

v2

# Threat Hunting Professional

## Threat Hunting Hypothesis

Section 01 | Module 04

© Caendra Inc. 2020  
All Rights Reserved

# Table of Contents

## MODULE 04 | THREAT HUNTING HYPOTHESIS

### 4.1 MITRE ATT&CK

### 4.2 Data Collection and Analysis

### 4.3 Hunting Hypothesis and Methodology

### 4.4 Hunting Metrics

# Learning Objectives

In this module, we will discuss a threat hunting methodology that you can follow to start hunting in your environments.



# MITRE ATT&CK



## 4.1 MITRE ATT&CK

One of the points we outlined in a previous module was that a good threat hunter must have knowledge of various types of attacks and often a deep understanding of how they manifest, which helps with knowing how and where to detect them.

Knowing about attacks also means knowing where to search for them. In many cases, it will be required that you identify how to detect the said attack, which requires you to simulate it and to fill knowledge gaps otherwise not provided by your sources.

## 4.1 MITRE ATT&CK

The detection methods can be based on specific artifacts left on the Operating System or in the Network traffic generated, which you can use to figure out where detection points.

The sources for attacks can be comprehensive, such as the [MITRE ATT&CK](https://attack.mitre.org/), vendor reports, white papers, conference talks, individual researchers, your organization's offensive team, etc. In this chapter, we will focus on the MITRE ATT&CK.

## 4.1 MITRE ATT&CK

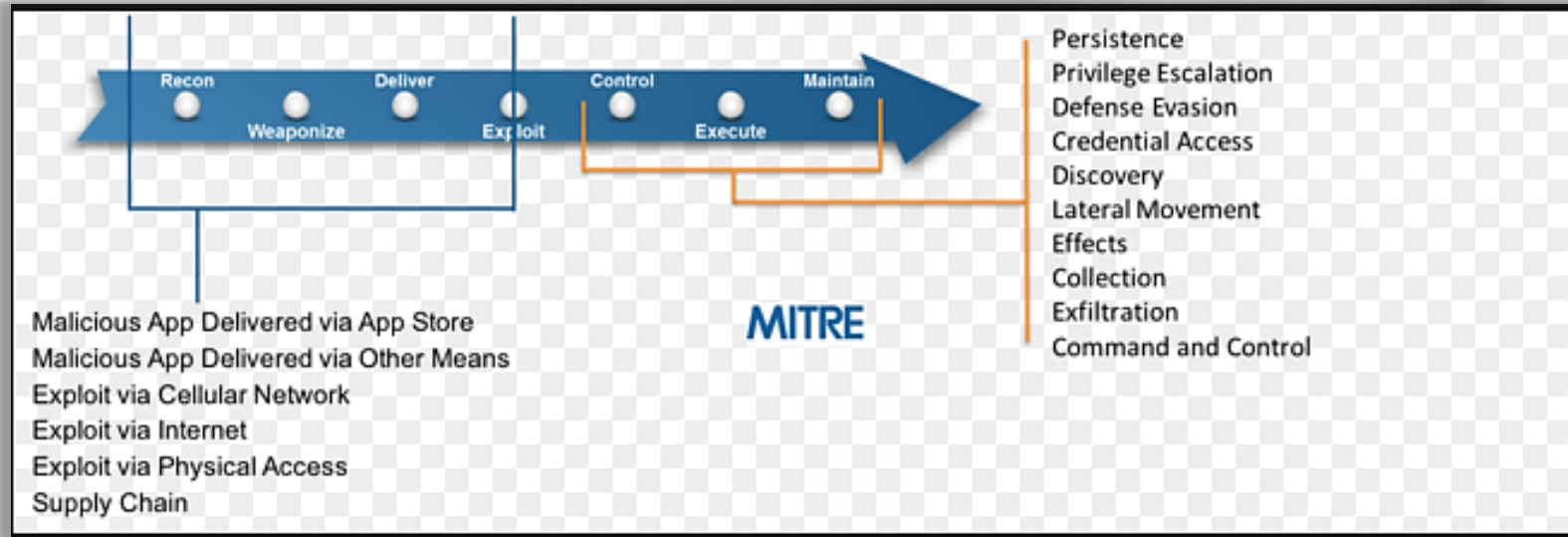
According to MITRE's [definition](#), MITRE's Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) is a curated knowledge base and model for cyber adversary behavior, reflecting the various phases of an adversary's attack lifecycle and the platforms they are known to target.

