MISP RESTSEARCH MODULE DEVELOP-MENT

BUILDING A SIMPLE EXPORT MODULE FOR THE CORE

CIRCL / TEAM MISP PROJECT



13TH ENISA-EC3 WORKSHOP



MISP restSearch module development

NG A SIMPLE EXPORT MODULE FOR THE CORE

MISP RESTSEARCH MODULE DEVELOP





BUILDING A NATIVE RESTSEARCH EXPORT

■ Similar in scope to an **export module** of the MISP modules system

Pros:

- ► Can be used for composited data coming from a **filtered query**
- Fast, native approach
- ► Can be built to support **several scopes** (events, attributes, sightings)
- Cons...

MISP restSearch module development

-Building a native restSearch export

NG A NATIVE RESTSEARCH EXPORT

Similar in scope to an **export module** of the MISP module: system Pros:

Can be used for composited data coming from a fitter
 Fast, mative approach
 Can be built to support several scopes (events, attrib sightings)

sightings)

Cons...

BUILDING A NATIVE RESTSEARCH EXPORT

■ Similar in scope to an **export module** of the MISP modules system

■ Pros:

- ► Can be used for composited data coming from a **filtered query**
- ► Fast, native approach
- Can be built to support several scopes (events, attributes, sightings)
- Cons...



MISP restSearch module development

Building a native restSearch export

LDING A NATIVE RESTSEASCH EXPORT

Similar in scope to an export module of the MISP modules yetten

Local be used for composited data coming from a filtered query

Figs., and suppress.

Figs., and suppress.

It is propriet several scopes (events, attributes, judgisting).

SO HOW DOES RESTSEARCH WORK?

- Standardised way of collecting parameters
- Using the parameters, a loop is started to chunk and gradually build our export data
- The chunk size depends on memory envelopes
- Each chunk is **converted piece by piece**...
- ... and subsequently are concatenated into a temporary file
- Once no more elements are left, the file is sent in the response

MISP restSearch module development

—So how does restSearch work?

SO HOW DOES RESTSEARCH WORK?

Standardised way of collecting **parameters** Using the parameters, a loop is started to **chunk an** Stradually build our export data

Each chunk is converted piece by piece...

 Once no more elements are left, the file is sent in the response

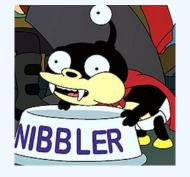
WHERE DOES THE MODULE SYSTEM COME INTO PLAY?

- The export modules handle 5 tasks:
 - ► Pass **meta-information** back to restSearch on the export format itself
 - Add a **start segment** to the exported data
 - ▶ Do the actual **conversion** from MISP's internal format to the desired export format
 - Provide a **separator** for data chunks
 - ► Have a **closing segment** for the returned data, based on the formats conventions

MISP restSearch module development

-Where does the module system come into play?

OUR LITTLE TRAINING MODULE: NIBBLER, THE EVER HUNGRY IDS/IPS



MISP restSearch module development

-Our little training module: Nibbler, the ever hungry IDS/IPS





NIBBLER

- Simplistic tool with its **own proprietary format**
- Meant to mimic a typical in-house tool
- Lightweight scope, for simplicityś sake
- **■** pipe separated values
- VALUE | TYPE | DESCRIPTION | REFERENCE | ACTION

MISP restSearch module development

Simplicit tool with its own proprietary format
Mibbler

Nibbler

Nibbler

NIBBLER FORMAT - CAVEATS

- Rules can be prepended by comments, each comment line starting with #
- Some characters have to be escaped in some custom, crazy ways

► linebreaks: ##LINEBREAK##

commas: ##COMMA##

▶ pipes: ##PIPE##

MISP restSearch module development

-Nibbler format - caveats

FORMAT - CAVEATS

Rules can be prepended by comments, each comment line starting with #
Some characters have to be escaped in some custom, craz

► linebreaks: MILINEBREAKIN ► commas: MICOMMANN

NIBBLER FORMAT

- Value: The actual indicator value
- **Type**: The format of the indicator
- **Description**: A quick description for analysts investigating the alert, why is this relevant
- **Reference**: A backreference that the analyst can use to find out more about the alert
- Action: What should Nibbler do if it trips over the value?

MISP restSearch module development

20-01-4202

-Nibbler format

NIBBLER FORMAT

- # Value: The actual indicator value # Type: The format of the indicator
- Type: The format of the indicator
 Description: A quick description for analysts investigating
- the alert, why is this relevant
 Reference: A backreference that the analyst can use to find
- out more about the alert
- m Action: What should Nibbler do if it trips over the value?

SUPPORTED TYPES

- IP
- Domain
- Hostname
- MD5
- SHA1
- SHA256
- Filename

MISP restSearch module development

—Supported types

2024-10-02

■ IP
■ Domain
■ Hostname
■ MD5
■ SHA1
■ SHA256
■ Filename

SUPPORTED VALUES

- ALERT default behaviour, create an alert.
- BLOCK block the action outright. Only set if the tag nibbler:block is present

MISP restSearch module development

—Supported values

JPPORTED VALUES

ALERT - default behaviour, create an alert.

