# Supporting *Secure Software Operations*

# Secure Software Operations

- Where secure development use cases required foundational knowledge and ways to package it and understand it within a static context, Secure Software Operations requires situational awareness & interpretation of foundational knowledge within a dynamic context
- Considering that secure operations is a key element of overall software assurance we need ways to:
  - Bridge the secure development and secure operations domains
  - Improve the analysis, characterization, collection, discovery & knowledge sharing of malware
  - Combine elements of the ecosystem as practical applications to support secure software operations
- This portion of the tutorial will focus on resources/efforts focused at addressing these three needs

# Secure Software Operations

- **Bridge the secure development and secure operations domains**
- **Improve the analysis, characterization, collection, discovery & knowledge sharing of malware**
- **Combine elements of the ecosystem as practical applications to support secure software operations**

Cyber Observable eXpression (CybOX)

Malware Attribute Enumeration & Characterization (MAEC)

Security Content Automation Protocol (SCAP) and other Automation Protocols

# The Balance of Secure Development and Secure Operations in the Software Security Equation

Sean Barnum

Software Assurance Principal
sbarnum@mitre.org

MITRE

# The Software Security Equation

- **Software security is about reducing the risk that software poses to those who use it or are affected by it.**
- **This requires thought and action more than simply at the point of development or use.**
- **It requires a more holistic approach, balancing secure development and secure operations.**
- **Bad news: these two capable domains typically do not interact much or understand each other.**
- **Good news: there are active ongoing efforts focused on addressing this gap.**

SS = Risk reduction

SS != SD or SO

SS = SD and SO

# Secure Development

- The objective of security in development is to prevent security issues in the software causing vulnerability.
- Best case, this means preventing such security issues from ever entering the software to begin with. (early lifecycle)
- Worst case, this means at least preventing such security issues from ever being fielded into live systems. (late lifecycle)

# Secure Operations

- The objective of security in operations is to prevent security issues in deployed systems by securing their *infrastructure, configuration, and use.*
- Beyond the initial security engineering of software operational deployment, the bulk of secure software operations is about continuous situational awareness and incident response.

# Foundational Questions of Secure Operations

- Are we being attacked? (Were we attacked?)
- How are we being attacked?
- What is the objective of the attack?
- What is our exposure?
- Who is attacking us?
- What should we do to protect against these attacks in the future?

## Mechanisms of Secure Development

- Effective Security Training
- Security Policy
- Security Requirements
- Secure Architecture & Design
- Secure Coding
- Security Testing
- Penetration Testing

## Mechanisms of Secure Operations

- Secure Configurations
- Firewalls
- Proxies
- Anti-Tamper (AT) Mechanisms
- Intrusion Detection Systems (IDS)
- Intrusion Prevention Systems (IPS)
- Real-time Data Monitoring
- Operational Monitoring and Control
- Incident Response
- Forensics

# Commonality of Attack

- **The commonality between the secure development and secure operations domains is the central role of understanding how adversaries attack software.**
- **The secure development domain needs to understand the attacker's perspective in *abstract terms* in order to improve security across a wide range of contexts, rather than individual instances.**
- **The secure operations domain needs to understand the attacker's specific variations of behavior in *gory detail* in order to recognize it, understand it, estimate its effect, and plan its mitigation.**
- **Reciprocal balance between the top-down perspective of secure development and the bottom-up perspective of secure operations yields opportunity for mutual benefit.**

# Attack Patterns

- Given the differing requirements between the two domains (to characterize attacks and potentially exchange this information), a flexible mechanism is required to capture, describe, and share knowledge about common patterns of attack.
- The attack pattern concept represents a description of common attack approaches abstracted from a set of known real-world exploits.
- Attack pattern object as specified and leveraged by the Common Attack Pattern Enumeration and Classification (CAPEC) - http://capec.mitre.org

# Common Attack Pattern Enumeration and Classification (CAPEC)

- Community effort targeted at:
  - Standardizing the capture and description of attack patterns
  - Collecting known attack patterns into an integrated enumeration that can be consistently and effectively leveraged by the community
  - Gives you an attacker's perspective you may not have on your own
  - Initially, attack-centric testing methods, now integrating with operations and malware
- Excellent resource for many key activities
  - Abuse Case development
  - Architecture attack resistance analysis
  - Risk-based security/Red team penetration testing
  - Whitebox and Blackbox testing correlation
  - Operational network observation correlation
- Where is CAPEC today?
  - http://capec.mitre.org
  - Currently 386 patterns, stubs, named attacks
  - Future plans
    - New patterns
    - Refine existing patterns for quality and consistency
    - Formalize patterns to finer granularity to support bridging with the malware and incident response communities

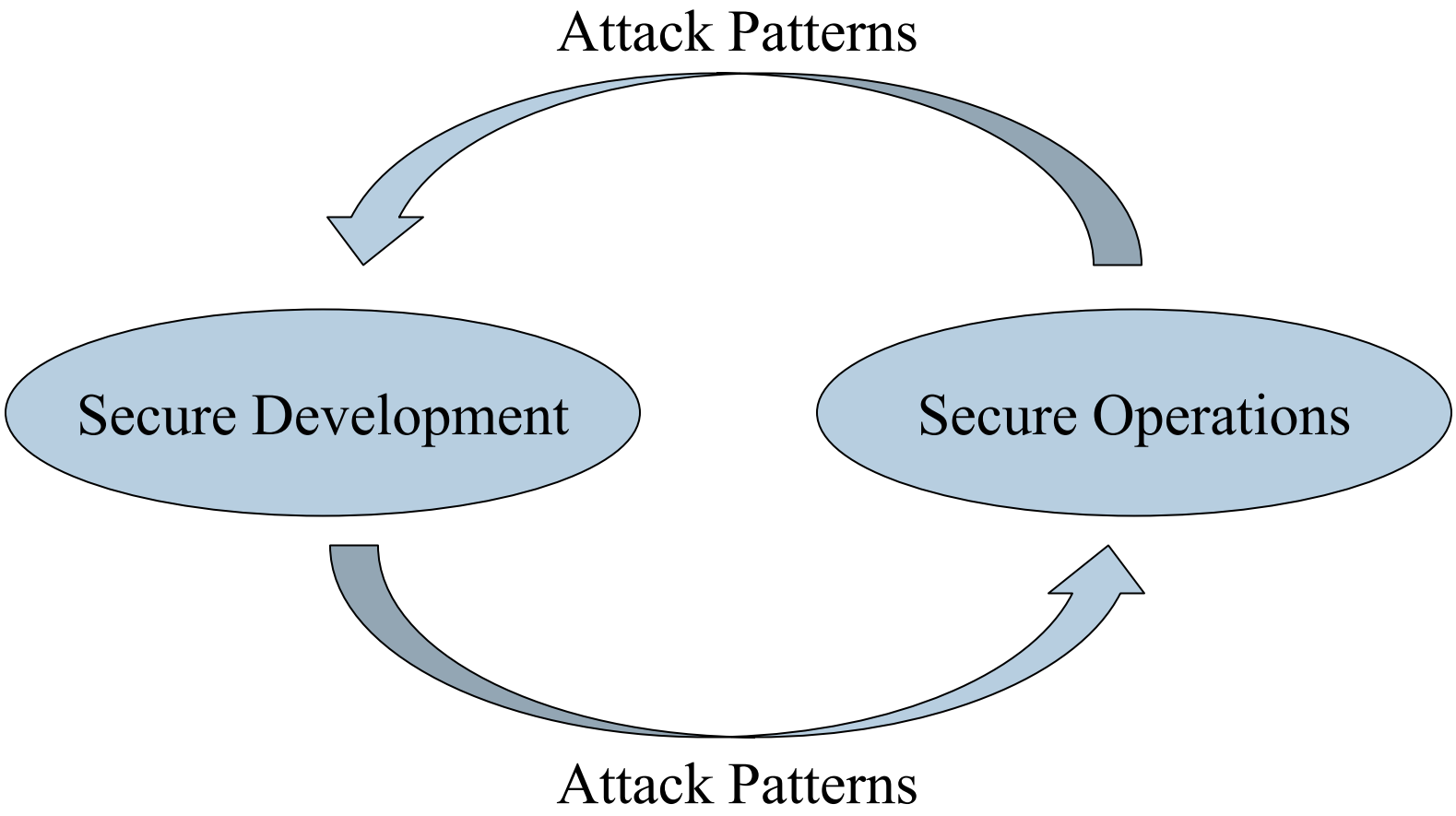
# Attack Pattern Value to Secure Development

- **While this source of raw data comes primarily from the secure operations domain, attack patterns today are primarily a construct used by the secure development community to aid software developers in improving the assurance profile of their software.**
- **Attack patterns offer the secure development community unique value in several areas such as:**
  - **Representing abuse cases (how an attacker would intentionally abuse a software system) during requirements elicitation, specification, and review.**
  - **Mapping identified threats to the software's modeled attack surface as part of threat modeling activities during architecture and design.**
  - **Guiding and prioritizing secure code analysis during implementation. This includes identifying specific high-risk areas requiring greater analysis rigor as well as the most relevant weaknesses to look for.**
  - **Identifying, specifying, and prioritizing security test cases.**
  - **Serving as attack templates for penetration testing and objective persona descriptors for red team penetration testing.**

# Attack Pattern Value to Secure Operations

- The secure operations community can utilize CAPEC to assist in situational awareness of deployed systems under attack and aid in response and mitigation.
- Several characteristics of attack patterns make them relevant for the secure operations community:
  - Attack patterns provide high-level rather than simply low-level detailed patterns of attacks against software. Much of secure operations is about analyzing low-level activity for patterns and composing them into higher levels of abstraction to detect, identify, and respond to attacks.
  - Software assurance attack patterns provide a top-down, high-level context for both the method and the intent of attacks.
  - Efforts are currently under way to formalize the CAPEC attack pattern schema in order to provide adequate detail of attacks for aligning and integrating their context with bottom-up incident analysis characterizations.

