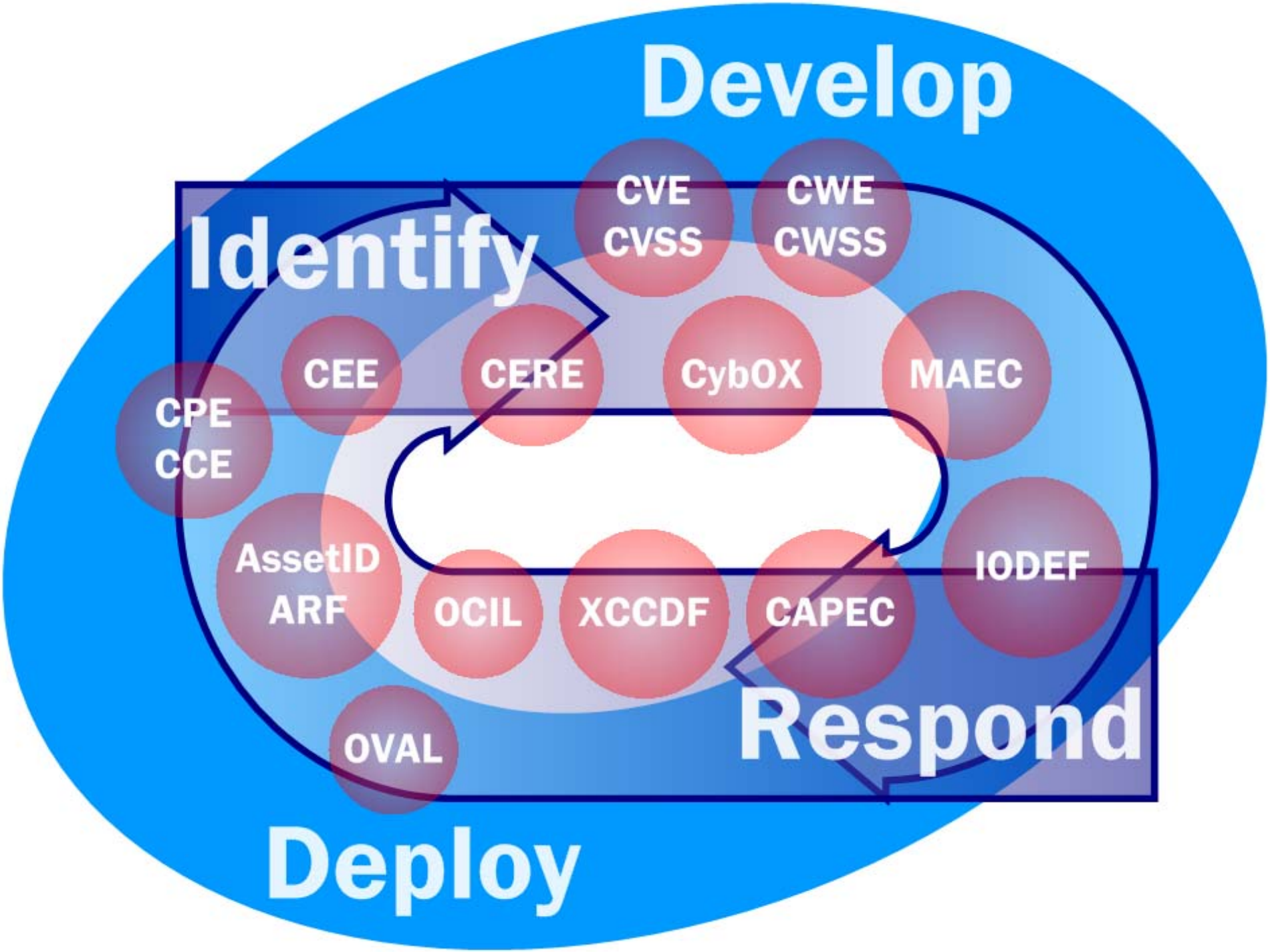
HS SEDI
Homeland Security Systems Engineering and Development Institute

Enabling Distributed Threat Analysis: Common Attack Patterns and Malware Characterization

Sean Barnum
Penny Chase

Aug 2011


Develop
Identify
Respond
Deploy
CVE
CVSS
CWE
CWSS
CEE
CERE
CybOX
MAEC
CPE
CCE
AssetID
ARF
OCIL
XCCDF
CAPEC
IODEF
OVAL

Premise

- Building secure systems and effectively responding to incidents requires an understanding of the relevant threats
- An actionable understanding of today's scale, complexity and volume of relevant threats requires structured representations to support automation

- Threat information
  - Attacker motivation
  - Attacker behavior (TTP)
  - Attacker capability
  - Targeted attack surface
  - Potential impact
  - How to detect threat
  - How to mitigate threat

- Solution Resources
  - Common Attack Pattern Enumeration and Classification (CAPEC) for structured characterization of attack patterns
  - Malware Attribute Enumeration and Characterization (MAEC) language for structured characterization of malware
  - Cyber Observable eXpression (CybOX) underpinning CAPEC & MAEC

The Long-established Principal of "Know Your Enemy"

- "One who knows the enemy and knows himself will not be endangered in a hundred engagements. One who does not know the enemy but knows himself will sometimes be victorious. Sometimes meet with defeat. One who knows neither the enemy nor himself will invariably be defeated in every engagement."
  - Chapter 3: "Planning the Attack"
    - The Art of War, Sun Tzu

An appropriate defense can only be established if you know how it will be attacked

What are Attack Patterns?

- Blueprint for creating a specific type of attack
- Abstracted common attack approaches from the set of known exploits

- Capture the attacker's perspective to aid software developers, acquirers and operators in improving the assurance profile of their systems

Leveraging Attack Patterns Throughout the Software Lifecycle

- **Guide definition of appropriate policies**
- **Guide creation of appropriate security requirements (positive and negative)**
- **Provide context for architectural risk analysis**
- **Guide risk-driven secure code review**
- **Provide context for appropriate security testing**
- **Identify and characterize threat TTPs for red teaming**
- **Provide a bridge between secure development and secure operations**
  - Identify patterns of malicious behavior for attack detection and characterization in operations

Common Attack Pattern Enumeration and Classification (CAPEC)

HS SEDI
Homeland Security Systems Engineering and Development Institute

- **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

- **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 observation and correlation

- **Where is CAPEC today?**
  - http://capec.mitre.org
  - Currently 386 patterns, stubs, named attacks
    68 Categories & 6 Views

CAPEC™

CAPEC – Common Attack Pattern Enumeration and Classification (CAPEC)

http://capec.mitre.org/

CAPEC Common Attack Pattern Enumeration and Classification

A Community Knowledge Resource for Building Secure Software

Search by ID: Go

- **CAPEC List**
  - Full CAPEC Dictionary
  - Methods of Attack View
  - Reports
- **About CAPEC**
  - Documents
  - Resources
- **Community**
  - Related Activities
  - Collaboration List
- **News & Events**
  - Calendar
  - Free Newsletter
- **Contact Us**
  - Search the Site

Building software with an adequate level of security assurance for its mission becomes more and more challenging every day as the size, complexity, and tempo of software creation increases and the number and the skill level of attackers continues to grow. These factors each exacerbate the issue that, to build secure software, builders must ensure that they have protected every relevant potential vulnerability; yet, to attack software, attackers often have to find and exploit only a single exposed vulnerability. To identify and mitigate relevant vulnerabilities in software, the development community needs more than just good software engineering and analytical practices, a solid grasp of software security features, and a powerful set of tools. All of these things are necessary but not sufficient. To be effective, the community needs to think outside of the box and to have a firm grasp of the attacker's perspective and the approaches used to exploit software.

Attack patterns are a powerful mechanism to capture and communicate the attacker's perspective. They are descriptions of common methods for exploiting software. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples.

To assist in enhancing security throughout the software development lifecycle, and to support the needs of developers, testers and educators, the **Common Attack Pattern Enumeration and Classification (CAPEC)** is sponsored by the Department of Homeland Security as part of the Software Assurance strategic initiative of the National Cyber Security Division. The objective of this effort is to provide a publicly available catalog of attack patterns along with a comprehensive schema and classification taxonomy. This site now contains the initial set of content and will continue to evolve with public participation and contributions to form a standard mechanism for identifying, collecting, refining, and sharing attack patterns among the software community.

Release 1.6 Available

Page Last Updated: February 07, 2011

MITRE

CAPEC is a Software Assurance strategic initiative co-sponsored by the National Cyber Security Division of the U.S. Department of Homeland Security.

This Web site is sponsored and managed by The MITRE Corporation to enable stakeholder collaboration.

Contact capec@mitre.org for more information.

Privacy policy
Terms of use
Contact us

Member of Making Security Measurable™

Done

What do Attack Patterns Look Like?