On the next slide, you'll see a visual of MITRE's representation of the attack lifecycle and where ATT&CK fits.



# 4.1 MITRE ATT&CK

## MITRE ATT&CK – Post Exploitation





# 4.1 MITRE ATT&CK

In the [matrix](#), columns define tactics, while each box represents a technique. There are over 200 techniques.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Account Access Removal
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Destruction
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Component Object Model and Distributed COM	Clipboard Data	Connection Proxy	Data Encrypted	Data Encrypted for Impact

## 4.1 MITRE ATT&CK

Each technique contains an explanation, procedure examples (often linked to threat reports), mitigation, and detection. It also includes metadata such as System requirements and Permissions Required to perform the technique.

Let's use "Kerberoasting" as an example.

# 4.1 MITRE ATT&CK

## Kerberoasting

Service principal names (SPNs) are used to uniquely identify each instance of a Windows service. To enable authentication, Kerberos requires that SPNs be associated with at least one service logon account (an account specifically tasked with running a service [1], [2] [3] [4] [5])

Adversaries possessing a valid Kerberos ticket-granting ticket (TGT) may request one or more Kerberos ticket-granting service (TGS) service tickets for any SPN from a domain controller (DC). [6] [7] Portions of these tickets may be encrypted with the RC4 algorithm, meaning the Kerberos 5 TGS-REP etype 23 hash of the service account associated with the SPN is used as the private key and is thus vulnerable to offline Brute Force attacks that may expose plaintext credentials. [7] [6] [5]

This same attack could be executed using service tickets captured from network traffic. [7]

Cracked hashes may enable Persistence, Privilege Escalation, and Lateral Movement via access to Valid Accounts. [4]

ID: T1208

Tactic: Credential Access

Platform: Windows

System Requirements: Valid domain account or the ability to sniff traffic within a domain.

Permissions Required: User

Data Sources: Windows event logs

Contributors: Praetorian

Version: 1.0

Created: 18 April 2018

Last Modified: 18 July 2019

# 4.1 MITRE ATT&CK

Under the Detection section, we see a suggestion on how to detect this attack:

## Detection

Enable Audit Kerberos Service Ticket Operations to log Kerberos TGS service ticket requests. Particularly investigate irregular patterns of activity (ex: accounts making numerous requests, Event ID 4769, within a small time frame, especially if they also request RC4 encryption [Type 0x17]). <sup>[1] [7]</sup>

## 4.1 MITRE ATT&CK

Because ATT&CK has the most public attack techniques, we can use it to learn a lot about them. ATT&CK does not define only a single signature on how to detect certain techniques, and that is also good news. We want to detect any occurrence of the technique; we want to detect based on the adversary's TTPs.

[Here](https://www.youtube.com/watch?v=tmW60vC0tHE) is a video from BSides Bristol on practically utilizing ATT&CK for hunting.

# Data Collection and Analysis



## 4.2 Data Collection and Analysis

Now that we are familiar with and can obtain information about various attacks, we need to be able to apply that knowledge against our data.