# Attack Patterns Bridge Secure Development and Operations



# Secure Operations Knowledge Offers Unique Value to Secure Development

- Using attack patterns makes it possible for the secure development domain to leverage significant value from secure operations knowledge, enabling them to:
  - Understand the real-world frequency and success of various types of attacks.
  - Identify and prioritize relevant attack patterns.
  - Identify and prioritize the most critical weaknesses to avoid.
  - Identify new patterns and variations of attack.

# Secure Development Knowledge Offers Unique Value to Secure Operations

- Attack patterns enable those in the secure operations domain to provide appropriate context to the massive amounts of data analyzed to help answer the foundational secure operations questions.

# Attack Patterns Help Answer Foundational Questions Regarding Secure Operations

| Question | Role of Attack Patterns |
|---|---|
| Are we being attacked? (Were we attacked?) | Attack patterns offer structured descriptions of common attacker behaviors to help interpret observed operational data and determine its innocent or malicious intent. |
| How are we being attacked? | Attack patterns offer detailed structured descriptions of common attacker behavior to help interpret observed operational data and determine exactly what sort of attack is occurring. |
| What is the objective of the attack? | Elements of attack patterns outlining attacker motivation and potential attack effects can be leveraged to help map observed attack behaviors to potential attacker intent. |
| What is our exposure? | The structure detail and weakness mapping of attack patterns can provide guidance in where to look and what to look for when certain attack pattern behaviors are observed. |
| Who is attacking us? | Attack pattern threat characterization and detailed attack execution flow can provide a framework for organizing real-world attack data to assist in attribution. |
| What should we do to prevent against attacks in the future? | Attack patterns offer prescriptive guidance on solutions and mitigation approaches that can be effective in improving the resistance tolerance and/or resilience to instances of a given pattern of attack. |

# So, this all sounds great but how do we map these high-level attack pattern abstractions to the low-level operational world?

## Cyber Observables

### The Secret Sauce for Bridging the Abstract to the Concrete

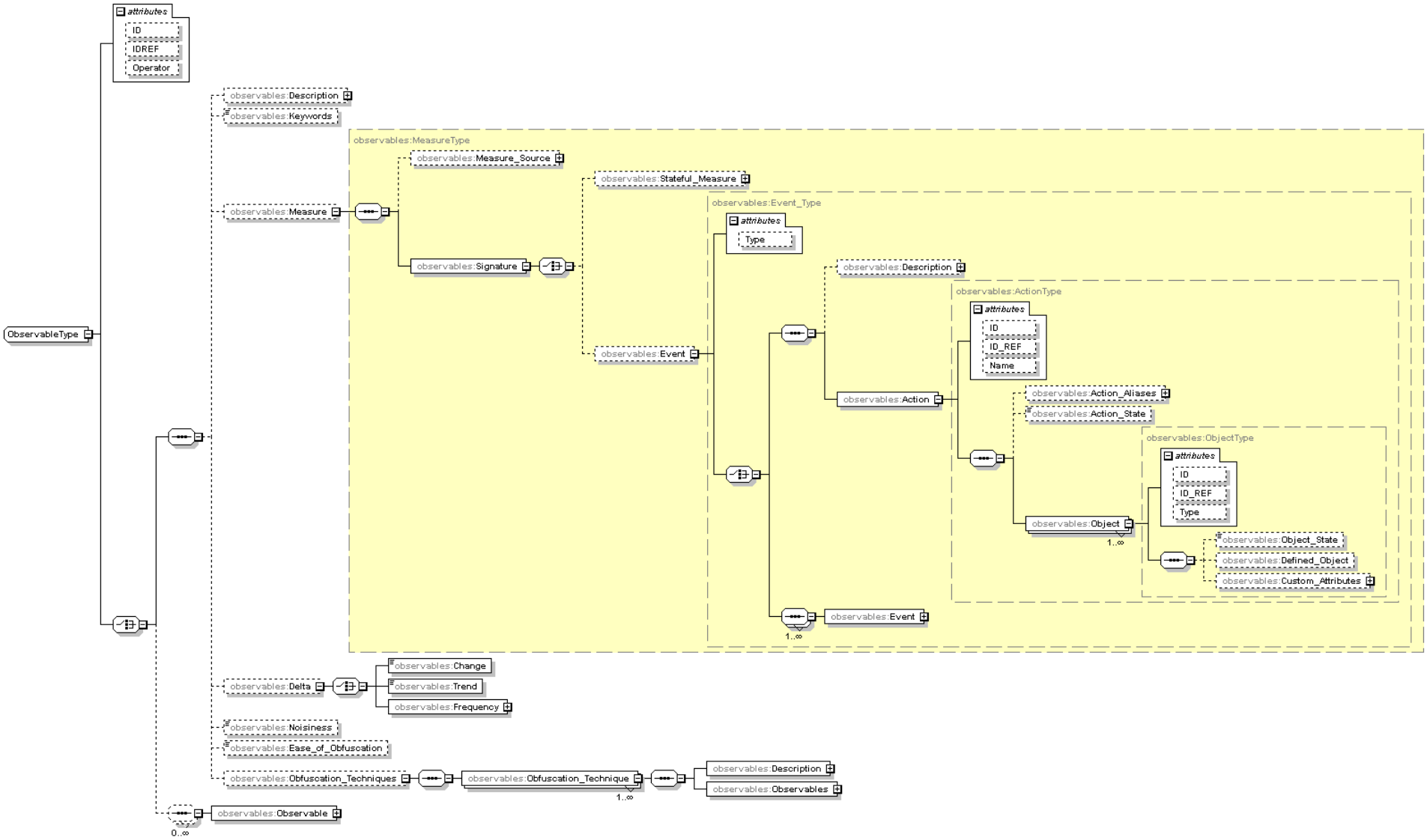
# Cyber Observables Overview

- The Cyber Observables construct is intended to capture and characterize events or properties that are observable in the operational domain.
- These observable events or properties can be used to adorn the appropriate portions of the attack patterns in order to tie the logical pattern constructs to real-world evidence of their occurrence or presence.
- This construct has the potential for being the most important bridge between the two domains, as it enables the alignment of the low-level aggregate mapping of observables that occurs in the operations domain to the higher-level abstractions of attacker methodology, motivation, and capability that exist in the development domain.
- By capturing them in a structured fashion, the intent is to enable future potential for detailed automatable mapping and analysis heuristics.

# A Brief History of Cyber Observables

- **September 2009: Concept introduced to CAPEC in Version 1.4 as future envisioned 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**

# Simplified Overview of Current Schema




Generated by XMLSpy www.altova.com

# Cyber Observable Broader Use Cases

- Detect malicious activity from attack patterns
- Empower & guide incident management
- Identify new attack patterns
- Prioritize existing attack patterns based on tactical reality

- Potential ability to analyze data from all types of tools and all vendors
- Improved sharing among all cyber observable stakeholders
- 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….

# Summary

- Effective software security requires a balanced approach between secure development and secure operations.
- The commonality between these two domains is the central role of understanding how adversaries attack software.
- CAPEC attack patterns offer a mechanism for structured characterization of common attacks that enable a useful exchange of information relevant to both domains, also aligning low-level observations to high-level contexts for mutual benefit.
- CAPEC is currently a resource leveraged primarily by the secure development community, but there is an opportunity and a strong need for increased collaboration from the secure operations community.
- Collaboration from secure operations & the introduction of structured cyber observables will help shape and refine CAPEC to more effectively serve both communities, potentially acting as an integrating bridge to eventually yield a more holistic software security capability.

# Questions?

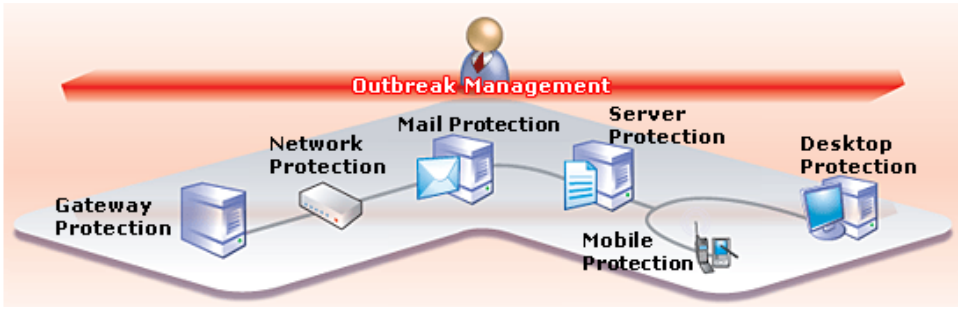
**The topic and content covered in this presentation was published as an article in the Sep/Oct 2010 issue of CrossTalk: The Journal of Defense Software Engineering**


Penny Chase

Ivan Kirillov – Desiree Beck – Robert Martin – Sean Barnum

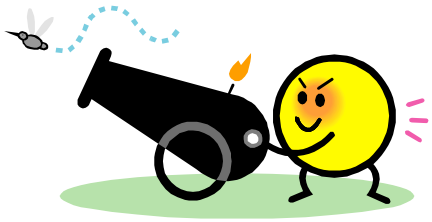
# Why Do We Need to Develop Standards for Malware?