 BLOCK - block the action outright. Only set if the tag

MAPPING THE TYPES TO MISP

- Though we have types to map from MISP, in some cases several types map to a Nibbler type
- We've created a rough **mapping** (this is probably the most difficult task) in advance
- Some MISP types map to a Nibbler type directly
- Composite MISP types map to 2 Nibbler types each

MISP restSearch module development

-Mapping the types to MISP

NG THE TYPES TO MISP

Though we have types to map from MISP, in some cases several types map to a Nibbler type

difficult task) in advance

Some MISP types map to a Nibbler type directly
 Composite MISP types map to 2 Nibbler types each

MAPPING THE TYPES TO MISP

- ip-dst :: IP
- ip-src :: IP
- domain :: Domain
- domain|ip :: Domain, IP
- hostname :: Hostname
- md5 :: MD5
- sha1 :: SHA1
- sha256 :: SHA256
- filename | md5 :: Filename, MD5
- malware-sample :: Filename, MD5
- filename|sha1 :: Filename, SHA1
- filename|sha256 :: Filename, SHA256

MISP restSearch module development

-Mapping the types to MISP

ip-dst :: IP

- m ip-src :: IP domain :: Domain

- shar :: SHAr

EXPORT MODULE SKELETON

```
<?php
class NibblerExport
    public $additional_params = array();
    public function handler(
        $data, $options = array()
    public function header(
        $options = array()
    public function footer() {}
    public function separator() {}
```

MISP restSearch module development

Export module skeleton

ADDITIONAL PARAMETERS

```
public $additional_params = array(
    'flatten' => 1
);
```

MISP restSearch module development

—Additional parameters

ublic Sadditional_params = array('flatten' => 1

ADDING OUR MAPPING

```
private $__mapping = array(
  'ip-dst' => 'IP',
  'ip-src' => 'IP'.
  'domain' => 'Domain'.
  'domain|ip' => ['Domain', 'IP'],
  'hostname' => 'Hostname',
  'md5' => 'MD5'.
  'sha1' => 'SHA1'.
  'sha256' => 'SHA256'.
  'filename|md5' => array('Filename', 'MD5'),
  'malware-sample' => array('Filename', 'MD5'),
  'filename|sha1' => array('Filename', 'SHA1'),
  'filename|sha256' => array('Filename', 'SHA256')
```

MISP restSearch module development

—Adding our mapping

private 5_mmpping = array(
'[p-dat' => [P']
'[p-set' => [P']
'domain' => 'Domain',
'domain' => 'SMASS,
'domain' => 'MASS',
'domain' => 'SMASS,
'domain' => 'MASS',
'domain'

```
public function header($options = array())
    return sprintf(
        "# Nibbler rules generated by MISP at %s\n",
        date('Y-m-d H:i:s')
```

FOOTER FUNCTION - HOW SHOULD THE OUTPUT END?

```
public function footer()
{
    return "\n";
}
```

MISP restSearch module development

Footer function - how should the output end?

blic function footer()

17

WHAT SEPARATES THE CHUNKS?

```
public function separator()
{
    return "\n";
}
```

MISP restSearch module development

-What separates the chunks?

ublic function separator()
return "\n";

THE ACTUAL LEGWORK, THE HANDLER

```
public function handler($data, $options = array())
{
   if ($options['scope'] === 'Attribute') {
        $data['Attribute']['AttributeTag'] = $data['AttributeTag'];
        return $this->__convertAttribute($data['Attribute'], $data['Event']);
   }
   if ($options['scope'] === 'Event') {
        $result = array();
        foreach ($data['Attribute'] as $attribute) {
            $temp = $this->__convertAttribute($attribute, $data['Event']);
            if ($temp) $result[] = $temp;
        }
        return implode($this->separator(), $result);
    }
   return '';
}
```

MISP restSearch module development

└─The actual legwork, the handler

patte tentes baster(pless, lapram = serg())

of (factors) tamed | m = relation | 1 |

of (factors) tamed | m = relation | 1 |

of (factors) tamed | m = relation | 1 |

of (factors) tamed | m = relation | 1 |

of (factors) tamed | m = relation | 1 |

tames factors | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1 |

or (factors) tamed | m = relation | 1

FUNCTION

```
private function __convertAttribute($attribute, $eve
 if (empty($this->__mapping[$attribute['type']])) {
   // mapping not found - invalid type for nibbler
   return '':
 if (is_array($this->__mapping[$attribute['type']])
   // handle mappings for composites - slide
   else {
   // handle simple mappings - slide
  // return 1 or 2 lines, separated by separator()
 return implode($this->separator(), $result);
```

MISP restSearch module development

Building an optional internal converter function

DING AN OPTIONAL INTERNAL CONVERTER
TION

te function __convertAttribute(Sattribute, Seve

private function _convertAttribute(\$attribute, \$e[if (empty(\$this->_mapping[\$attribute['type']])] // mapping onf found - invalid type for nibble _converted for the found invalid type for nibble _converted for the found in the found for the