To start hunting, we need to determine what we want to hunt for, based on a hypothesis, and then perform the hunt by looking at the data we have.

```
23 # the experiment result is a dict
24 # observations is an array of Observations, in this case
25 # control is the control observation
26
27 def initialize(experiment, observations = [], control = null)
28
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = observations - [control]
33   evaluate_candidates
34
35   freeze
36
37   @experiment
38   experiment.context
39
40   @experiment_name
41   experiment.name
42
43   # ...
44
45   # ...
46
47   # ...
48
49   # ...
50
51   # ...
52
53   # ...
54
55   # ...
56
57   # ...
58
59   # ...
60
61   # ...
62
63   # ...
64
65   # ...
66
67   # ...
68
69   # ...
70
71   # ...
72
73   # ...
74
75   # ...
76
77   # ...
78
79   # ...
80
81   # ...
82
83   # ...
84
85   # ...
86
87   # ...
88
89   # ...
90
91   # ...
92
93   # ...
94
95   # ...
96
97   # ...
98
99   # ...
100
101   # ...
102
103   # ...
104
105   # ...
106
107   # ...
108
109   # ...
110
111   # ...
112
113   # ...
114
115   # ...
116
117   # ...
118
119   # ...
120
121   # ...
122
123   # ...
124
125   # ...
126
127   # ...
128
129   # ...
130
131   # ...
132
133   # ...
134
135   # ...
136
137   # ...
138
139   # ...
140
141   # ...
142
143   # ...
144
145   # ...
146
147   # ...
148
149   # ...
150
151   # ...
152
153   # ...
154
155   # ...
156
157   # ...
158
159   # ...
160
161   # ...
162
163   # ...
164
165   # ...
166
167   # ...
168
169   # ...
170
171   # ...
172
173   # ...
174
175   # ...
176
177   # ...
178
179   # ...
180
181   # ...
182
183   # ...
184
185   # ...
186
187   # ...
188
189   # ...
190
191   # ...
192
193   # ...
194
195   # ...
196
197   # ...
198
199   # ...
200
201   # ...
202
203   # ...
204
205   # ...
206
207   # ...
208
209   # ...
210
211   # ...
212
213   # ...
214
215   # ...
216
217   # ...
218
219   # ...
220
221   # ...
222
223   # ...
224
225   # ...
226
227   # ...
228
229   # ...
230
231   # ...
232
233   # ...
234
235   # ...
236
237   # ...
238
239   # ...
240
241   # ...
242
243   # ...
244
245   # ...
246
247   # ...
248
249   # ...
250
251   # ...
252
253   # ...
254
255   # ...
256
257   # ...
258
259   # ...
260
261   # ...
262
263   # ...
264
265   # ...
266
267   # ...
268
269   # ...
270
271   # ...
272
273   # ...
274
275   # ...
276
277   # ...
278
279   # ...
280
281   # ...
282
283   # ...
284
285   # ...
286
287   # ...
288
289   # ...
290
291   # ...
292
293   # ...
294
295   # ...
296
297   # ...
298
299   # ...
300
301   # ...
302
303   # ...
304
305   # ...
306
307   # ...
308
309   # ...
310
311   # ...
312
313   # ...
314
315   # ...
316
317   # ...
318
319   # ...
320
321   # ...
322
323   # ...
324
325   # ...
326
327   # ...
328
329   # ...
330
331   # ...
332
333   # ...
334
335   # ...
336
337   # ...
338
339   # ...
340
341   # ...
342
343   # ...
344
345   # ...
346
347   # ...
348
349   # ...
350
351   # ...
352
353   # ...
354
355   # ...
356
357   # ...
358
359   # ...
360
361   # ...
362
363   # ...
364
365   # ...
366
367   # ...
368
369   # ...
370
371   # ...
372
373   # ...
374
375   # ...
376
377   # ...
378
379   # ...
380
381   # ...
382
383   # ...
384
385   # ...
386
387   # ...
388
389   # ...
390
391   # ...
392
393   # ...
394
395   # ...
396
397   # ...
398
399   # ...
400
401   # ...
402
403   # ...
404
405   # ...
406
407   # ...
408
409   # ...
410
411   # ...
412
413   # ...
414
415   # ...
416
417   # ...
418
419   # ...
420
421   # ...
422
423   # ...
424
425   # ...
426
427   # ...
428
429   # ...
430
431   # ...
432
433   # ...
434
435   # ...
436
437   # ...
438
439   # ...
440
441   # ...
442
443   # ...
444
445   # ...
446
447   # ...
448
449   # ...
450
451   # ...
452
453   # ...
454
455   # ...
456
457   # ...
458
459   # ...
460
461   # ...
462
463   # ...
464
465   # ...
466
467   # ...
468
469   # ...
470
471   # ...
472
473   # ...
474
475   # ...
476
477   # ...
478
479   # ...
480
481   # ...
482
483   # ...
484
485   # ...
486
487   # ...
488
489   # ...