HS SEDI
Homeland Security Systems Engineering and Development Institute

- **Primary Schema Elements**
  - Identifying Information
    - Attack Pattern ID
    - Attack Pattern Name
  - Describing Information
    - Description
    - Related Weaknesses
    - Related Vulnerabilities
    - Method of Attack
    - Examples-Instances
    - References
  - Prescribing Information
    - Solutions and Mitigations
  - Scoping and Delimiting Information
    - Typical Severity
    - Typical Likelihood of Exploit
    - Attack Prerequisites
    - Attacker Skill or Knowledge Required
    - Resources Required
    - Attack Motivation-Consequences
    - Context Description

- **Supporting Schema Elements**
  - **Describing Information**
    - Injection Vector
    - Payload
    - Activation Zone
    - Payload Activation Impact
  - **Diagnosing Information**
    - Probing Techniques
    - Indicators-Warnings of Attack
    - Obfuscation Techniques
  - **Enhancing Information**
    - Related Attack Patterns
    - Relevant Security Requirements
    - Relevant Design Patterns
    - Relevant Security Patterns

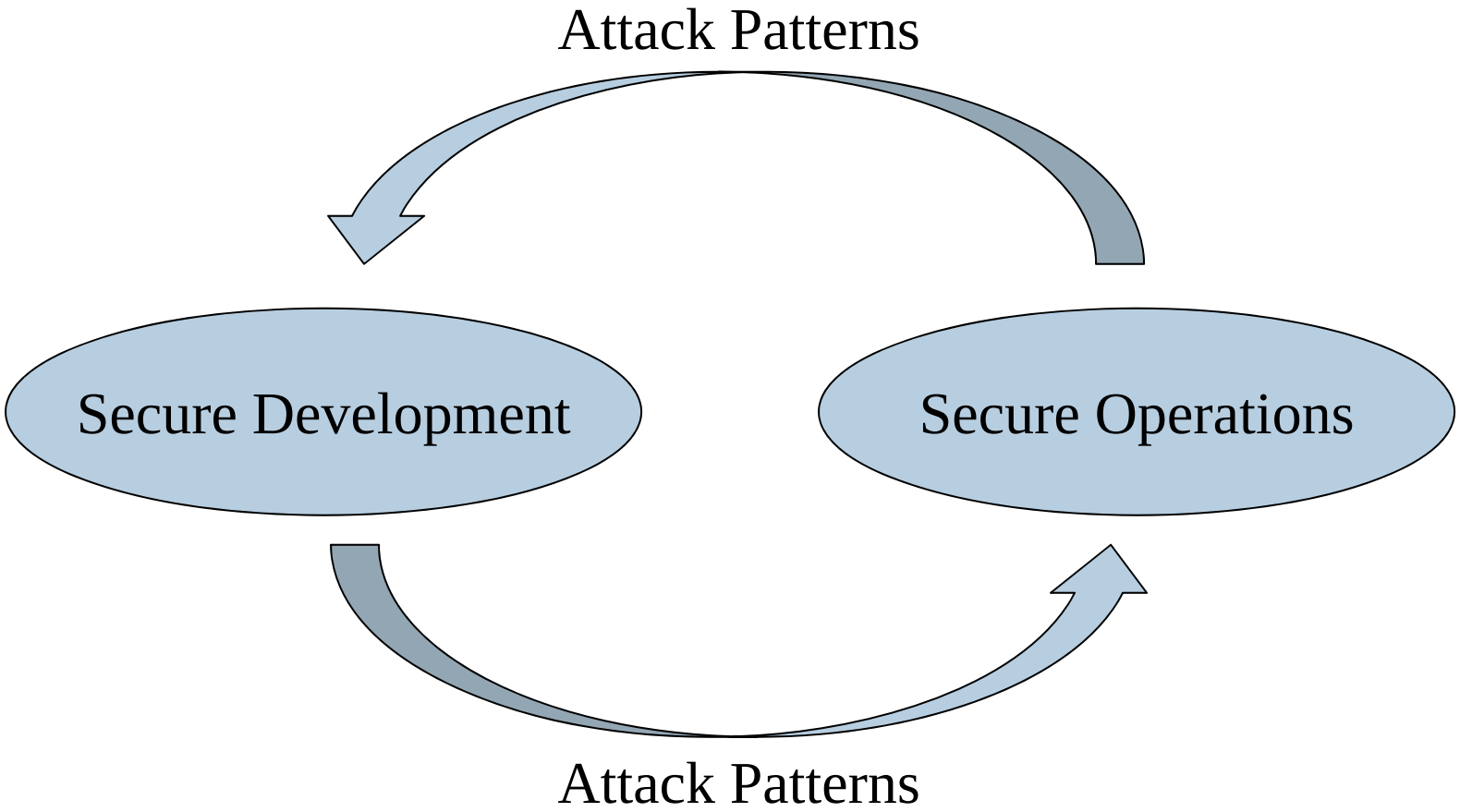
Attack Pattern Description Schema Formalization

HS SEDI
Homeland Security Systems Engineering and Development Institute

Description

- **Summary**
- **Attack_Execution_Flow**
  - Attack_Phase$^{1..3}$ (Name(Explore, Experiment, Exploit))
    - **Attack_Step$^{1..*}$**
      - **Attack_Step_Title**
      - **Attack_Step_Description**
      - **Attack_Step_Technique $^{0..*}$**
        - **Attack_Step_Technique_Description**
        - **Leveraged_Attack_Patterns**
        - **Relevant_Attack_Surface_Elements**
        - **Observables$^{0..*}$**
        - **Environments**
      - **Indicator$^{0..*}$ (ID, Type(Positive, Failure, Inconclusive))**
        - **Indicator_Description**
        - **Relevant_Attack_Surface_Elements**
        - **Environments**
      - **Outcome$^{0..*}$ (ID, Type(Success, Failure, Inconclusive))**
        - **Outcome_Description**
        - **Relevant_Attack_Surface_Elements**
        - **Observables$^{0..*}$**
        - **Environments**
      - **Security Control$^{0..*}$ (ID, Type(Detective, Corrective, Preventative))**
        - **Security_Control_Description**
        - **Relevant_Attack_Surface_Elements**
        - **Observables$^{0..*}$**
        - **Environments**
      - **Observables$^{0..*}$**

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.

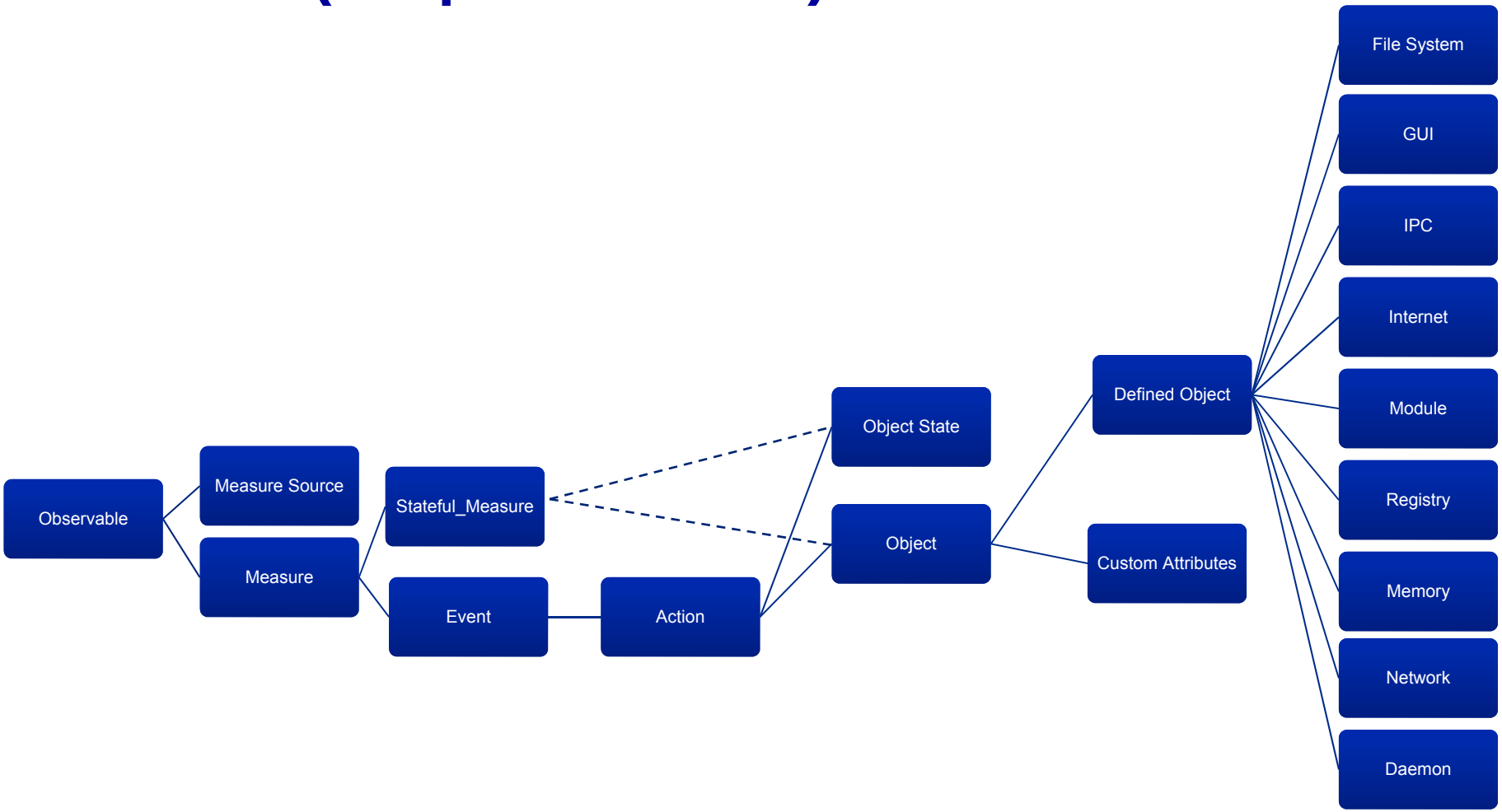
Secure Development Knowledge Offers Unique Value to Secure Operations

- 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.

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.

Common Cyber Observables (CybOX) Schema (simple overview)



UNCLASSIFIED

- These same cyber observables also apply to numerous other domains
  - Malware characterization
  - Operational Events
  - Logging
  - Cyber situational awareness
  - Incident response
  - Forensics
  - Etc.

Individual CAPEC Dictionary Definition (Release 1.2)

Blind SQL Injection

Attack Pattern ID	7	**Pattern Abstraction: Detailed**
Typical Severity	High	

Description

Summary

Blind SQL Injection results from an insufficient mitigation for SQL Injection. Although suppressing database error messages are considered best practice, the suppression alone is not sufficient to prevent SQL Injection. Blind SQL Injection is a form of SQL Injection that overcomes the lack of error messages. Without the error messages that facilitate SQL Injection, the attacker constructs input strings that probe the target through simple Boolean SQL expressions. The attacker can determine if the syntax and structure of the injection was successful based on whether the query was executed or not. Applied iteratively, the attacker determines how and where the target is vulnerable to SQL Injection.

