SPECTEROPS

ShareHound

Mapping Network Share Rights into Bloodhound with OpenGraph

Remi GASCOU (Podalirius) | rgascou@specterops.io

A blind spot in data collection: Network shares

- Network shares contains lots of useful data (passwords, scripts, information on the targets …)
- Easy low hanging fruits for ransomwares, targeting them to encrypt their content
- Mapping network shares is hard. We would need to have a specific collector and import it into a graph visualization software
- Since the release 8.0 of BloodHound, OpenGraph was introduced and allows us to import custom collections!

Why collecting network shares data is a hard problem

- Many machines to target in the network, lots of possible bottlenecks between network hops.
- Access Control Lists everywhere!
- Very fast snowballing of output size if no filters are applied.

A Domain Specific Language for network shares

To address a wide range of possible use cases, we need to have a language of sorts dedicated to network shares. This is the goal of the ShareQL domain specific language we released alongside ShareHound (https://github.com/p0dalirius/shareql). This language works similarly to the rules of a firewall. Each rule is evaluated sequentially from the first one and when one rule matches, evaluation is stopped.

rules > skip_common_shares.shareql

```
1  DENY EXPLORATION IF SHARE.NAME IN ['c$','print$','ipc$','admin$']
2  ALLOW EXPLORATION
```

As a small bonus to end users, we also provide a syntax highlighting extension for the ShareQL language in VSCode: https://github.com/p0dalirius/shareql-vscode-ext

Source: https://github.com/p0dalirius/shareql

A Domain Specific Language for network shares

- **Basic Rule Structure:** `ACTION [OPERATION] [IF condition]`
- **ACTION**: `ALLOW` or `DENY`
- **OPERATION**: `ALL`, `PROCESSING`, or `EXPLORATION`
- **CONDITION**: Boolean expression using available fields and operators on fields of objects

Source: https://github.com/p0dalirius/shareql

A Domain Specific Language for network shares

File Fields:

- `FILE.MODIFIED_AT` - Last modification timestamp
- `FILE.CREATED_AT` - Creation timestamp
- `FILE.SIZE` - File size in bytes
- `FILE.NAME` - File name
- `FILE.PATH` - Full file path

Other Fields:

- `DEPTH` - Directory depth level

Share Fields:

- `SHARE.NAME` - Share name
- `SHARE.DESCRIPTION` - Share description
- `SHARE.TYPE` - Share type

Directory Fields:

- `DIRECTORY.PATH` - Full directory path
- `DIRECTORY.NAME` - Directory name
- `DIRECTORY.MODIFIED_AT` - Last modification timestamp
- `DIRECTORY.CREATED_AT` - Creation timestamp

Source: https://github.com/p0dalirius/shareql

A Domain Specific Language for network shares

Operators

- `MATCHES` - String matching
- `IN` - Check if value is in a list
- `>=`, `<=`, `>`, `<`, `==` - Comparison operators
- `STARTSWITH` - String starts with pattern
- `ENDSWITH` - String ends with pattern
- `CONTAINS` - String contains pattern

Values

- Strings: `"quoted string"` or `'single quoted'`
- Numbers: `1234`
- Lists: `["item1", "item2", "item3"]`
- Regex: `REGEX("pattern.*")`

Source: https://github.com/p0dalirius/shareql

A Domain Specific Language for network shares

Interested in using this domain specific language into your own tool? You can!

```
from shareql.grammar.parser import RuleParser
from shareql.evaluate.evaluator import RulesEvaluator


# Parse rules from text
parser = RuleParser()
rules, errors = parser.parse("""
DENY PROCESSING IF FILE.SIZE >= 1000
ALLOW EXPLORATION IF DIRECTORY.NAME MATCHES "public"
""")


# Create evaluator
evaluator = RulesEvaluator(rules)


# Evaluate against target objects
# (target_object should be a RuleObjectFile, RuleObjectDirectory, or RuleObjectShare)
rule, allowed, result = evaluator.evaluate(target_object)
```