490
491   # ...
492
493   # ...
494
495   # ...
496
497   # ...
498
499   # ...
500
501   # ...
502
503   # ...
504
505   # ...
506
507   # ...
508
509   # ...
510
511   # ...
512
513   # ...
514
515   # ...
516
517   # ...
518
519   # ...
520
521   # ...
522
523   # ...
524
525   # ...
526
527   # ...
528
529   # ...
530
531   # ...
532
533   # ...
534
535   # ...
536
537   # ...
538
539   # ...
540
541   # ...
542
543   # ...
544
545   # ...
546
547   # ...
548
549   # ...
550
551   # ...
552
553   # ...
554
555   # ...
556
557   # ...
558
559   # ...
560
561   # ...
562
563   # ...
564
565   # ...
566
567   # ...
568
569   # ...
570
571   # ...
572
573   # ...
574
575   # ...
576
577   # ...
578
579   # ...
580
581   # ...
582
583   # ...
584
585   # ...
586
587   # ...
588
589   # ...
590
591   # ...
592
593   # ...
594
595   # ...
596
597   # ...
598
599   # ...
600
601   # ...
602
603   # ...
604
605   # ...
606
607   # ...
608
609   # ...
610
611   # ...
612
613   # ...
614
615   # ...
616
617   # ...
618
619   # ...
620
621   # ...
622
623   # ...
624
625   # ...
626
627   # ...
628
629   # ...
630
631   # ...
632
633   # ...
634
635   # ...
636
637   # ...
638
639   # ...
640
641   # ...
642
643   # ...
644
645   # ...
646
647   # ...
648
649   # ...
650
651   # ...
652
653   # ...
654
655   # ...
656
657   # ...
658
659   # ...
660
661   # ...
662
663   # ...
664
665   # ...
666
667   # ...
668
669   # ...
670
671   # ...
672
673   # ...
674
675   # ...
676
677   # ...
678
679   # ...
680
681   # ...
682
683   # ...
684
685   # ...
686
687   # ...
688
689   # ...
690
691   # ...
692
693   # ...
694
695   # ...
696
697   # ...
698
699   # ...
700
701   # ...
702
703   # ...
704
705   # ...
706
707   # ...
708
709   # ...
710
711   # ...
712
713   # ...
714
715   # ...
716
717   # ...
718
719   # ...
720
721   # ...
722
723   # ...
724
725   # ...
726
727   # ...
728
729   # ...
730
731   # ...
732
733   # ...
734
735   # ...
736
737   # ...
738
739   # ...
740
741   # ...
742
743   # ...
744
745   # ...
746
747   # ...
748
749   # ...
750
751   # ...
752
753   # ...
754
755   # ...
756
757   # ...
758
759   # ...
760
761   # ...
762
763   # ...
764
765   # ...
766
767   # ...
768
769   # ...
770
771   # ...
772
773   # ...
774
775   # ...
776
777   # ...
778
779   # ...
780
781   # ...
782
783   # ...
784
785   # ...
786
787   # ...
788
789   # ...
790
791   # ...
792
793   # ...
794
795   # ...
796
797   # ...
798
799   # ...
800
801   # ...
802
803   # ...
804
805   # ...
806
807   # ...
808
809   # ...
810
811   # ...
812
813   # ...
814
815   # ...
816
817   # ...
818
819   # ...
820
821   # ...
822
823   # ...
824
825   # ...
826
827   # ...
828
829   # ...
830
831   # ...
832
833   # ...
834
835   # ...
836
837   # ...
838
839   # ...
840
841   # ...
842
843   # ...
844
845   # ...
846
847   # ...
848
849   # ...
850
851   # ...
852
853   # ...
854
855   # ...
856
857   # ...
858
859   # ...
860
861   # ...
862
863   # ...
864
865   # ...
866
867   # ...
868
869   # ...
870
871   # ...
872
873   # ...
874
875   # ...
876
877   # ...
878
879   # ...
880
881   # ...
882
883   # ...
884
885   # ...
886
887   # ...
888
889   # ...
890
891   # ...
892
893   # ...
894
895   # ...
896
897   # ...
898
899   # ...
900
901   # ...
902
903   # ...
904
905   # ...
906
907   # ...
908
909   # ...
910
911   # ...
912
913   # ...
914
915   # ...
916
917   # ...
918
919   # ...
920
921   # ...
922
923   # ...
924
925   # ...
926
927   # ...
928
929   # ...
930
931   # ...
932
933   # ...
934
935   # ...
936
937   # ...
938
939   # ...
940
941   # ...
942
943   # ...
944
945   # ...
946
947   # ...
948
949   # ...
950
951   # ...
952
953   # ...
954
955   # ...
956
957   # ...
958
959   # ...
960
961   # ...
962
963   # ...
964
965   # ...
966
967   # ...
968
969   # ...
970
971   # ...
972
973   # ...
974
975   # ...
976
977   # ...
978
979   # ...
980
981   # ...
982
983   # ...
984
985   # ...
986
987   # ...
988
989   # ...
990
991   # ...
992
993   # ...
994
995   # ...
996
997   # ...
998
999   # ...
1000
```