Multiple layers of protection



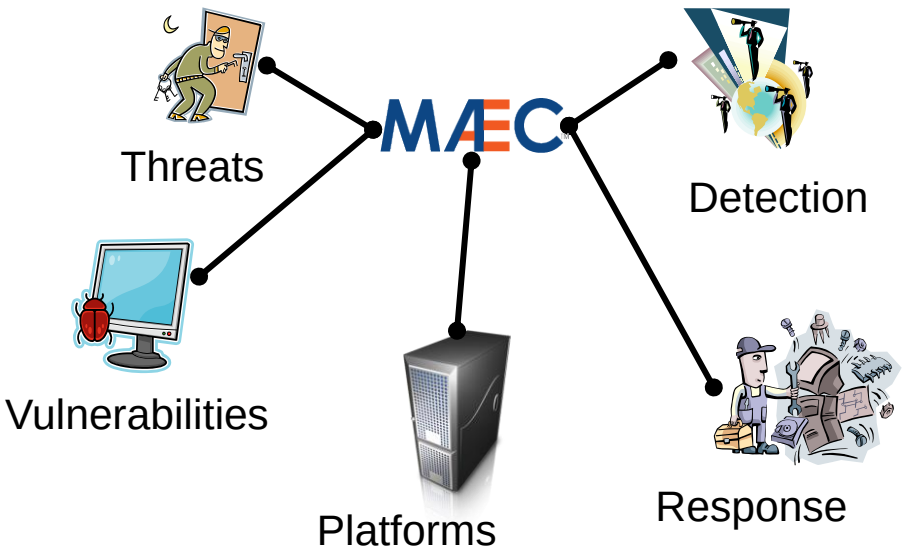

Lots of products


Inconsistent reports



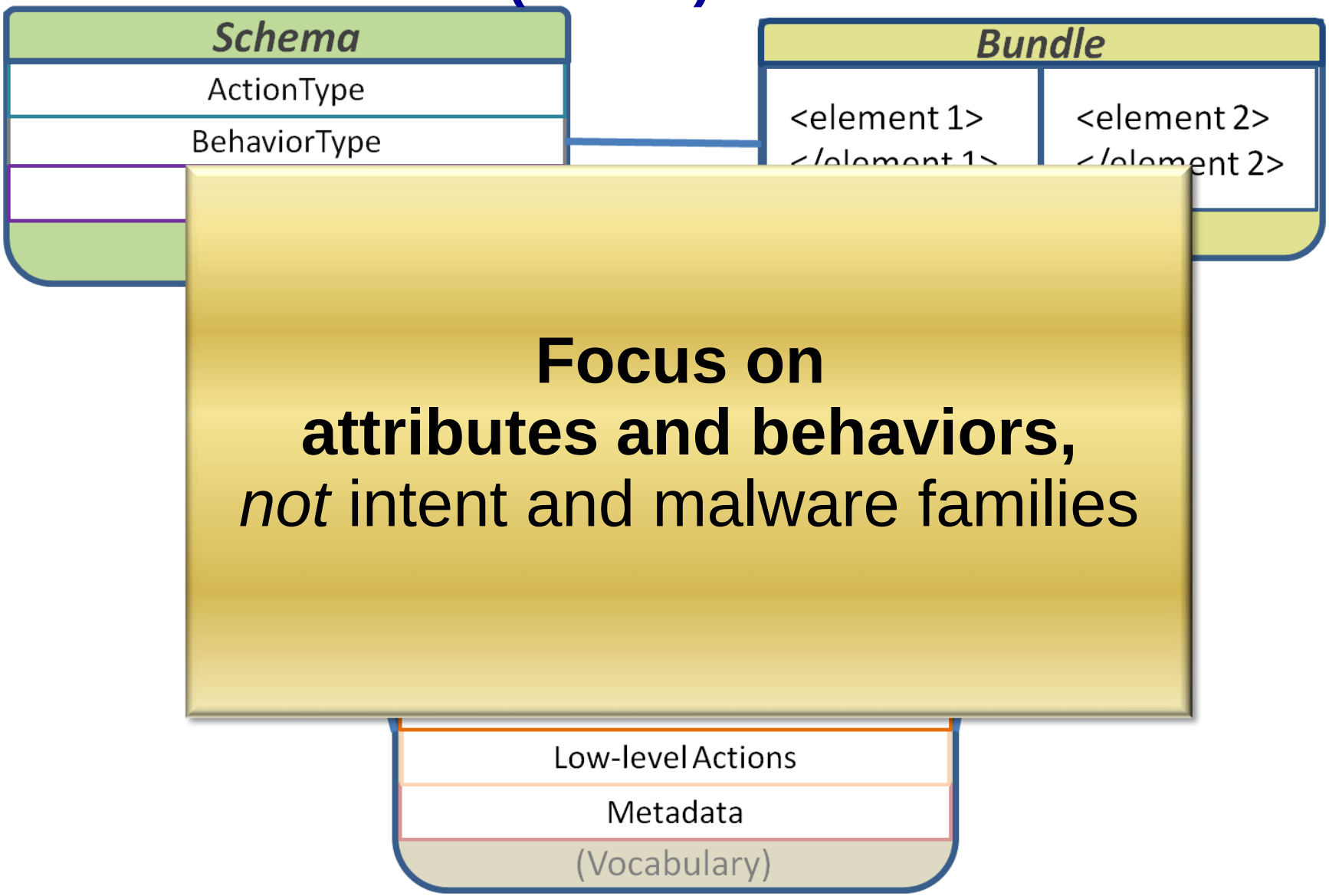
There's an arms race

# Malware Attribute Enumeration and Characterization (MAEC)



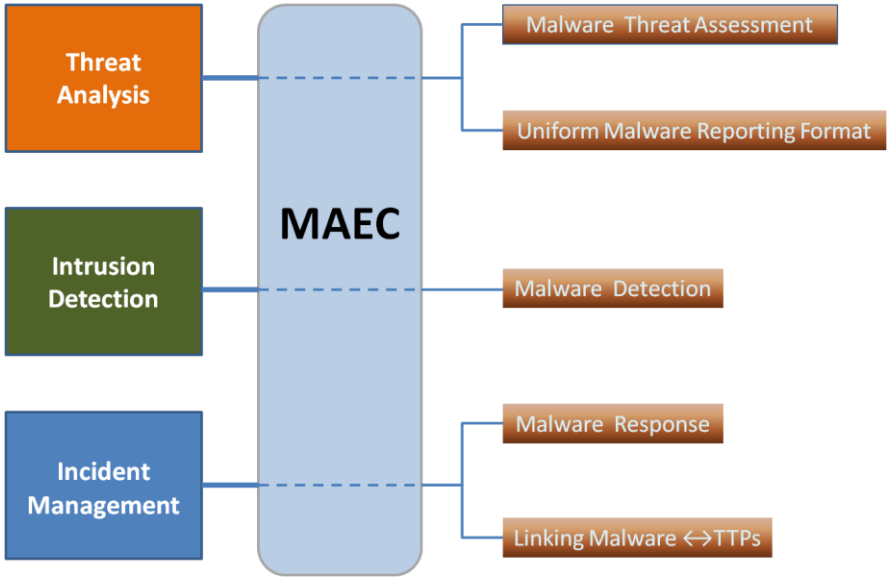

- **Language for sharing structured information about malware**
  - **Grammar (Schema)**
  - **Vocabulary (Enumerations)**
  - **Collection Format (Bundle)**
- **Focus on attributes and behaviors**
- **Enable correlation, integration, and automation**