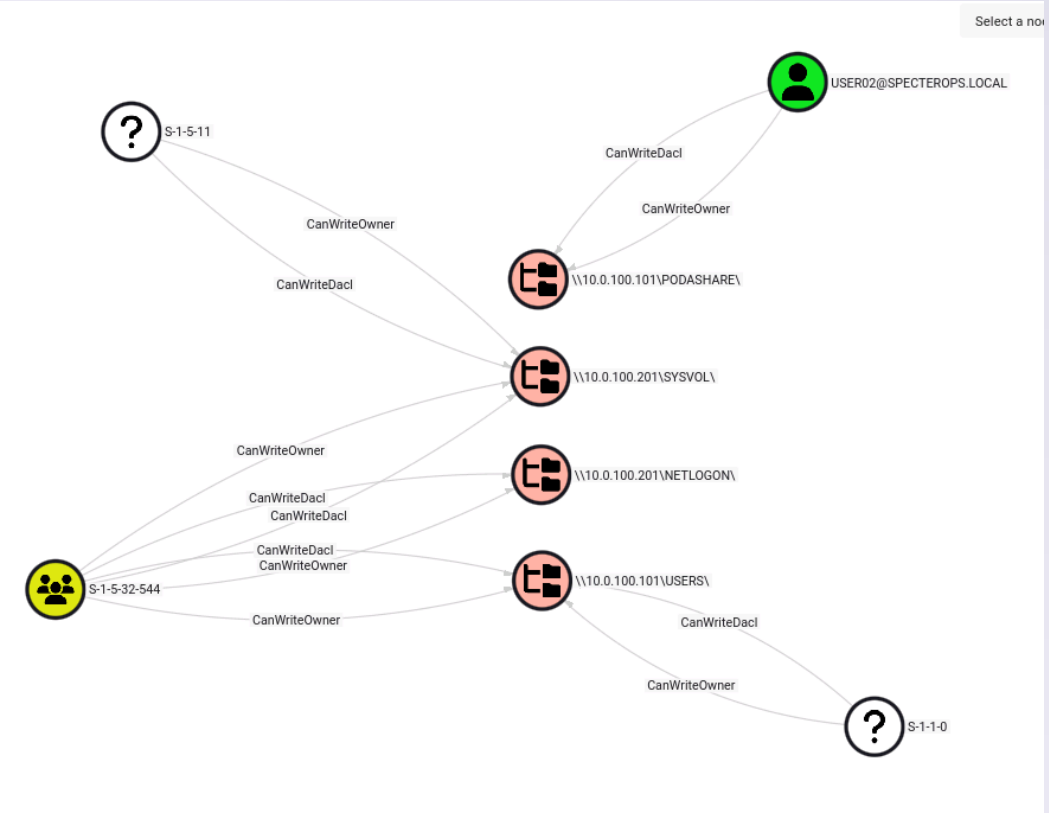
Source: https://github.com/p0dalirius/shareql

Finding Principals with Write Access to a Network Share

To find principals with Write access to a network share, we need to look for the `WRITE_DAC`, `WRITE_OWNER`, `DS_WRITE_PROPERTY`, and `DS_WRITE_EXTENDED_PROPERTY` rights.

We can write this simple cypher query to achieve this using the edges `CanWriteDacl`, `CanWriteOwner`, `CanDsWriteProperty`, and `CanDsWriteExtendedProperties`

```
MATCH x=(p)-
[r:CanWriteDacl|CanWriteOwner|CanDsWriteProperty|
CanDsWriteExtendedProperties]→(s:NetworkShareSMB)
RETURN x
```



Finding Principals with Full Control on Network Shares

Full Control is a set of rights containing all rights except the Generic rights. Therefore, we can create a Cypher query to find principals having `FULL_CONTROL` on network shares like this:

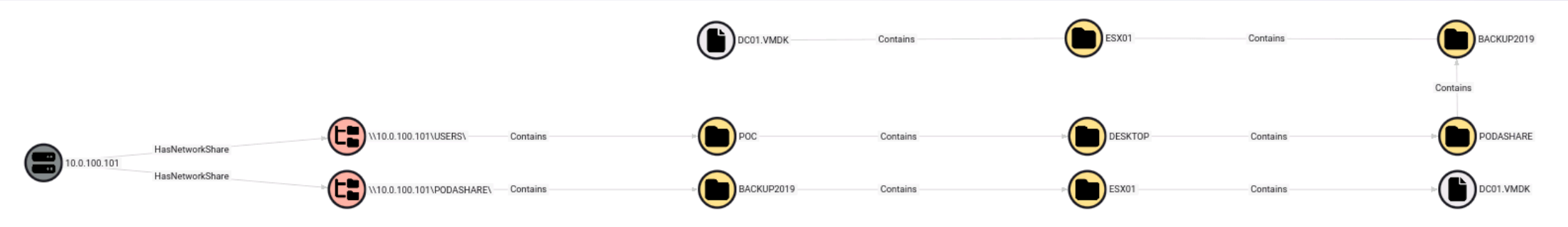
```
MATCH (p)-[r]→(s:NetworkShareSMB)
WHERE (p)-[:CanDelete]→(s)
  AND (p)-[:CanDsControlAccess]→(s)
  AND (p)-[:CanDsCreateChild]→(s)
  AND (p)-[:CanDsDeleteChild]→(s)
  AND (p)-[:CanDsDeleteTree]→(s)
  AND (p)-[:CanDsListContents]→(s)
  AND (p)-[:CanDsListObject]→(s)
  AND (p)-[:CanDsReadProperty]→(s)
  AND (p)-[:CanDsWriteExtendedProperties]→(s)
  AND (p)-[:CanDsWriteProperty]→(s)
  AND (p)-[:CanReadControl]→(s)
  AND (p)-[:CanWriteDacl]→(s)
  AND (p)-[:CanWriteOwner]→(s)
RETURN p,r,s
```

And, in result, we get this Onion looking graph on the share where my user has `FULL_CONTROL` rights:

Finding a VMDK File of a Domain Controller in a Share with Read Access

Full Control is a set of rights containing all rights except the Generic rights. Therefore, we can create a Cypher query to find principals having `FULL_CONTROL` on network shares like this:

```
MATCH p=(h:NetworkShareHost)-[:HasNetworkShare]->(s:NetworkShareSMB)-[:Contains*0..]->(f:File)
WHERE toLower(f.extension) = toLower(".vmdk")
RETURN p
```



SPECTEROPS

Thank you

Remi GASCOU (Podalirius) | rgascou@specterops.io