ioc_writer

OpenIOC 1.1 and tools for creating them

PRESENTED BY: William Gibb (william.gibb AT mandiant.com)

AUGUST 1ST, 2013

OpenIOC 1.0

- Open standard for encapsulating threat intelligence
- Allows for a common format for different parties to use to share threat intelligence
- Open sourced in Fall of 2011
- Supported by Mandiant Intelligent Response®, Mandiant Redline™, IOC Finder
- Couple of pitfalls
  - We didn't publish any open tools to process OpenIOC documents
  - We were bounded by limitations of Lucene
  - We only had a limited number of operators (is / contains)

OpenIOC 1.1

- Kept what worked, added what we needed
- Some parts of the schema were reorganized
- Added new operators
  - IS is IS
  - Contains is now a true substring match
  - Begins-with / ends-with
  - Greater-than / less-than
  - Matches => This is regular expressions!
  - Case sensitivity
- Operator functionality is now based on XPATH 2.0 functions, not on Lucene
- OpenIOC 1.1 Draft materials available upon request

Parameters!

- Parameters have been added, and can be attached to Indicator or IndicatorItem nodes
- This allows embedding additional metadata about Indicators into the IOC
  - Comments, Confidence, Criticality
- This allows for prototyping features prior to inclusion to the OpenIOC schema
- This also allows adding in application-specific processing directives to the IOC
  - Can use this to extend the capabilities of OpenIOC without have to make schema changes
  - We will see this later with OpenIOC to YARA conversion

ioc_writer – what is it?

- Tl;dr – python library to build OpenIOC 1.1 objects
- No need to worry about understanding a schema or writing XML by hand
- Can create IOCs through API calls
- Built on top of the python lxml library – need have that installed!
- Supports a set of CRUD operations
  - Creating, updating and deleting components are directly supported
  - Reading is supported through the elementTree interface offered by lxml.etree and through the use of XPATH
- Available @ **Github.com/mandiant/ioc_writer**

Easy to use

- Easy to start using – full HTML documentation available
- Installation is easy
  - Python setup.py install
- Things to care about in the library and repository
  - ioc_writer
    - ioc_api.py -> Contains IOC class, few helper functions
    - ioc_common.py -> Contains helper functions to build IndicatorItem nodes with predefined values
  - Examples
    - simple_ioc_writer -> Example of creating IOCs from lists of data
    - openioc_to_yara -> Example of processing an IOC based on parameters

Easy to use

- Easy to start using – full HTML documentation available
- Example opening a IOC file and adding a term to an IOC
  - from ioc_Writer import ioc_api, ioc_common
  - fp = ‘123412341234-1234-1234-12341234.ioc’
  - ioc_obj = ioc_api.IOC(fn=fp)
  - evil_file = ‘evil.exe’
  - node = ioc_common. make_fileitem_filename(evil_file)
  - ioc_obj.top_level_indicator.append(node)
  - ioc_obj.write_ioc_to_file()
- No knowledge of OpenIOC schema / XML required!

Simple_ioc_writer.py demo

- Simple_ioc_writer.py demo
  - Quickly convert a list of items into OpenIOC format
  - This was done for rapid generation of the IOC released based on the Nettraveler campaign

Integrating with other tools

- Since a standard API is provided, it can be used to easily integrate with other tools
  - Sandboxes
    - Have a summary report from a sandbox?
    - Process it to automatically generate an IOC!
  - Custom editors
    - Could use this as a basis for a simple Python IOC editor
  - Text-scrappers
    - Automatically scrape text for IOCs and store them into OpenIOC
  - Could integrate it into malware analysis workflow to generate IOCs during analysis
- Demo – IOCextractor

YARA

- Parameters allow us to define application specific directives
- YARA is a natural candidate for encapsulating in OpenIOC format
  - At its core, YARA is a set of strings and conditions placed upon them in the form of a boolean expression
  - OpenIOC is ideal for storing data that can be expressed as boolean expressions
- The trick is figuring out how to:
  - Store all the parts of a signature in OpenIOC sanely
  - Extract a useful signature out of the OpenIOC

YARA continued - implementation

- YARA specifies either a list of strings and conditions for them, or just a condition, required to match.
  - This differs from OpenIOC, which specifies the items to match in the structure in which they match.
- This requires processing the YARA IndicatorItems in order to build the strings of interest. Parameters are used to determine additional modifiers such as ‘fullword’, ‘wide’, ‘ascii’; while built in case-sensitivity was used for ‘nocase’
- YARA hex strings, text strings, and regular expressions cleanly map into IndicatorItem terms. Some conditions, like FileSize and Rule names, also map to IndicatorItem terms.

YARA continued - implementation

- Generating the condition string requires walking the IOC logic tree, which branches at Indicator nodes [the AND and OR parts of the IOC], in order to generate the condition.
  - Each Indicator node could potentially generate a set condition
  - Each IndicatorItem node may have conditions applied to it, through parameters, such as a ‘count’ condition.
  - An IndicatorItem node may actually represent a filesize or another rule, which does not have a corresponding YARA string
  - All of these can affect the resulting condition expression, and must be considered
  - Recursion for the win ☺

YARA continued

- Supported
  - YARA text strings, hex strings and regular expressions
  - Set conditions, referencing other rules, file sizes
  - Offsets and counting
  - OpenIOC to YARA format
- Not currently supported
  - Accessing data at given positions
  - For expressions
  - Includes
  - YARA rules to OpenIOC format
- YARA demo

YARA IOC terms and parameters

IOC Terms

- String Terms
  - Yara/HexString
  - Yara/TextString
  - Yara/Regex
- Condition Terms
  - Yara/FileSize
  - Yara/RuleName

Parameters

- String modifiers
  - yara/wide
  - yara/ascii
  - yara/fullword
- Condition modifiers
  - yara/set
  - yara/count
  - yara/offset/at
  - Yara/offset/in

Snort

- Proved to be a more difficult translation than anticipated
- Difficult to maintain the agility of OpenIOC while maintaining Snort SID rules
- Still room for future work though…

OpenIOC 1.1 to 1.0

- ioc_writer does include a script, allowing users to downgrade OpenIOC 1.1 IOCs into the OpenIOC 1.0 format
  - Allows use of 1.1 IOCs in tools that consume 1.0 IOCs
  - This is lossy!
    - Certain operators, like greater-than, less-than, matches, do not translate into OpenIOC 1.0
    - Since there is no equivalent term, these are not included in the output.

Questions?

Get ioc_writer

- ioc_writer can be obtained on Github!
  - **Github.com/mandiant/ioc_writer**
- It does require lxml
  - https://pypi.python.org/pypi/lxml
  - http://lxml.de/