# Malware Attribute Enumeration and Characterization (MAEC)

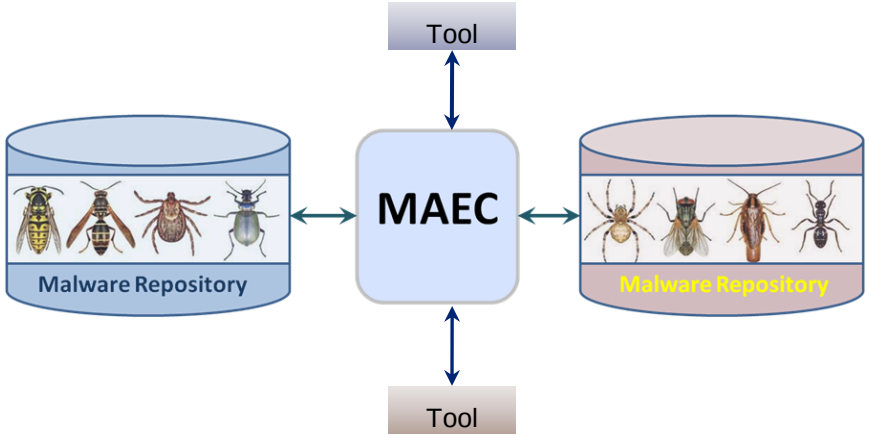


# MAEC Use Cases

- **Operational**




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

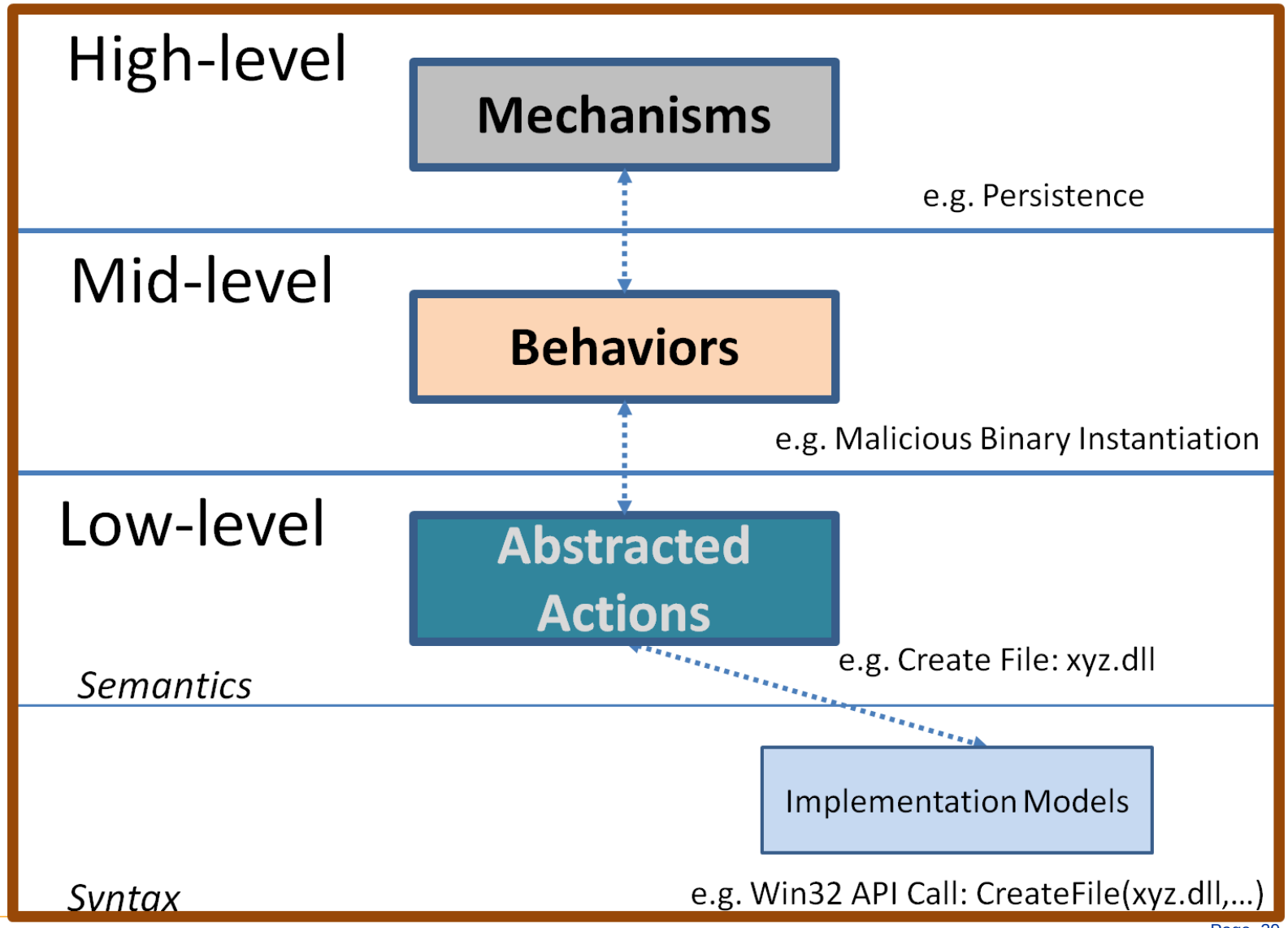





MITRE

UNCLASSIFIED

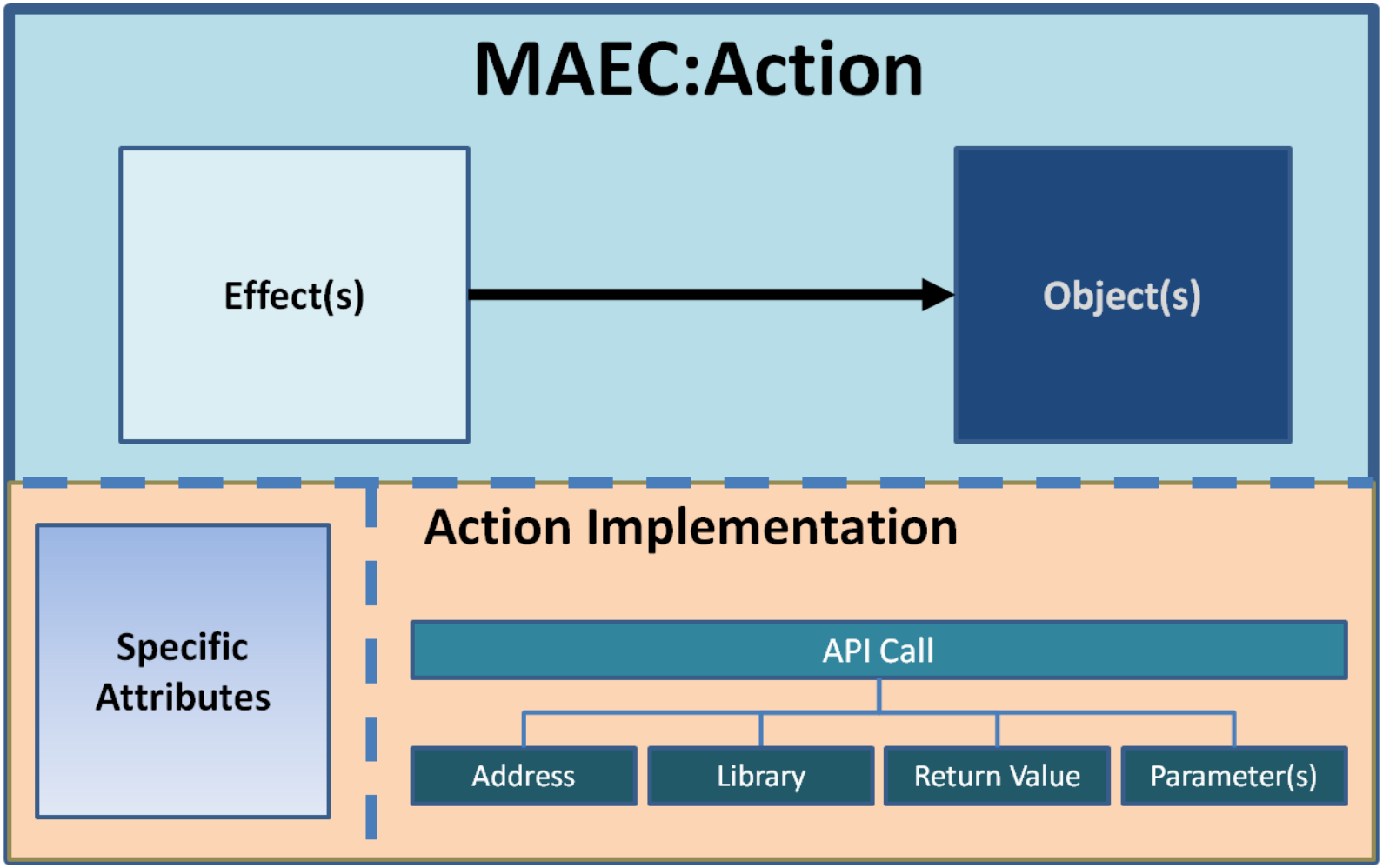
