OS & host-based objects
  - Limited or no network observables

# Questions / Comments?


Sean Barnum

sbarnum@mitre.org

ITSAC (Oct 31 - Nov 2) – Crystal City