In order to achieve this using Blind SQL Injection, an attacker:

For example, an attacker may try entering something like "username' AND 1=1; --" in an input field. If the result is the same as when the attacker entered "username" in the field, then the attacker knows that the application is vulnerable to SQL Injection. The attacker can then ask yes/no questions from the database server to extract information from it. For example, the attacker can extract table names from a database using the following types of queries:

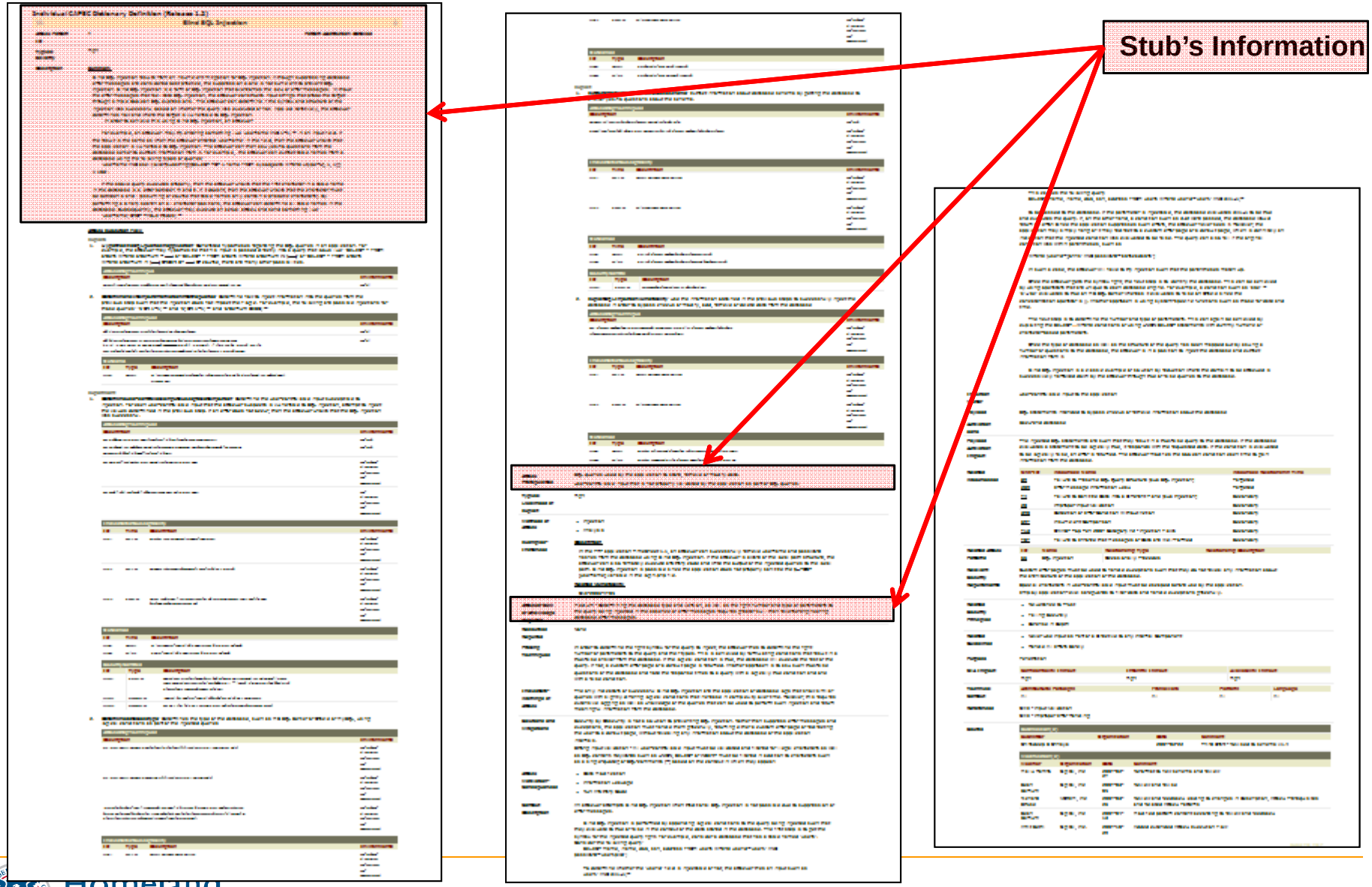
"username' AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype='U'), 1, 1))) > 108".

If the above query executes properly, then the attacker knows that the first character in a table name in the database is a letter between m and z. If it doesn't, then the attacker knows that the character must be between a and l (assuming of course that table names only contain alphabetic characters). By performing a binary search on all character positions, the attacker can determine all table names in the database. Subsequently, the attacker may execute an actual attack and send something like:

"username'; DROP TABLE trades; --

Complete CAPEC Entry Information

HS SEDI
Homeland Security Systems Engineering and Development Institute



CAPEC Status

HS SEDI
Homeland Security Systems Engineering and Development Institute

Where is CAPEC today?

- Recent versions
  - Significant schema changes
    - Including addition of Observables structure
  - New content
  - Enhanced & refined content
  - Added new categories of patterns
    - Network attack patterns
    - Physical security attacks
    - Social engineering attacks
    - Supply chain attacks

Currently 386 patterns, stubs, named attacks; 68 categories and 6 views

CAPEC Current Content
(15 Major Categories)

1000 - Mechanism of Attack

- **Data Leakage Attacks - (118)**
- **Resource Depletion - (119)**
- **Injection (Injecting Control Plane content through the Data Plane) - (152)**
- **Spoofing - (156)**
- **Time and State Attacks - (172)**
- **Abuse of Functionality - (210)**
- **Exploitation of Authentication - (225)**
- **Probabilistic Techniques - (223)**
- **Exploitation of Privilege/Trust - (232)**
- **Data Structure Attacks - (255)**
- **Resource Manipulation - (262)**
- **Physical Security Attacks (436)**
- **Network Reconnaissance - *(286)***
- **Social Engineering Attacks (403)**
- **Supply Chain Attacks (437)**

CAPEC Current Content (Which Expand to…)

HS SEDI
Homeland Security Systems Engineering and Development Institute

- 1000 - Mechanism of Attack
  - Data Leakage Attacks - (118)
    - Data Excavation Attacks - (116)
    - Data Interception Attacks - (117)
  - Resource Depletion - (119)
    - Violating Implicit Assumptions Regarding XML Content (aka XML Denial of Service (XDoS)) - (82)
    - Resource Depletion through Flooding - (125)
    - Resource Depletion through Allocation - (130)
    - Resource Depletion through Leak - (131)
    - Denial of Service through Resource Depletion - (227)
  - Injection (Injecting Control Plane content through the Data Plane) - (152)
    - Remote Code Inclusion - (253)
    - Analog In-band Switching Signals (aka Blue Boxing) - (5)
    - SQL Injection - (66)
    - Email Injection - (134)
    - Format String Injection - (135)
    - LDAP Injection - (136)
    - Parameter Injection - (137)
    - Reflection Injection - (138)
    - Code Inclusion - (175)
    - Resource Injection - (240)
    - Script Injection - (242)
    - Command Injection - (248)
    - Character Injection - (249)
    - XML Injection - (250)
    - DTD Injection in a SOAP Message - (254)
  - Spoofing - (156)
    - Content Spoofing - (148)
    - Identity Spoofing (Impersonation) - (151)
    - Action Spoofing - (173)
  - Time and State Attacks - (172)
    - Forced Deadlock - (25)
    - Leveraging Race Conditions - (26)
    - Leveraging Time-of-Check and Time-of-Use (TOCTOU) Race Conditions - (29)
    - Manipulating User State - (74)
  - Abuse of Functionality - (210)
    - Functionality Misuse - (212)
    - Abuse of Communication Channels - (216)
    - Forceful Browsing - (87)
    - Passing Local Filenames to Functions That Expect a URL - (48)
    - Probing an Application Through Targeting its Error Reporting - (54)
    - WSDL Scanning - (95)
    - API Abuse/Misuse - (113)
    - Try All Common Application Switches and Options - (133)
    - Cache Poisoning - (141)
    - Software Integrity Attacks - (184)
    - Directory Traversal - (213)
    - Analytic Attacks - (281)
  - Probabilistic Techniques - (223)
    - Fuzzing - (28)
    - Manipulating Opaque Client-based Data Tokens - (39)
    - Brute Force - (112)
    - Screen Temporary Files for Sensitive Information - (155)
  - Exploitation of Authentication - (225)
    - Exploitation of Session Variables, Resource IDs and other Trusted Credentials - (21)
    - Authentication Abuse - (114)
    - Authentication Bypass - (115)
  - Exploitation of Privilege/Trust - (232)
    - Privilege Escalation - (233)
    - Exploiting Trust in Client (aka Make the Client Invisible) - (22)
    - Hijacking a Privileged Thread of Execution - (30)
    - Subvert Code-signing Facilities - (68)
    - Target Programs with Elevated Privileges - (69)
    - Exploitation of Authorization - (122)
    - Hijacking a privileged process - (234)
  - Data Structure Attacks - (255)
    - Accessing/Intercepting/Modifying HTTP Cookies - (31)
    - Buffer Attacks - (123)
    - Attack through Shared Data - (124)
    - Integer Attacks - (128)
    - Pointer Attack - (129)
  - Resource Manipulation - (262)
    - Accessing/Intercepting/Modifying HTTP Cookies - (31)
    - Input Data Manipulation - (153)
    - Resource Location Attacks - (154)
    - Infrastructure Manipulation - (161)
    - File Manipulation - (165)
    - Variable Manipulation - (171)
    - Configuration/Environment manipulation - (176)
    - Abuse of transaction data strutcture - (257)
    - Registry Manipulation - (269)
    - Schema Poisoning - (271)
    - Protocol Manipulation - (272)
  - Network Reconnaissance - (286)
    - ICMP Echo Request Ping - (285)
    - TCP SYN Scan - (287)
    - ICMP Echo Request Ping - (288)
    - Infrastructure-based footprinting - (289)
    - Enumerate Mail Exchange (MX) Records - (290)
    - DNS Zone Transfers - (291)
    - Host Discovery - (292)
    - Traceroute Route Enumeration - (293)
    - ICMP Address Mask Request - (294)
    - ICMP Timestamp Request - (295)
    - ICMP Information Request - (296)
    - TCP ACK Ping - (297)
    - UDP Ping - (298)
    - TCP SYN Ping - (299)
    - Port Scanning - (300)
    - TCP Connect Scan - (301)
    - TCP FIN scan - (302)
    - TCP Xmas Scan - (303)
    - TCP Null Scan - (304)
    - TCP ACK Scan - (305)
    - TCP Window Scan - (306)
    - TCP RPC Scan - (307)
    - UDP Scan - (308)

CAPEC Current Content (386 Attacks…)

HS SEDI
Homeland Security Systems Engineering and Development Institute

Current Maturation Paths

- Extend coverage of CAPEC
- Improve quality of CAPEC
- Expand the scope of CAPEC
- Bridge secure development with secure operations
- Improve integration with other standards (MAEC, CEE, etc.)
- Expand use of CAPEC
- Establish initial compatibility program

HS SEDI
Homeland Security Systems Engineering and Development Institute

Malware Attribute Enumeration & Characterization

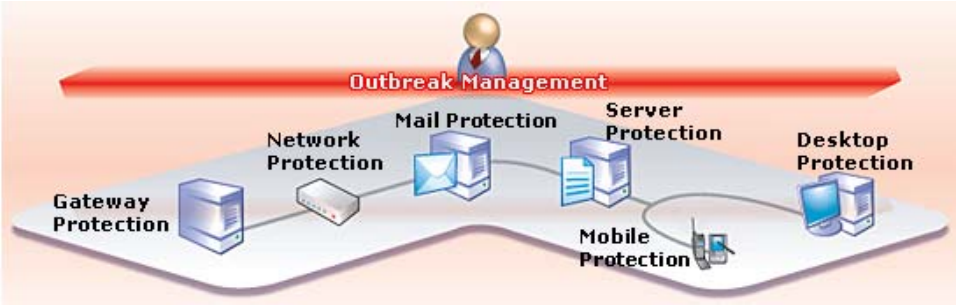
MAEC™

Homeland Security

The HS SEDI FFRDC is managed and operated by The MITRE Corporation for DHS.