HANDLING THE SIMPLE CASE

```
$result[] = sprintf(
   '%s|%s|%s|%s',
   $this->__escapeSpecialChars($attribute['value']),
   $this->_mapping[$attribute['type']],
   $event['uuid'],
   $this->__escapeSpecialChars($event['info']),
   'ALERT'
);
```

MISP restSearch module development

Handling the simple case

ING THE SIMPLE CASE

result[] = sprintf('%clsclsclsclsclsc'

\$this->__escapeSpecialChars(\$attribute['val \$this->__mapping[\$attribute['type']],

\$tnis->_mapping[sattribute['type']], \$event['uuid'], \$this->__escapeSpecialChars(\$event['info' 'ALERT'

HANDLING THE CASE FOR COMPOSITES

```
$attribute['value'] = explode(
  '|'. $attribute['value']
foreach (array(0,1) as $part) {
 $result[] = sprintf(
    '%s|%s|%s|%s|%s'.
    $this -> __escapeSpecialChars(
      $attribute['value'][$part]
    $this -> __mapping[$attribute['type']][$part],
    $event['uuid'].
    $this -> __escapeSpecialChars($event['info']),
    'ALERT'
```

MISP restSearch module development

-Handling the case for composites

PUTTING IT TOGETHER

```
private function convertAttribute($attribute, $event) {
  if (empty($this->__mapping[$attribute['type']])) return '';
  $result = array();
  $attributes = array();
  if (is array($this-> mapping[$attribute['type']])) {
    $attribute['value'] = explode('|', $attribute['value']);
   foreach (array(0,1) as $part) {
     $result[] = sprintf(
       '%s|%s|%s|%s|%s',
        $this -> escapeSpecialChars($attribute['value'][$part]),
        $this -> mapping[$attribute['type']][$part],
        /events/view/ . $event['uuid'],
        $this->__escapeSpecialChars($event['info']),
       $this -> decideOnAction($attribute['AttributeTag'])
   else
    $result[] = sprintf(
     '%s|%s|%s|%s|%s',
     $this ->__escapeSpecialChars($attribute['value']),
     $this ->__mapping[$attribute['type']],
      /events/view/ . $event['uuid'],
     $this -> escapeSpecialChars($event['info']).
     $this ->__decideOnAction($attribute['AttributeTag'])
 return implode($this->separator(), $result);
```

MISP restSearch module development

-Putting it together

ADDING THE FUNCTION THAT DECIDES ON THE ACTION

```
private function __decideOnAction($attributeTags)
 foreach($attributeTags as $attributeTag) {
      $attributeTag['Tag']['name'] ===
        'nibbler:block'
      return 'BLOCK';
  return 'ALERT';
```

MISP restSearch module development

LAdding the function that decides on the action

TION THAT DECIDES ON THE ACTION

__decideOnAction(\$attributeTags)

FINALISING THE EXPORT MODULE... THE ESCAPING FUNCTION

```
private function __escapeSpecialChars($value)
 $value = preg_replace(
   "/\r|\n/", "##LINEBREAK##", $value
 $value = preg_replace(
   "/,/", "##COMMA##", $value
 $value = preg_replace(
   "/\|/", "##PIPE##", $value
 return $value;
```

MISP restSearch module development

Finalising the export module... The escaping function

MODIFYING THE MISP CORE TO KNOW ABOUT THE EX-PORT MODULE

- The **models** that we are targeting by scope (Event, Attribute) **need to be updated**
- They are located in /var/www/MISP/app/Model/
- The global variable **\$validFormats** houses all mappings
- Simply add a new line such as the following:
- 'nibbler' => array('nibbler', 'NibblerExport', 'nibbler')

MISP restSearch module development

2024-10-0

-Modifying the MISP core to know about the export module

YING THE MISP CORE TO KNOW ABOUT THE EX-MODULE

- m The models that we are targeting by scope (Event, Attribute)
- They are located in /var/www/MISP/app/Model/
- The global variable SvalidFormats houses all m
- Simply add a new line such as the following:
- "nibbler" => array('nibbler', 'NibblerExport', 'nibl

LET US TEST THE MODULE!

- Use the **rest client** to test it conveniently
- Both the event and attribute level restSearch function should work
- Simply set the **returnFormat** to nibbler, which should also show up as a valid export format

MISP restSearch module development

Let us test the module!

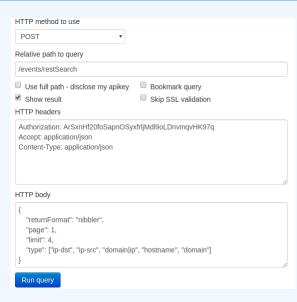
T US TEST THE MODULE!

m Use the rest client to test it conveniently

 Both the event and attribute level restSearch function should work

 Simply set the returnFormat to nibbler, which should also show up as a valid export format

REST CLIENT



MISP restSearch module development

-REST client