# MAEC Overview



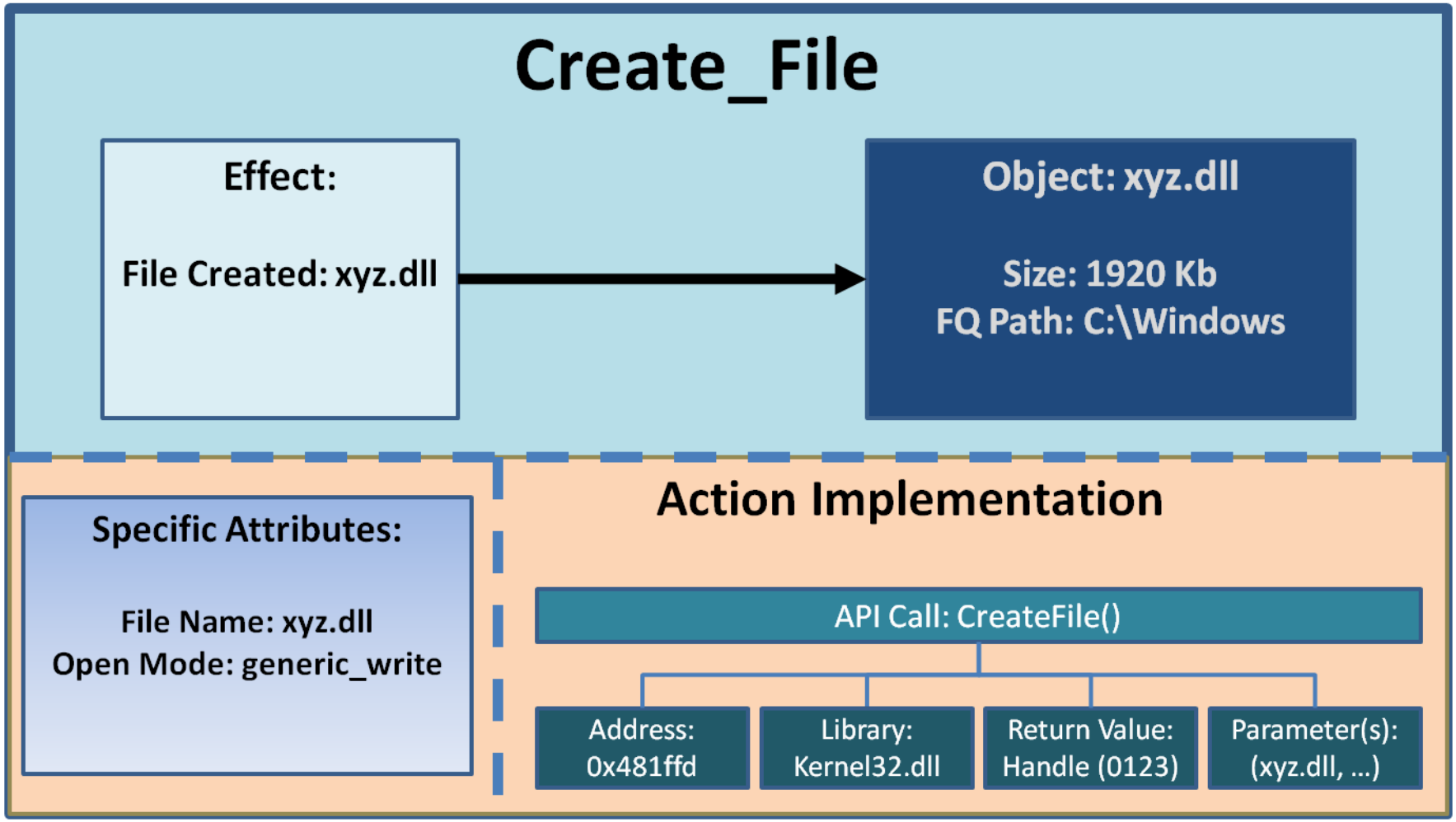



MITRE

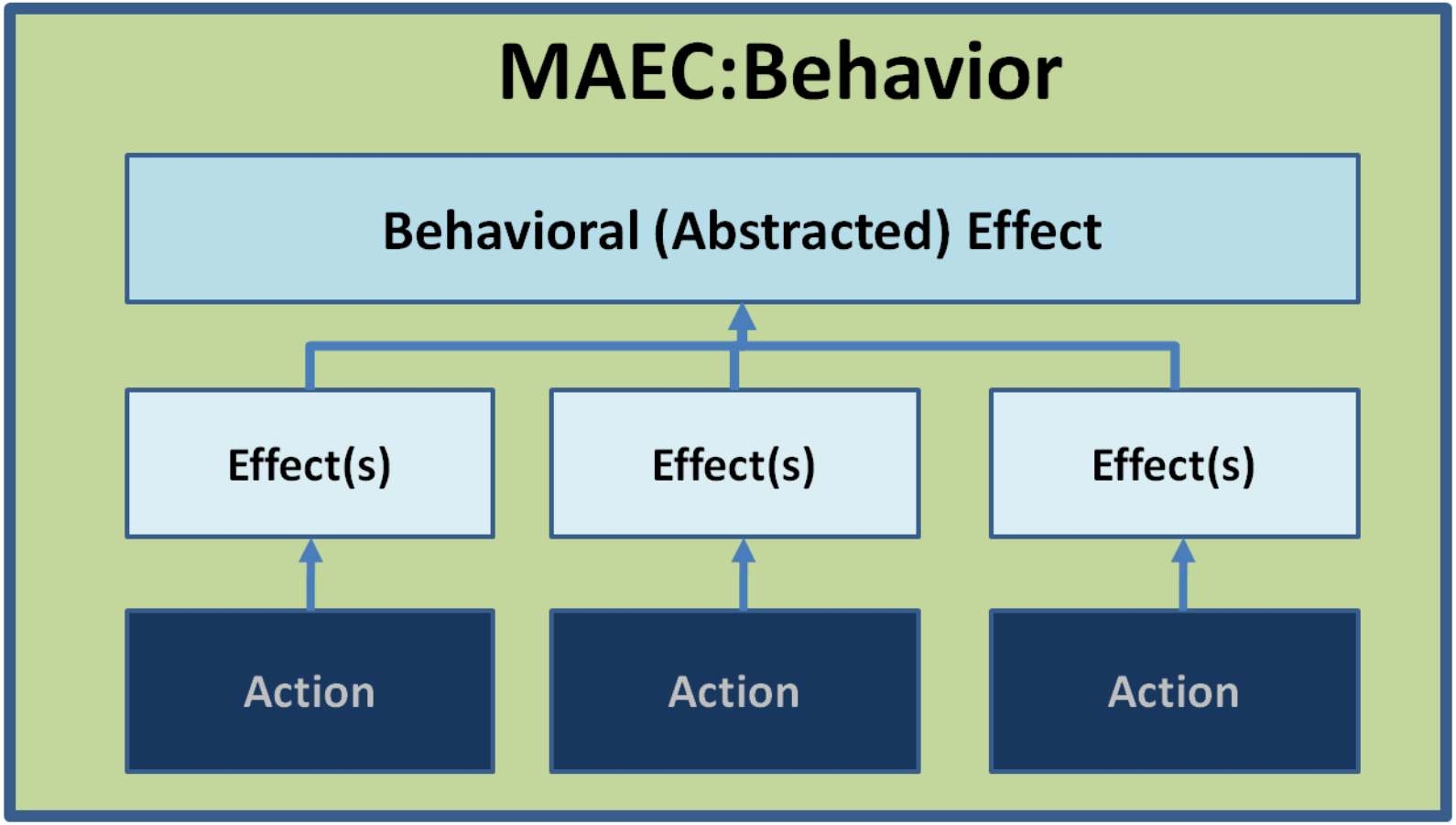
# MAEC Action Model



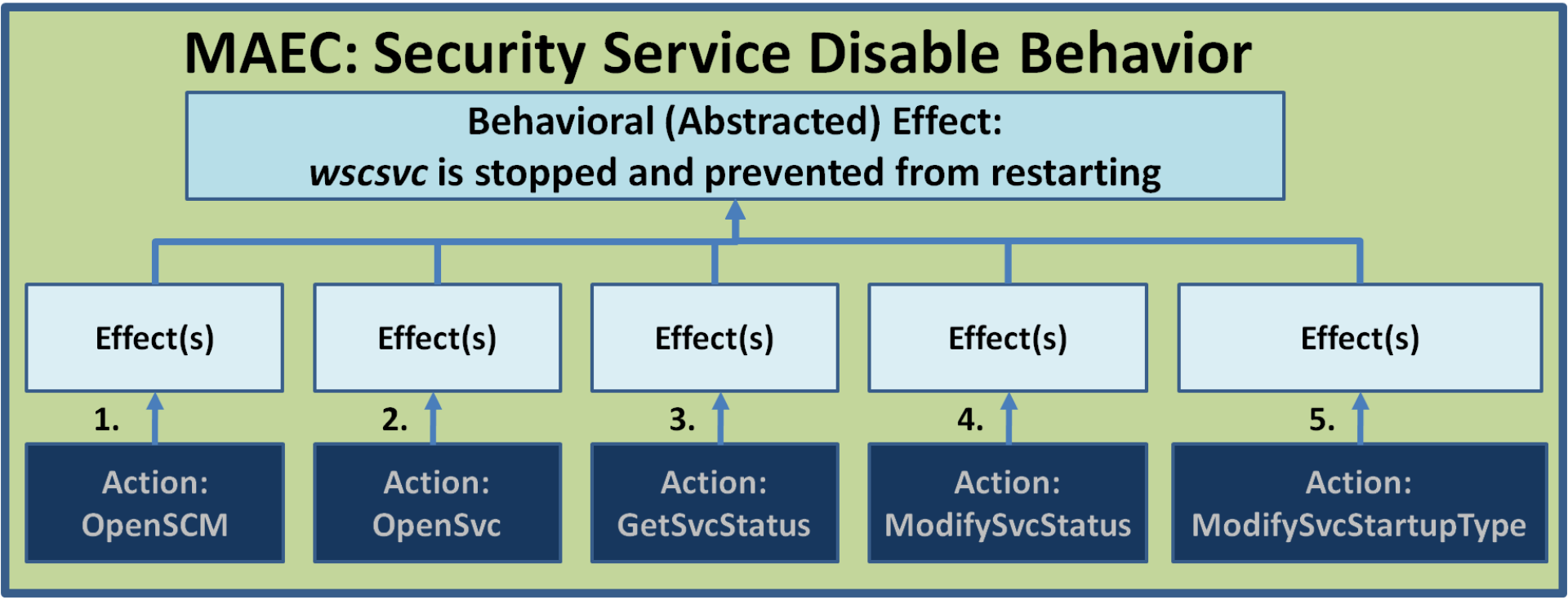
# Action Example



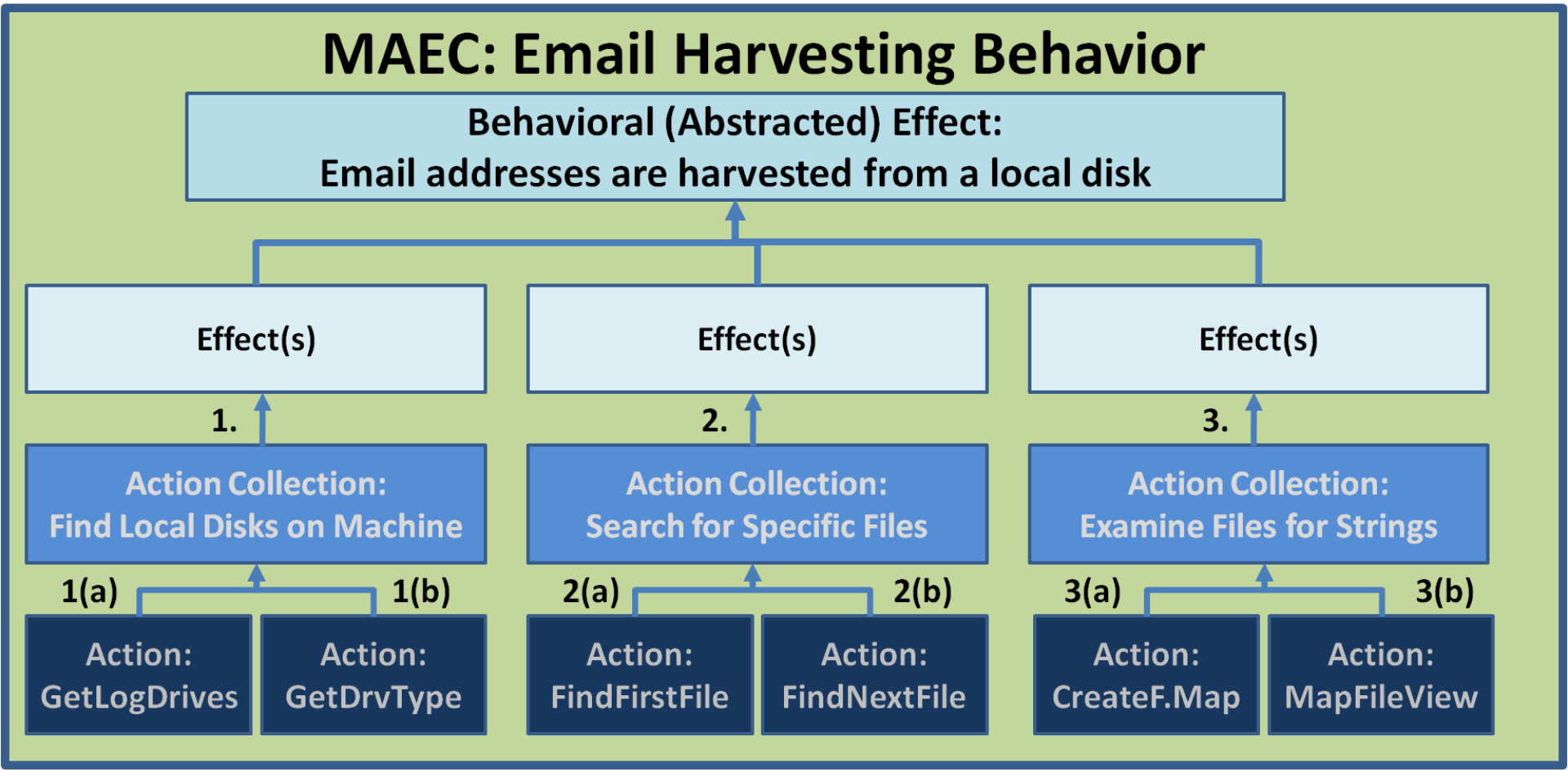
# MAEC Behavior Model



# Basic Behavior Example

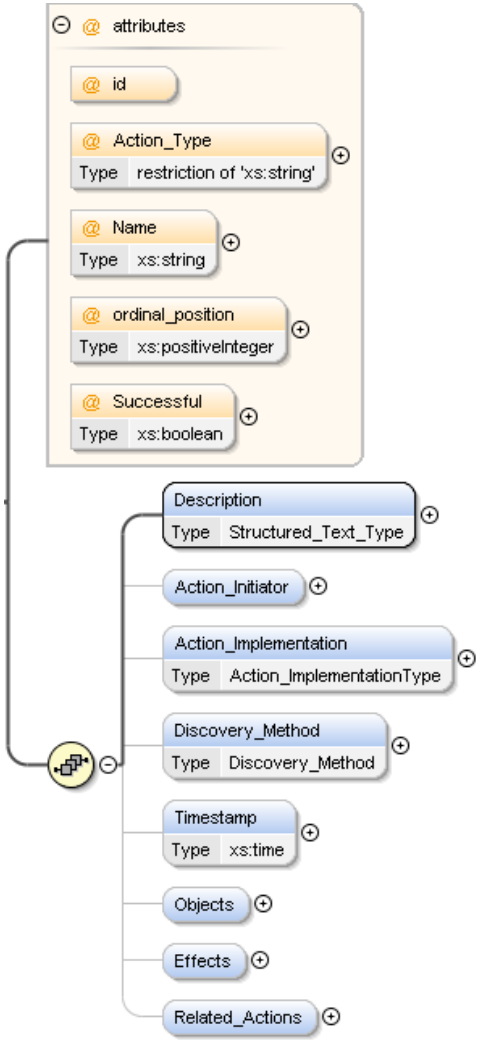