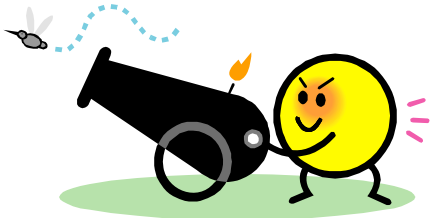
Why Do We Need to Develop Standards for Malware?

Multiple layers of protection



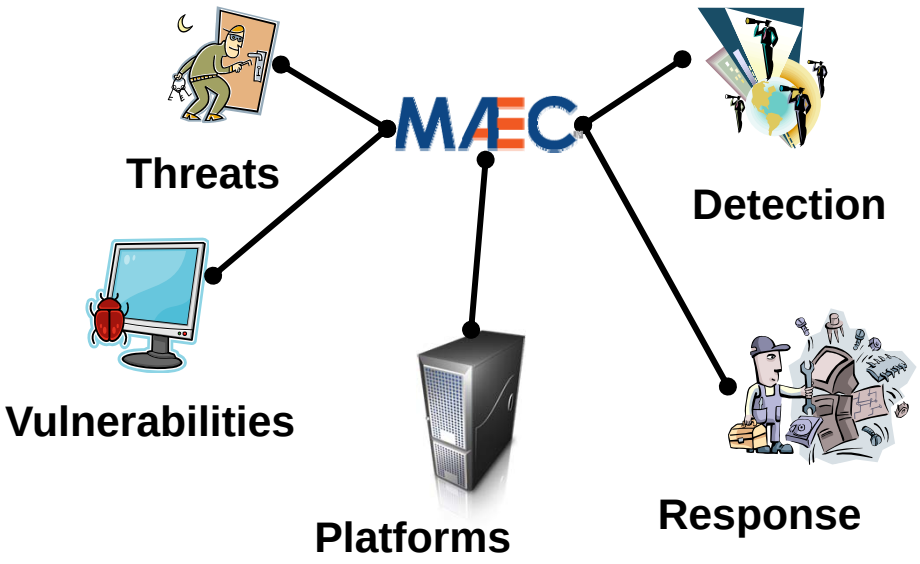
Lots of products

Inconsistent reports



There's an arms race

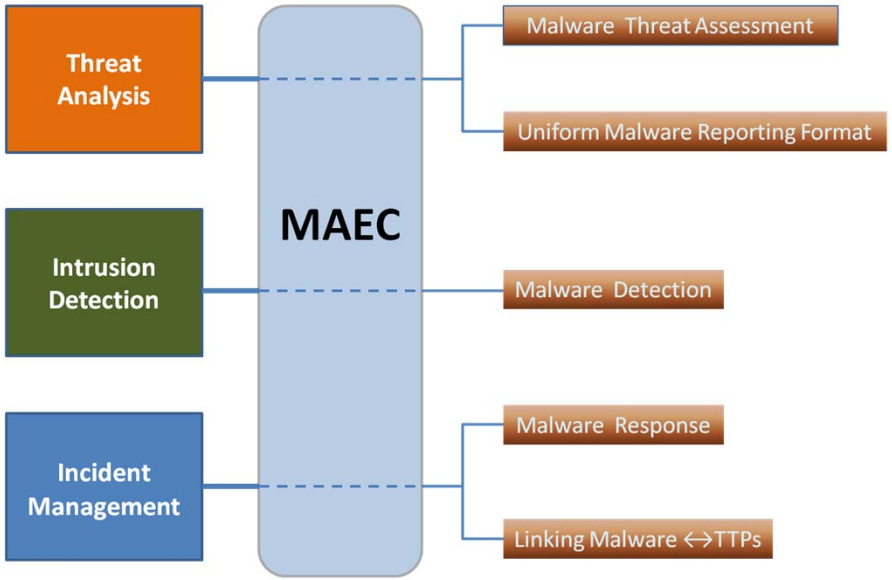
Malware Attribute Enumeration and Characterization (MAEC)



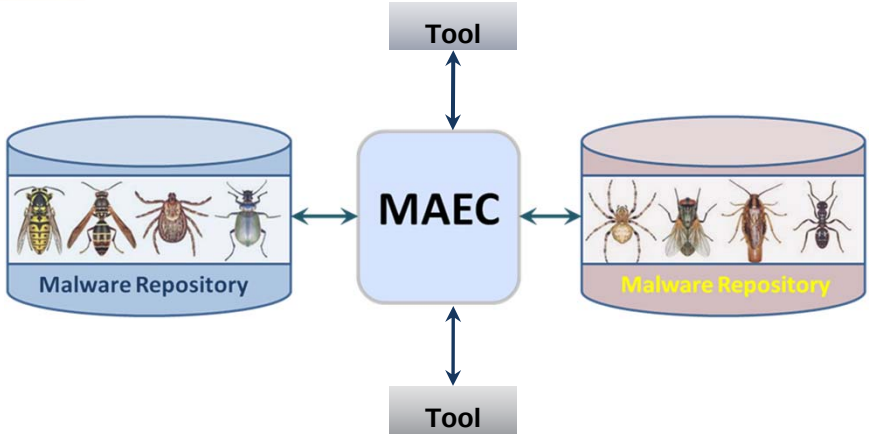
- Language for sharing structured information about malware
  - Grammar (Schema)
  - Vocabulary (Enumerations)
  - Collection Format (Bundle)
- Focus on attributes and behaviors
- Enable correlation, integration, and automation

MAEC Use Cases

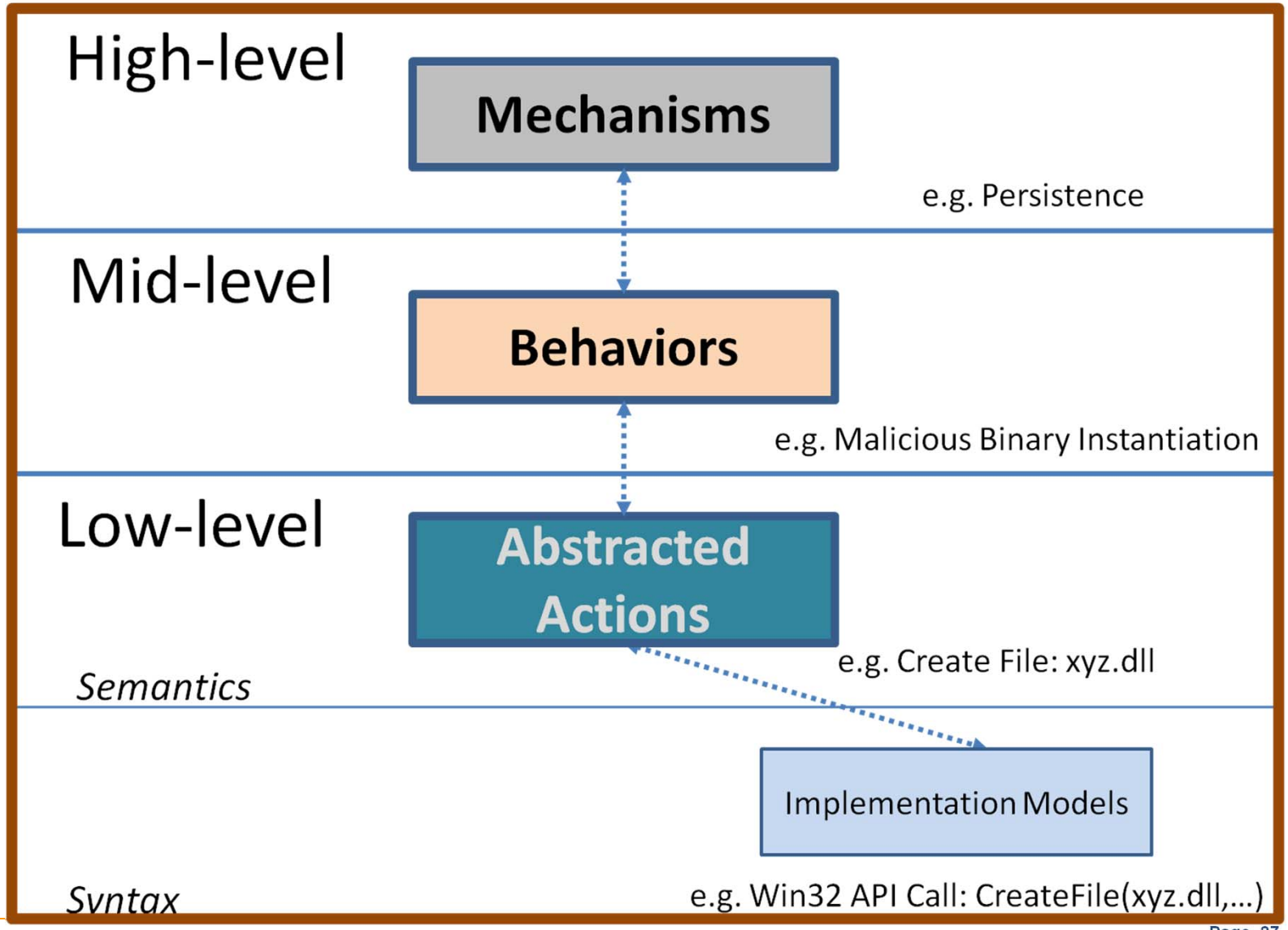
- **Operational**



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



MAEC Overview



MAEC Description: Profiling Zeus C2

MAEC Mechanism: C2

- MAEC Behavior: C2 Get Configuration
- MAEC Behavior: C2 Beacon
- MAEC Behavior: C2 Receive Command
- MAEC Behavior: C2 Send Data