## 4.2 Data Collection and Analysis

**But what is the data that we have?**

**How do we ensure it is qualified to our needs?**

**How do we transform and make it useful to us?**

## 4.2 Data Collection and Analysis

Before we hunt, we need to collect data. When collecting, we should ensure that we have a purpose based on what we want to find in that data to avoid collecting a mountain of 'noise populated' logs.

As mentioned in a previous module, the primary data types are host and network data.

```
24 def initialize_experiment, observations = []
25
26 # Create the experiment's context
27
28 @experiment = experiment
29 @observations = observations
30 @control = control
31 @candidates = observations - @control
32 evaluate_candidates
33
34 freeze
35
36 # TODO: the experiment's context
37
38 def context
39   experiment.context
40
41   experiment_name
42   experiment_name
43
44 # TODO: add the results a match between all
45
46 def matchest
47   @candidates/result.rb 1.1
```

## 4.2 Data Collection and Analysis

Once we identify what data we exactly want to collect, we need to find a method for exporting that data from the local machines to our analytics software.

Generally, the methods are:

- Push – agent on the host forwards log data as its captured
- Pull – remotely connect and collect data at the time of the connection
- A mix of the above

[illegible]

## 4.2 Data Collection and Analysis

With the exporting method in place, we should be able to conclude our hunting capability and expectations as a bare minimum based on the following:

- How much of the needed data is available for a hunt?
- How much of my environment would I cover with the available data during a hunt?
- How far back in time can I search?
- What about the quality of that data; for example, how consistent is it across the different data sources? How do we govern that?

## 4.2.1 Data Governance

**Data governance** (DG) is the overall management of the availability, usability, integrity, and security of data used in an enterprise. Businesses benefit from data governance because it ensures data is consistent and trustworthy.



## 4.2.1 Data Governance

With DG, we can define data quality that ensures, among other things:

- **Data completeness** - Is all required data available and for how long?
- **Data consistency** - Is a standard naming convention applied to data fields from different data sources?
- **Data timeliness** – Do timestamps represent the actual creation time of the events?

## 4.2.1 Data Governance

The benefit of Data Governance should be clear to us. So, how do we identify anomalies in that data?

A good first step is to understand what is “normal” in the environment, because if you do not know what is currently there, it is much harder to identify what should not be.



## 4.2.1 Data Governance

Getting familiar with “normal” can be achieved by the baselining of regular activity that is expected on a golden image of an endpoint and in the network traffic. This includes:

- Running Processes
- User logons -> where, when, and what type of login
- Network connections
- Services and scheduled tasks
- Software that is allowed to execute

```
21 # def init_context
22 #
23 # @param context - the experiment's context
24 # @param observations - an array of Observations, or None
25 # @param control - the control Observation
26 #
27 def initialize(experiment, observations = [], control = null)
28   @experiment = experiment
29   @observations = observations
30   @control = control
31   @candidates = observations + [control]
32   evaluate_candidates
33
34   # TODO: the experiment's context
35   def context
36     experiment.context
37   end
38
39   # TODO: the name of the experiment
40   def experiment_name
41     experiment.name
42   end
43
44   # TODO: test the results a match between all
45   def matcher
46     # ...
47   end
48   @score = result[0]
49 end
```

## 4.2.2 Data Analysis

And that brings us finally to data analysis, that is how do we get the useful parts of the collected data?