# More Complex Behavior Example

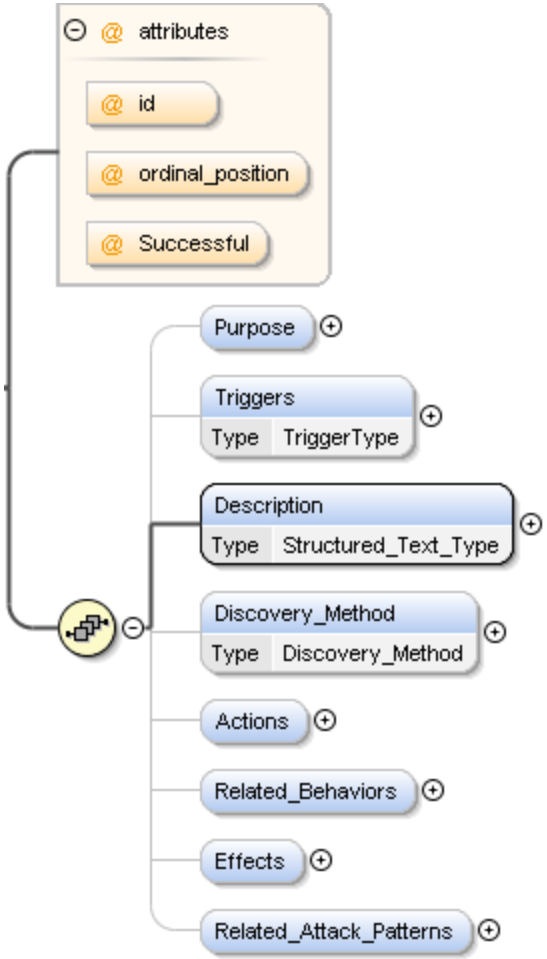


# MAEC Schema Overview – Initial Release

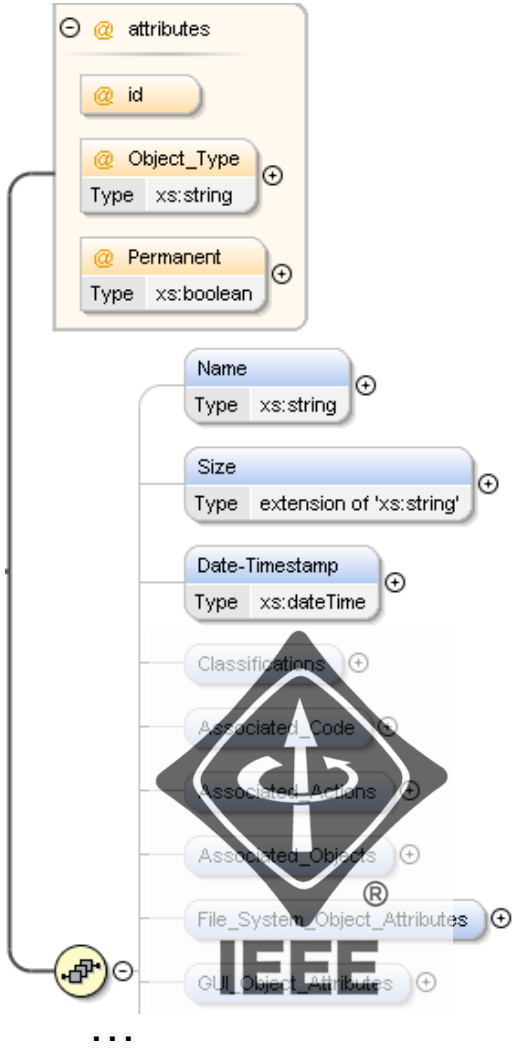
## ActionType




## BehaviorType

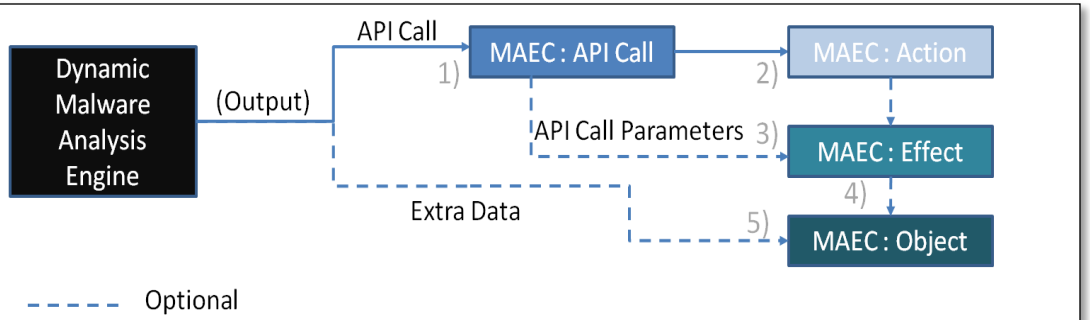



## ObjectType




...