MITRE

MAEC & Zeus C2 I

Mechanism: C2

- **Behavior: Get Configuration**
- **Behavior: Beacon**
- **Behavior: Recv Command**
- **Behavior: Send Data**

MAEC Behavior: C2 Get Configuration

Protocol: HTTP
Encryption Type: RC4/custom

MAEC Action: http_get

MAEC Object: tcp_connection

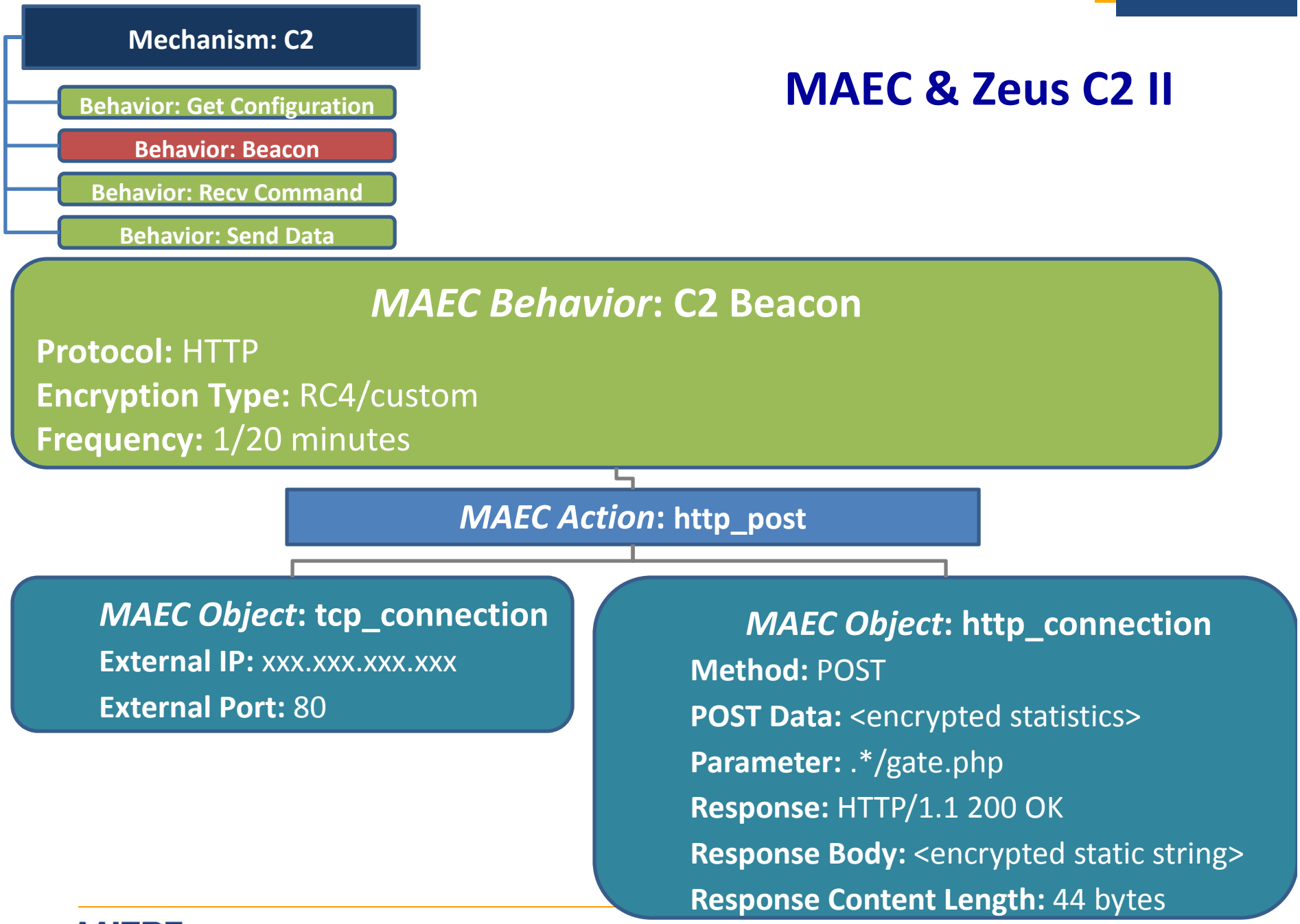
External IP: xxx.xxx.xxx.xxx
External Port: 80

MAEC Object: http_connection

Method: GET
Parameter: /config.bin
Response: HTTP/1.1 200 OK
Response Body: <encrypted config.bin file>
Response Content Length: 1212 bytes

MITRE

MAEC & Zeus C2 II



MITRE

MAEC & Zeus C2 III

Mechanism: C2

- Behavior: Get Configuration
- Behavior: Beacon
- Behavior: Recv Command
- Behavior: Send Data

***MAEC Behavior*: C2 Receive Command**

Protocol: HTTP

Encryption Type: RC4/custom

Supported Commands: reboot, kos, shutdown, bc_add, bc_del, block_url, unblock_url, block_fake, getfile, getcerts, resetgrab, upcfg, rename_bot …

***MAEC Action*: decode_http_response**

***MAEC Object*: tcp_connection**

External IP: xxx.xxx.xxx.xxx

External Port: 80

***MAEC Object*: http_connection**

Response Body: <encrypted command string>

Response Content Length: > 44 bytes

MITRE

MAEC & Zeus C2 IV

Mechanism: C2

- Behavior: Get Configuration
- Behavior: Beacon
- Behavior: Recv Command
- Behavior: Send Data