An analysis is usually performed on a SIEM system, such as ELK/HELK, Splunk, Graylog, and many others.

```
def initialize(experiment, observations = [], control = null)
  @experiment = experiment
  @observations = observations
  @control = control
  @candidates = observations - @control
  evaluate_candidates

  freeze

  experiment.context
  experiment.name
  experiment.name
end

def match?
  # ...
end

def match?
  # ...
end
```

## 4.2.2 Data Analysis

Analyzing the data means that we are manipulating it by:

- Searching it
- Aggregating it
- Filtering it
- Joining data together

What is searching and aggregating?

```
21 # @param: control - the control observation
22 # @param: observations - an array of observations, in ascending order
23 # @param: candidates - the control observation
24 # @param: candidates - the control observation
25
26 def initialize(experiment, observations = [], control = null)
27   @experiment = experiment
28   @observations = observations
29   @control = control
30   @candidates = observations - [control]
31   evaluate_candidates
32
33   freeze
34 end
35
36 # @param: the experiment's context
37 def context
38   @experiment.context
39 end
40
41 # @param: the name of the experiment
42 def experiment_name
43   @experiment.name
44 end
45
46 # @param: the result a match between two candidates
47 def match
48   @match
49 end
50
51 # @param: result - 1
```

## 4.2.2 Data Analysis

Searching data gives us the ability to find answers to our questions. In practice, the first searches we perform will most likely not meet our expectations. It will have to be modified in one way or another (e.g., reduce the timespan or exclude a user, etc.) for us to obtain what we want.





## 4.2.2 Data Analysis

The searching queries will be specific to the tool that you are using for analysis, but in general, the tool shouldn't matter, as their syntax is similar. Most (if not all) support Boolean operators, comparison operators, direct key-value comparison, wildcard matching, and many more.

The more searches you do, the better you become at it. Practice is crucial!

## 4.2.2 Data Analysis

Aggregation, on the other hand, is grouping values within the same field together. In many cases, aggregated data is sorted by either the amount of the most common or least common data in that field – it gives us a different perspective in the data. We can see an oversimplified example in the table below:

Process name	Occurrences
Svchost.exe	13
Winword.exe	9
Calc.exe	1



## 4.2.2 Data Analysis

The aggregated data and terms displayed can be extended to match your exact criteria. For example, if you are looking at executed PowerShell commands, you may want to display:

- The command itself
- Who executed it
- On which host was it executed
- How many times it was executed

```
24 # Get the results of the experiment
25 # observations - an array of observations
26 # control - the control observation
27
28 def initialize(experiment, observations = [], control = null)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = observations + [control]
33   evaluate_candidates
34
35   freeze
36 end
37
38 # Returns the experiment's context
39 def context
40   @experiment.context
41 end
42
43 # Returns the name of the experiment
44 def experiment_name
45   @experiment.name
46 end
47
48 # Returns the results a match between the
49 def matches
50   @experiment.result
51 end
```

## 4.2.2 Data Analysis

In many occasions, we will do a search, which will be aggregated afterward. One example is looking at the least common occurrence of a search, which returned a lot of results. By looking at the aggregated data, we attempt to identify an outlier/odd occurrences.

Naturally, through aggregation, we are performing statistical analysis (count, sum, average of, frequency analysis, etc.) on the data, which gives us a different perspective into it that can outline the abnormal.

## 4.2.2 Data Analysis

As hunters, you should expect that some of the results may contain only partial answers on what you are looking for. So how do you continue?