# Dynamic Malware Analysis → MAEC




Process

1) An API call is captured by the analysis engine and mapped to MAEC's enumeration of API calls.
2) The MAEC enumerated call is mapped to its corresponding action.
3) The MAEC defined action is mapped to a corresponding MAEC effect (as necessary), which is populated by the parameters of the call.
4) The MAEC effect is linked to a MAEC object (as necessary).
5) Any extra data output (e.g. file attributes, network capture, etc.) from the analysis engine is mapped to its corresponding object (as necessary).

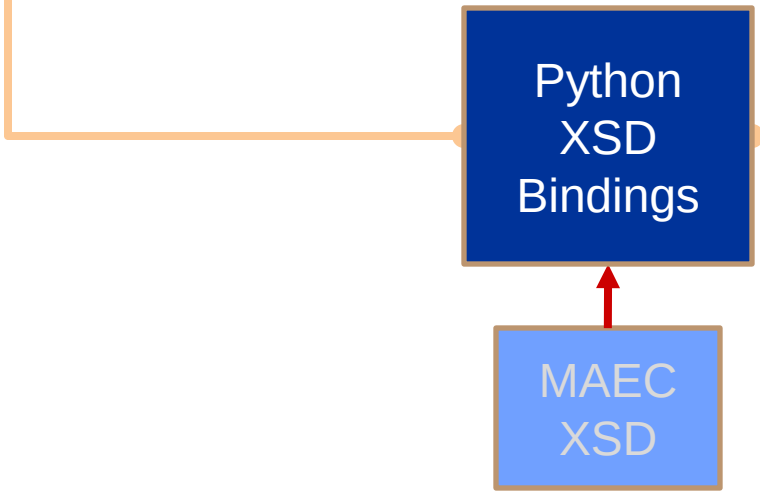
- **Demonstrate the ability to generate MAEC XML descriptions from dynamic analysis tools**
- **Developed proof-of-concept translators for:**
  - **CW Sandbox (Sunbelt)**
  - **ASAT (MITRE)**
  - **Anubis**
  - **ThreatExpert**

# Test Case: CWSandbox Output -> MAEC

```
PID:1080,TID:1812,Caller:$00400000("KB823988.exe"),BEFORE,typFileSystem."FindFirstFile
PID:1080,TID:1812,Caller:$00400000("KB823988.exe"),BEFORE,typFileSystem."SetFileAttrib
PID:1080,TID:1812,Caller:$00400000("KB823988.exe"),BEFORE,typFileSystem."DeleteFileW" -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegOpenKeyExA" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegEnumKeyA" - <e
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegOpenKeyExA" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegOpenKeyExA" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegEnumKeyA" - <e
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegOpenKeyExA" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegEnumValueW" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegOpenKeyExA" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegEnumValueW" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegOpenKeyExA" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegOpenKeyExA" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegOpenKeyExW" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegOpenKeyExW" - -
PID:1080,TID:1812,Caller:$77A80000("CRYPT32.dll"),AFTER,typRegistry."RegCreateKeyExW" -
```

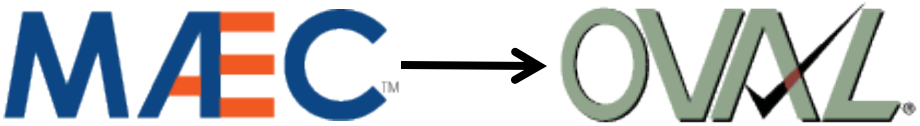
## Raw CWSandbox Output

```
<Action Successful="true" id="10" Action_Type="copy" Name="copy_file">
    <Description/>
    <Action_Initiator type="Process">
        <Initiator_Name>KB823988.exe</Initiator_Name>
        <Process_ID>1080</Process_ID>
        <Thread_ID>1812</Thread_ID>
    </Action_Initiator>
    <Action_Implementation>
        <API_Call>
            <Name>CopyFileW</Name>
            <API_Call_Parameter ordinal_position="1">
                <Name>filetype</Name>
                <Value>file</Value>
            </API_Call_Parameter>
            <API_Call_Parameter ordinal_position="2">
                <Name>srcfile</Name>
                <Value>c:\\KB823988.exe</Value>
            </API_Call_Parameter>
            <API_Call_Parameter ordinal_position="3">
                <Name>dstfile</Name>
                <Value>C:\\WINDOWS\\system32\\ntos.exe</Value>
            </API_Call_Parameter>
            <API_Call_Parameter ordinal_position="4">
                <Name>creationdistribution</Name>
                <Value>CREATE_ALWAYS</Value>
            </API_Call_Parameter>
            <API_Call_Parameter ordinal_position="5">
                <Name>desiredaccess</Name>
                <Value>FILE_ANY_ACCESS</Value>
            </API_Call_Parameter>
            <API_Call_Parameter ordinal_position="6">
                <Name>flags</Name>
                <Value>SECURITY_ANONYMOUS</Value>
            </API_Call_Parameter>
        </API_Call>
    </Action_Implementation>
```




## MAEC XML

- MAEC Actions
- **MAEC Objects**
- MAEC Behaviors

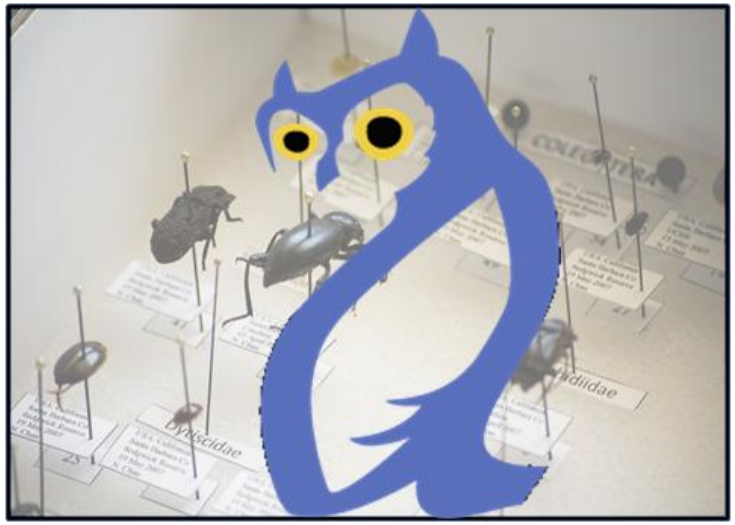



- **MAEC XML to OVAL XML Converter**
  - **Extracts MAEC Objects (defined as being created by malware)**
  - **Converts Objects into OVAL Representations**
  - **Creates definitions and tests to check for the existence of these objects**

- **Capabilities/Use cases**
  - **When used with an OVAL interpreter, it permits the automated testing of the existence of malware artifacts on any host system**
  - **Facilitates the interconnection of malware analysis and malware response**

- **Currently supported artifacts:**
  - **(Windows) Files/Directories/Named Pipes**
  - **Registry Keys**

# Malware Ontologies



- Started to develop a semantic web version of MAEC using NetOwl
  - Many things we'd like to do with MAEC—express complex relationships and constraints—are awkward in XML
- Semantic MAEC will facilitate:
  - Correlation across multiple data sources
  - Using MAEC's multiple levels of abstraction
  - Support automation

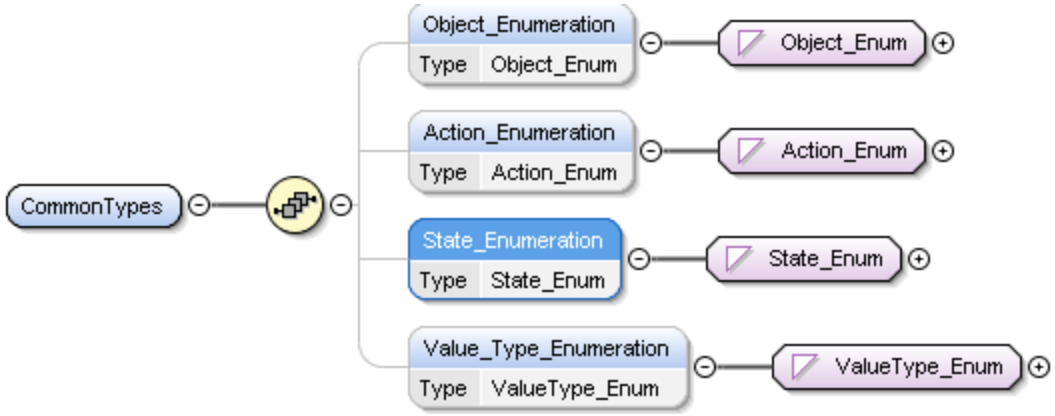
# Collaboration (1/2)