***MAEC Behavior*: C2 Send Data**

Protocol: HTTP

Encryption Type: RC4/custom

***MAEC Action*: http_post**

***MAEC Object*: tcp_connection**

External IP: xxx.xxx.xxx.xxx

External Port: 80

***MAEC Object*: http_connection**

Method: POST

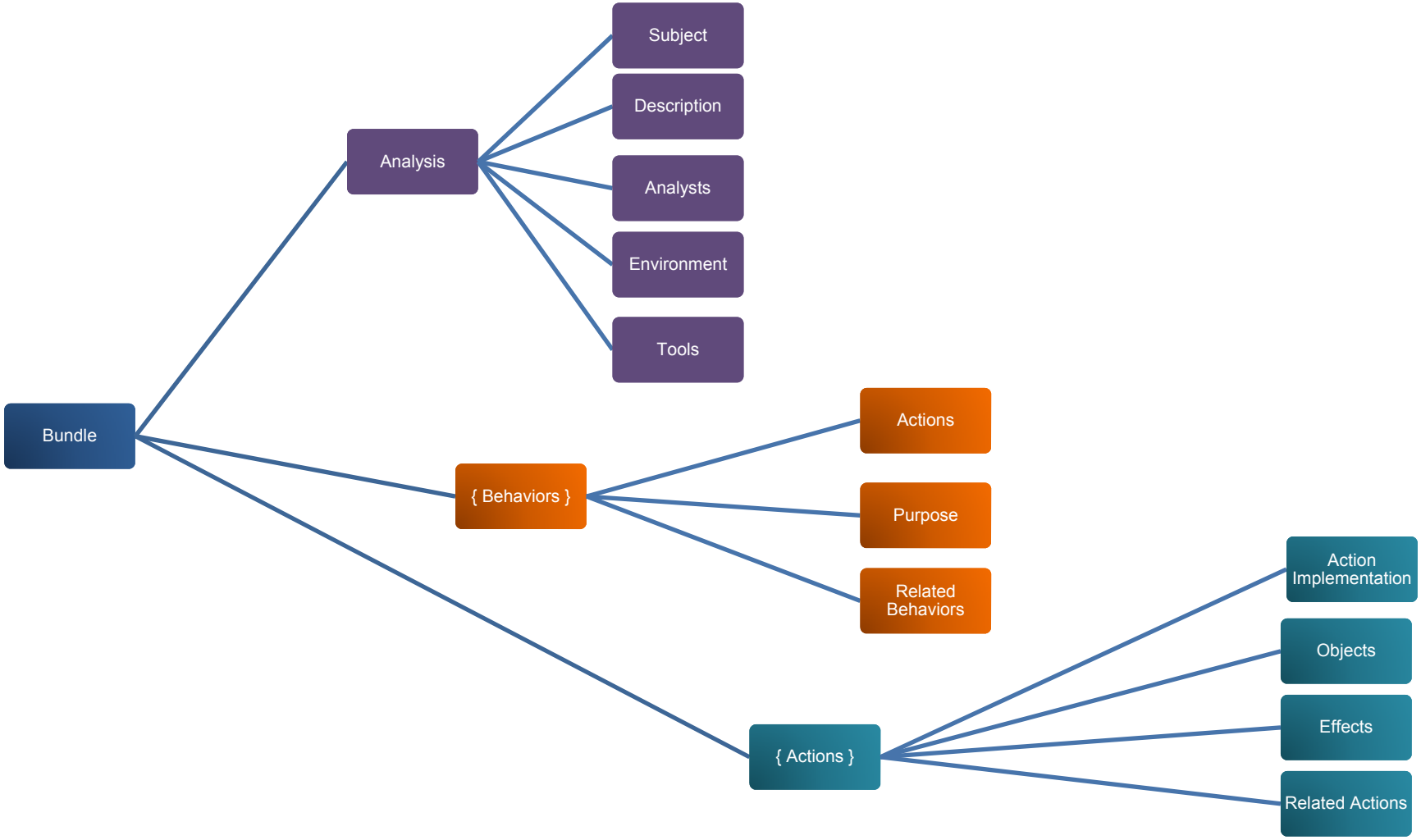
POST Data: <encrypted stolen data>

Parameter: .*/gate.php

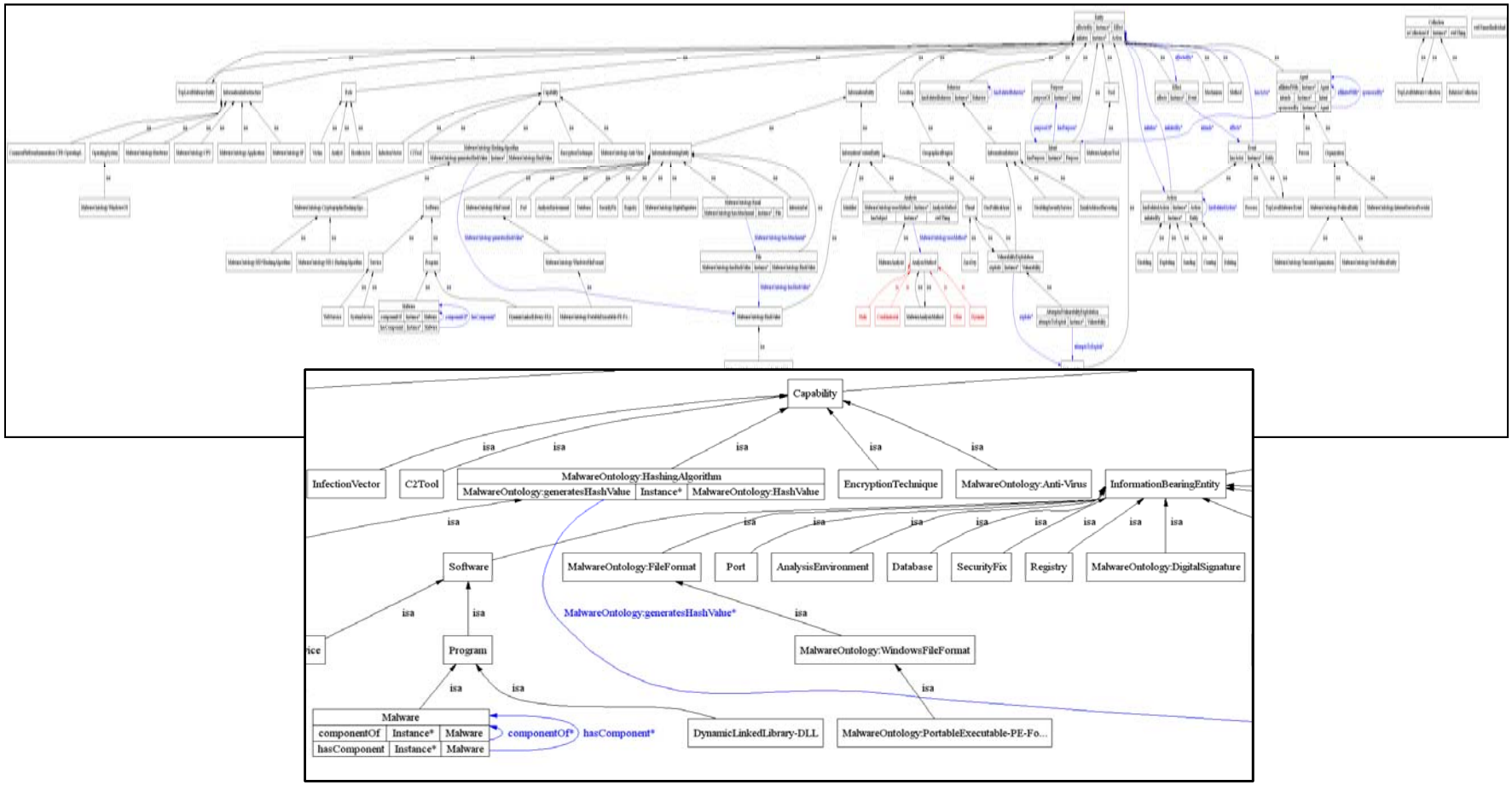
Response: HTTP/1.1 200 OK

MITRE

MAEC Schema v 1.1 (XML)

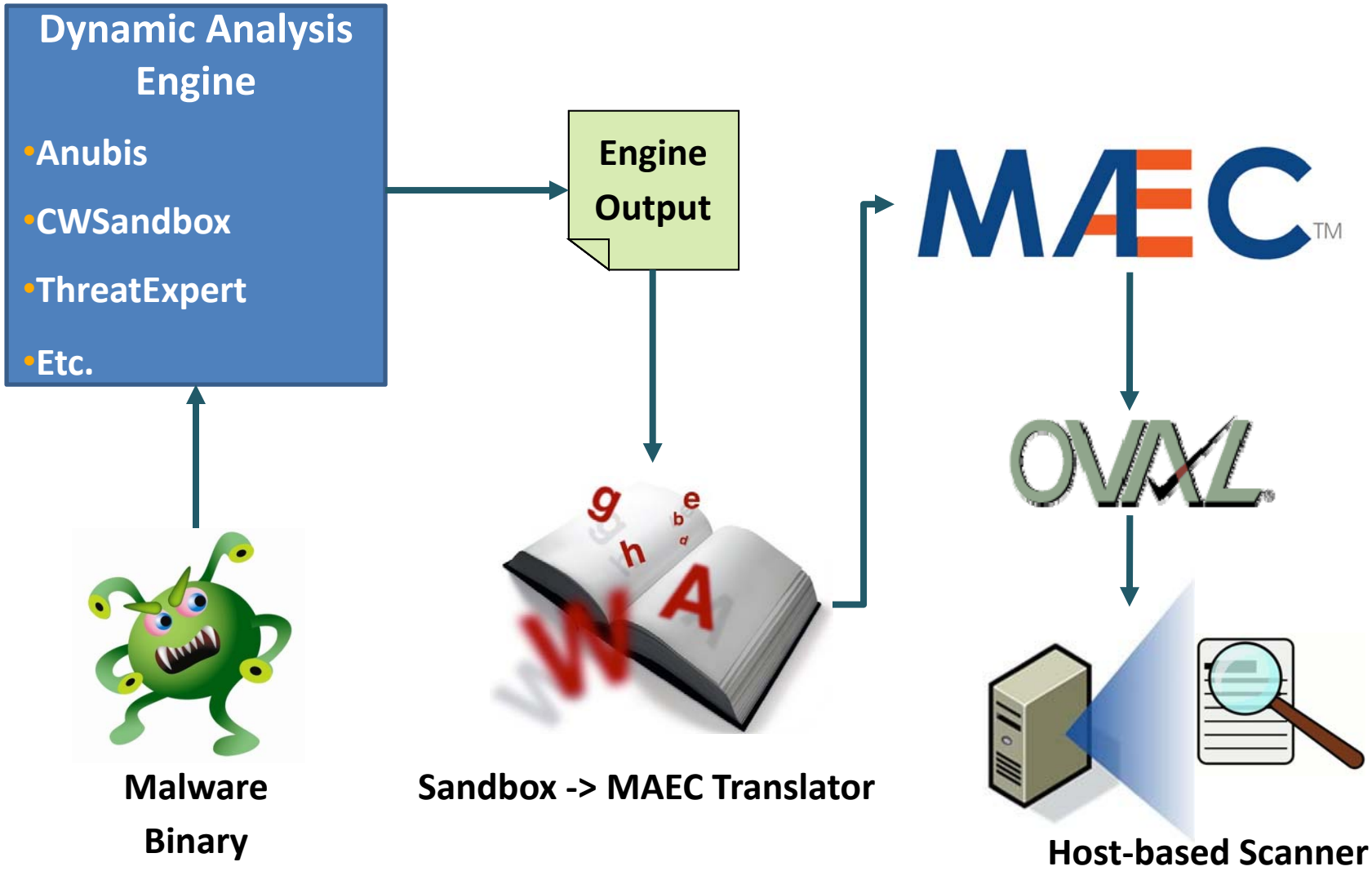


Malware Ontology (OWL)