- **IEEE ICSG Malware Working Group**
  - **Developed Malware Metadata exchange schema to facilitate the sharing of sample data between AV product vendors**
    - **Attributes for AV classifications, source (URIs), object properties (file hashes, registry keys), boolean properties (isKernel, isPolymorphic)**
  - **MAEC currently imports the IEEE ICSG Malware Metadata exchange schema**
  - **In the future, the IEEE schema may import certain MAEC elements**

- **Industry /Government**
  - **Although non-standardized, there has been some related work in this realm done by industry and government**
  - **We are actively collaborating with several companies on how to best leverage each other's efforts**
  - **Likewise, we are planning on leveraging the work done by government in the anti-malware space**

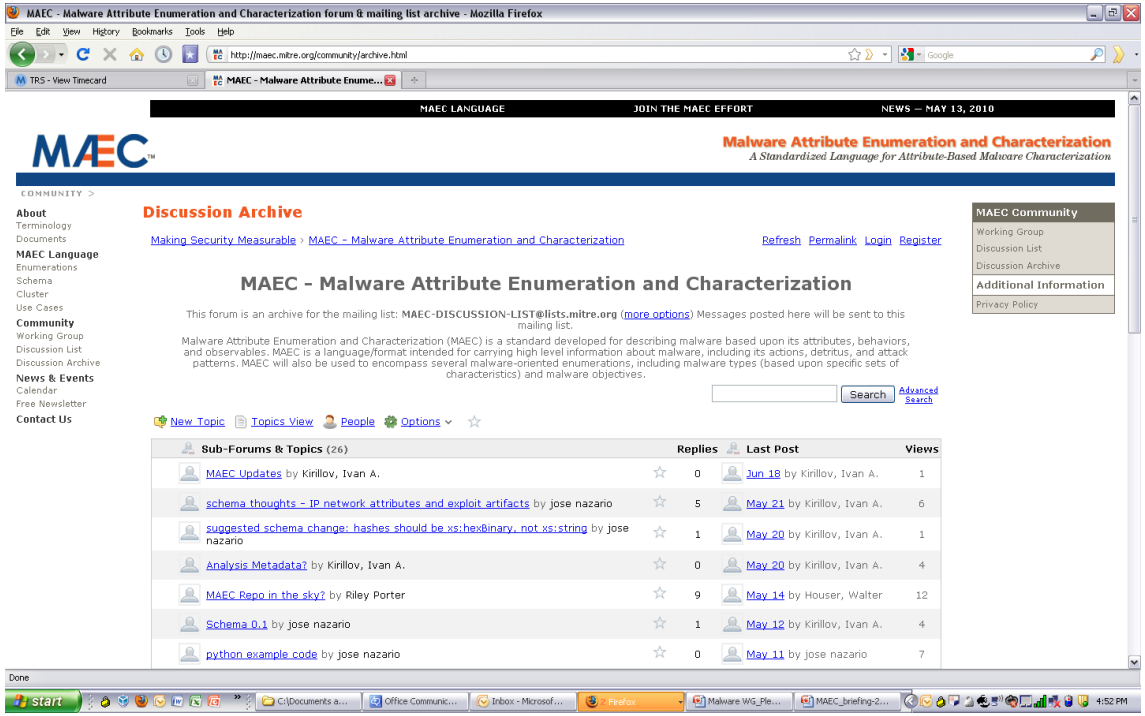
# Collaboration (2/2)


- **Related Making Security Measurable Efforts**
  - **There is significant overlap between MAEC, CAPEC, and CEE in describing observed actions, objects, and states.**
  - **As such, we're working on developing a common schematic structure of observables for use in these efforts:**



# MAEC Community: Discussion List

- **Request to join: http://maec.mitre.org/community/discussionlist.html**
- **Archives available**

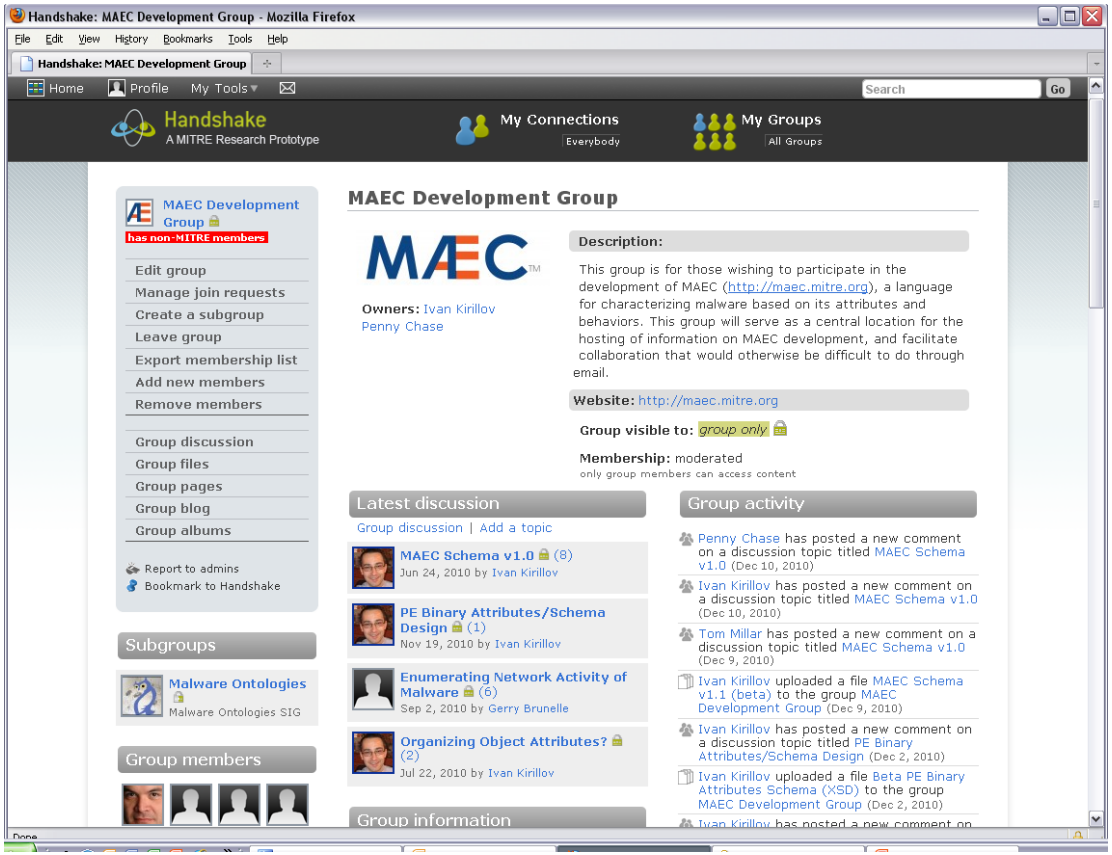





MITRE

UNCLASSIFIED

# MAEC Community: MAEC Development Group on Handshake




- **MITRE hosts a social networking collaboration environment: https://handshake.mitre.org**
- **Supplement to mailing list to facilitate collaborative schema development**
- **Malware Ontologies SIG Subgroup**

# MAEC Schema Dimensions

## Schema Level

### Observables

- Host-based
- Network-based
- Malware metadata

### Mid-Level Behaviors

- Exploitation
- Propagation
- Persistence
- C2
- Exfil, damage, etc.

## Use Cases

### Analysis

- Triage
  - Dynamic
  - Static
- In-Depth

### Operational

- Detection
- Mitigation/Response
- Attribution

## Expressiveness
## Abstraction

# MAEC Schema Roadmap

- **MAEC v 1.0**
  - **Analysis: Dynamic**
  - **Operational: Detection (Host-based through OVAL)**
  - **Schema Level: Host-based observables**
- **MAEC v 1.1**
  - **Analysis: Static**
  - **Schema Level: Malware metadata**
- **Future Schemas**
  - **In-Depth Analysis**
    - **Mid-level behaviors**
  - **Operational**
    - **Signature and Indicators of Compromise (IOCs) management**
    - **Mitigation and response support**
  - **Expressiveness**
    - **Operators, constraints, relationships**

# Next Steps

- Complete OWL ontology based on MAEC schema
- XSLT transformation of MAEC XML → HTML
- Implement common observables schema (v 1.2?)
  - Based on MAEC/CAPEC/CEE collaboration
- Prioritize schema roadmap
- Encourage and invite more participation in the development process
  - MAEC Website: http://maec.mitre.org (contains MAEC Discussion list sign-up)
  - MAEC Handshake Group (send email to maec@mitre.org to request an invitation)

# Questions?

Sean Barnum
MITRE
sbarnum@mitre.org



MITRE