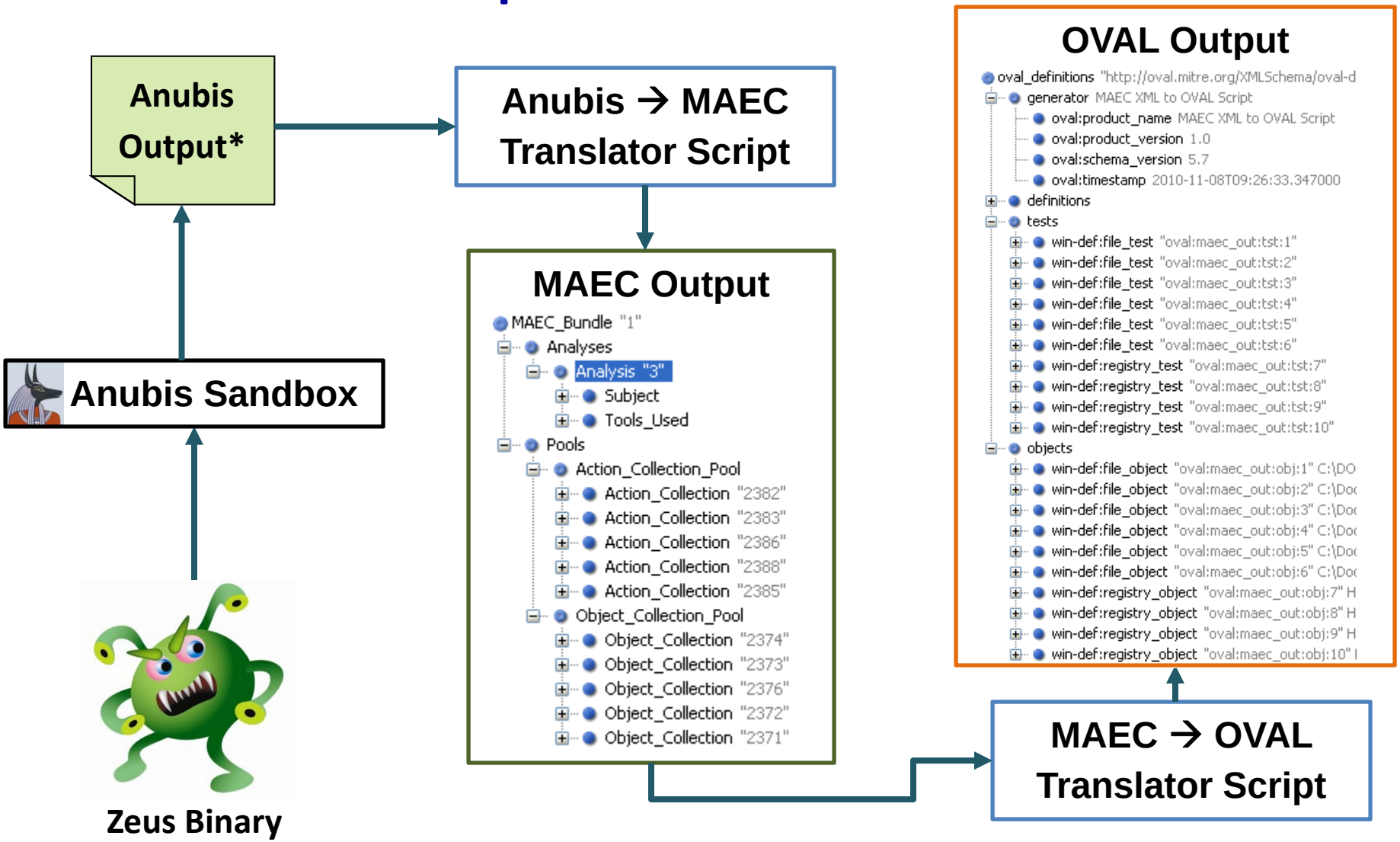


MITRE

Use Case: Host Based Detection

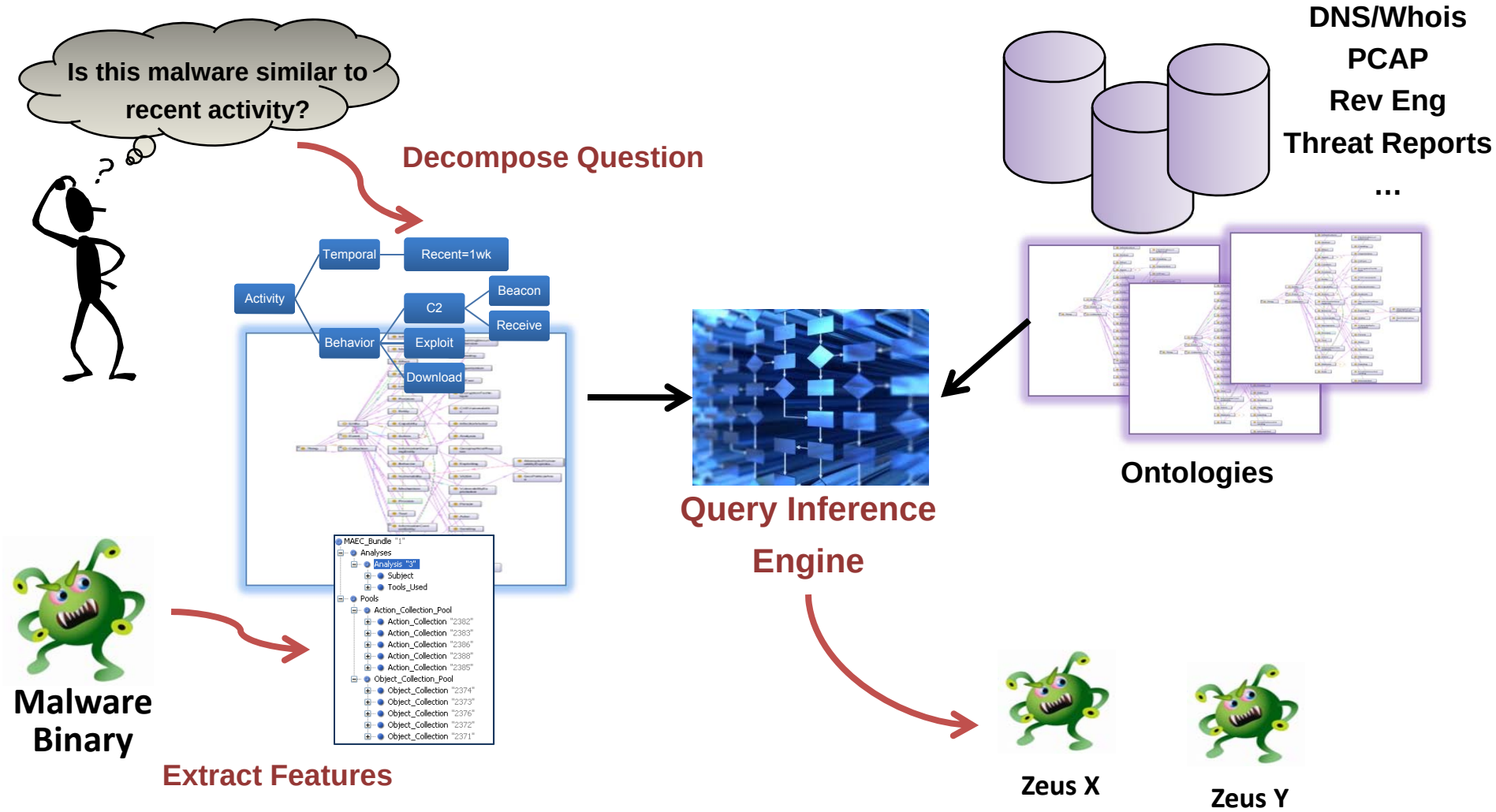


Real World Example: MAEC & Zeus Bot



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

Use Case: Data Fusion & Correlation



MAEC™ v2.0 Changes

- MAEC object model replaced with CybOX
- ActionType simplified
- EffectType refined, made easier to use
  - More object-focused
  - Permits definition of much more complex effects
- Lots of 'under the hood' tweaks and minor additions
  - MAEC Bundle now supports storage of Objects, Actions, Behaviors, and Analyses at the top level
  - Additional analysis environment attributes added to Analysis Type
  - Enumerations made into separate types
  - Main entities streamlined to remove unnecessary hierarchy
  - Many more

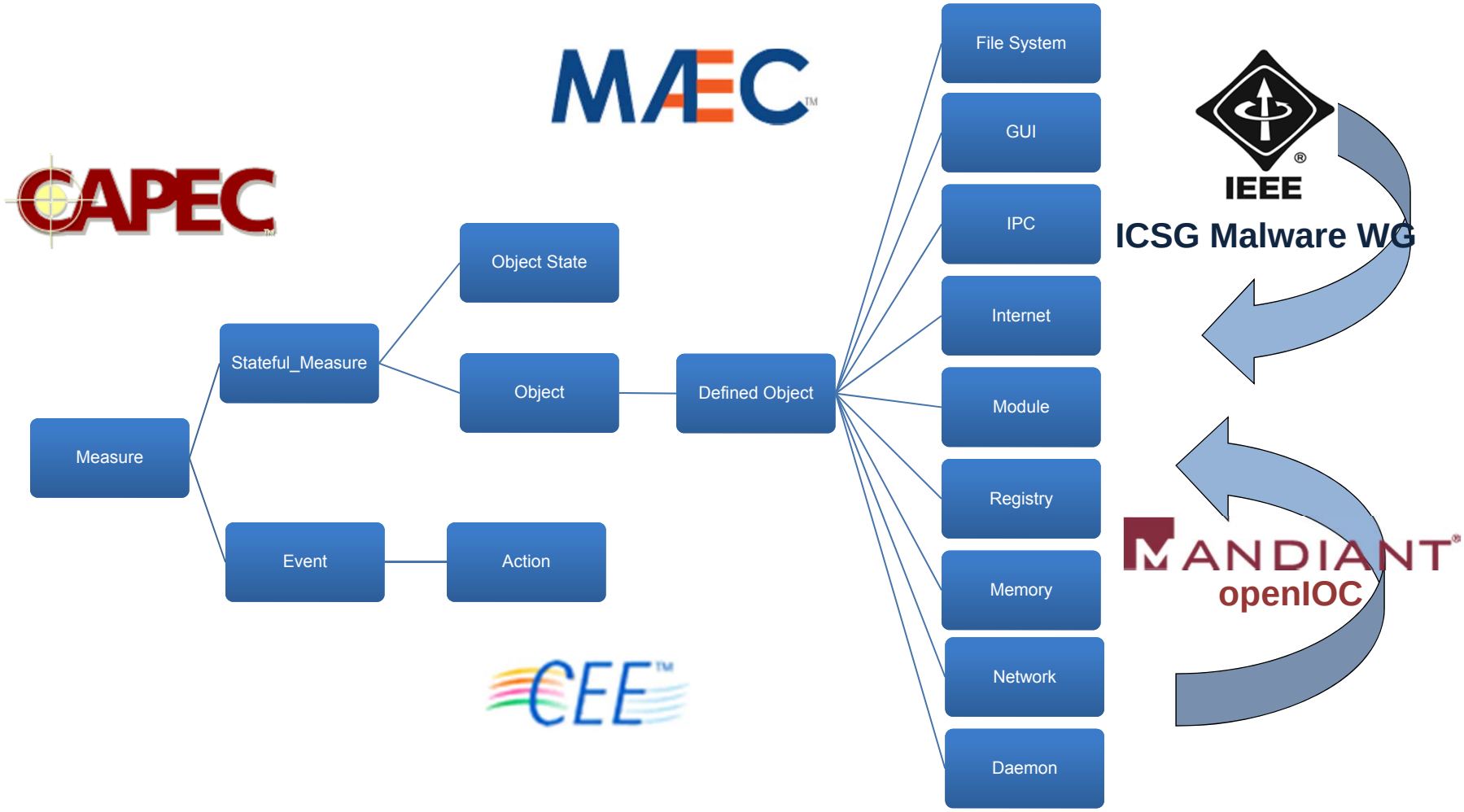
MAEC™ v2.0 Additions

- **Signature/Indicator Management Capability**
  - **Permits standard method of defining anti-malware signatures and indicators. Examples: OpenIOCs, ClamAV, Yara, Snort, etc.**
  - **Linkages to other MAEC entities where appropriate. E.g. objects for specifying signature/indicator used in detection.**

- **Relationship Support**
  - **Allows defining simple relationships between MAEC entities in an easy to use fashion. Examples: ParentOf, ChildOf, PrecededBy, etc.**

- **Many new enumerated types**
  - **Actions, Effects, Relationships, etc.**

Common Cyber Observables (CybOX) Schema



MAEC v2.0 Objects (imported from CybOX)

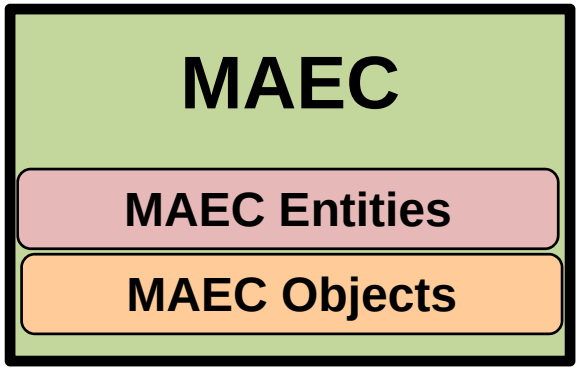
- Account
- Disk
- Disk Partition
- DNS Cache
- Email Message
- File
- GUI
- Library
- Package
- Memory
- Network Connection
- Network Route
- Linux Package
- Product
- Service
- Socket
- System
- User Session
- Volume
- Win Critical Section
- Win Driver
- Win Event
- Win Event Log
- Win Kernel
- Win 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

…

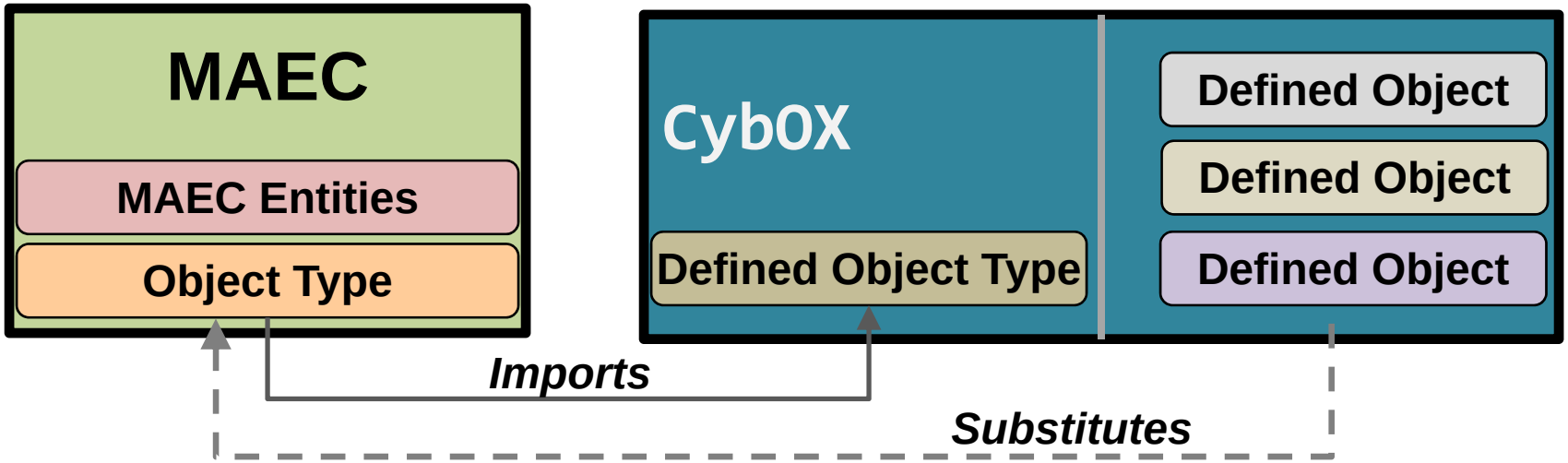
(more on the way)

MAEC & CybOX

- Before (MAEC 1.x)



- After (MAEC 2.0 and up)



Community Engagement

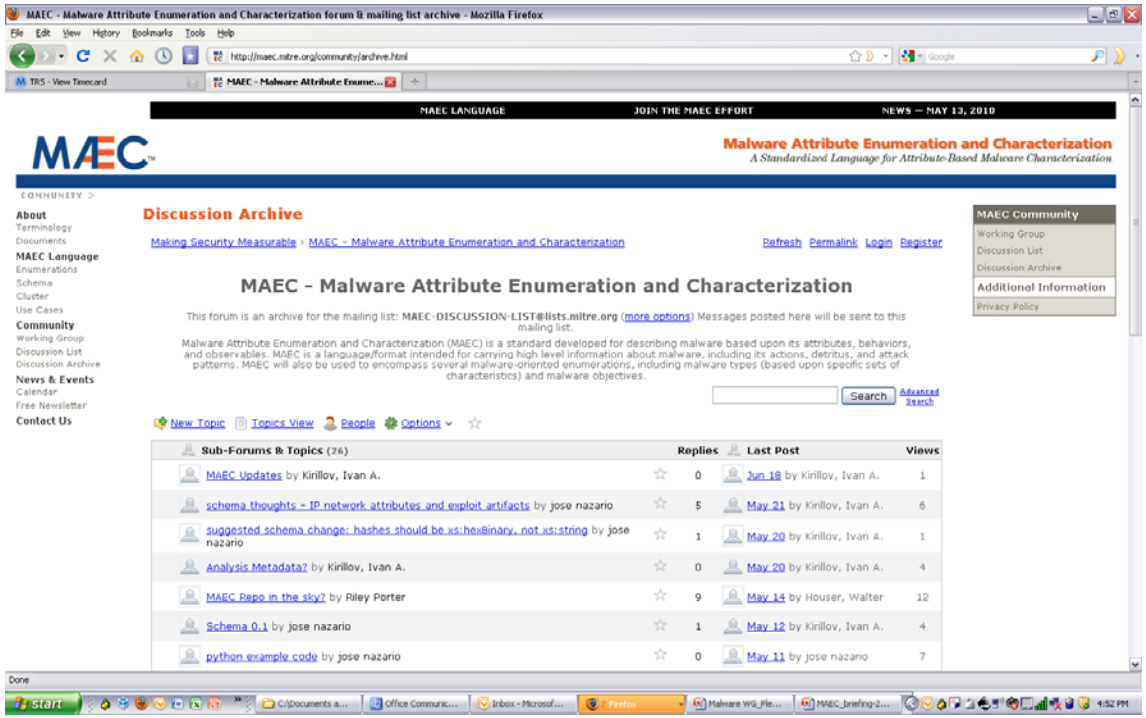
- Industry Collaborations
  - Working with Mandiant on MAEC <-> openIOC
  - Tool vendors supported our development of MAEC translators:
    - CWSandbox : GFI Software
    - ThreatExpert : Symantec
    - Anubis : International Secure Systems (Isec) Lab
  - Discussions with tool vendors about adopting MAEC as a native output format (under NDAs)
  - Malware analysts experimenting with MAEC (e.g., to compare multiple tool output)

Community Engagement

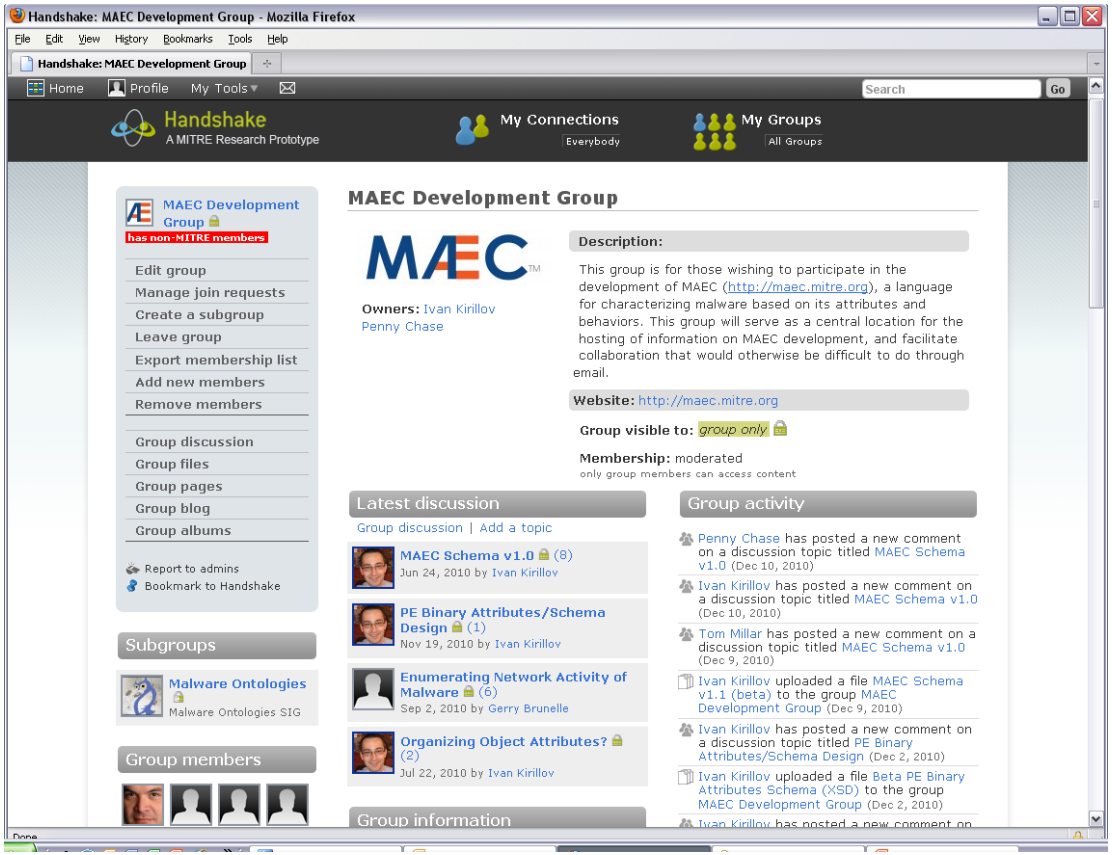
- IEEE Industry Connections Security Group (ICSG)
  - Malware Working Group developed an exchange schema to facilitate the sharing of sample data between AV product vendors
    - MAEC currently imports the IEEE ICSG Malware Metadata exchange schema
  - Recently established Malware Metadata Exchange Format 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

MAEC Community: Discussion List

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



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**

CAPEC™

Questions?

sbarnum@mitre.org
pc@mitre.org

MAEC™