Remember how we discussed multiple data sources? The answer is by switching to other data sources (when possible).

```
24 # @param context - the Experiment Data Context
25 # @param observations - an array of Observations, or Observations
26 # @param control - the control Observation
27
28 def initialize(experiment, observations = [], control = null)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = observations - [control]
33   evaluate_candidates
34
35   freeze
36
37   @context = context
38   experiment.context
39
40   @experiment_name =
41     experiment.name
42   @experiment_name
43
44   # @param result - the result a match between an observation
45   def match?
46     !!@candidates[result.id]
47   end
48
49   def match?
50     !!@candidates[result.id]
51   end
```



## 4.2.2 Data Analysis

A great example of this is an anomaly that someone attempted a connection to a suspicious IP address.

You can correlate this with PCAP/NetFlow data to review exactly what the communication contained and to verify if it is a true malicious event, or disregard it as a false positive.

# Hunting Hypothesis and Methodology



## 4.3 Hunting Hypothesis and Methodology

Every hunt begins by defining a hypothesis. It consists of the following:

- Identify the specific behavior we want to hunt for
- Understand the attack technique behind it
- Identify what data (and from which source) we need to detect it

To achieve this, we'll use a 5-step process.

## 4.3 Hunting Hypothesis and Methodology

## The 5 steps are:

1. Pick a tactic and technique
2. Identify associated procedure(s)
3. Perform an attack simulation
4. Identify evidence to collect
5. Set scope

```

10:         return result
11:
12:     def __init__(self, experiment, observations, control):
13:         """
14:         Constructor for the experiment
15:         """
16:         self.experiment = experiment
17:         self.observations = observations
18:         self.control = control
19:
20:     def __str__(self):
21:         """
22:         String representation of the experiment
23:         """
24:         return "Experiment: %s, Observations: %s, Control: %s" % (self.experiment, self.observations, self.control)
25:
26:     def __repr__(self):
27:         """
28:         String representation of the experiment
29:         """
30:         return "Experiment: %s, Observations: %s, Control: %s" % (self.experiment, self.observations, self.control)
31:
32:     def __getitem__(self, index):
33:         """
34:         Get the observation at the given index
35:         """
36:         return self.observations[index]
37:
38:     def __setitem__(self, index, value):
39:         """
40:         Set the observation at the given index
41:         """
42:         self.observations[index] = value
43:
44:     def __len__(self):
45:         """
46:         Length of the experiment
47:         """
48:         return len(self.observations)
49:
50:     def __iter__(self):
51:         """
52:         Iterate over the experiment
53:         """
54:         return iter(self.observations)
55:
56:     def __contains__(self, value):
57:         """
58:         Check if the value is in the experiment
59:         """
60:         return value in self.observations
61:
62:     def __del__(self):
63:         """
64:         Delete the experiment
65:         """
66:         pass
67:
68:     def __copy__(self):
69:         """
70:         Copy the experiment
71:         """
72:         return Experiment(self.experiment, self.observations, self.control)
73:
74:     def __deepcopy__(self, memo):
75:         """
76:         Deep copy the experiment
77:         """
78:         return Experiment(self.experiment, self.observations, self.control)
79:
80:     def __eq__(self, other):
81:         """
82:         Check if the experiment is equal to another
83:         """
84:         return self.experiment == other.experiment and self.observations == other.observations and self.control == other.control
85:
86:     def __neq__(self, other):
87:         """
88:         Check if the experiment is not equal to another
89:         """
90:         return not self.__eq__(other)
91:
92:     def __hash__(self):
93:         """
94:         Hash the experiment
95:         """
96:         return hash(self.experiment) + hash(self.observations) + hash(self.control)
97:
98:     def __cmp__(self, other):
99:         """
100:         Compare the experiment
101:         """
102:         return cmp(self.experiment, other.experiment)
103:
104:     def __lt__(self, other):
105:         """
106:         Check if the experiment is less than another
107:         """
108:         return self.experiment < other.experiment
109:
110:     def __gt__(self, other):
111:         """
112:         Check if the experiment is greater than another
113:         """
114:         return self.experiment > other.experiment
115:
116:     def __le__(self, other):
117:         """
118:         Check if the experiment is less than or equal to another
119:         """
120:         return self.experiment <= other.experiment
121:
122:     def __ge__(self, other):
123:         """
124:         Check if the experiment is greater than or equal to another
125:         """
126:         return self.experiment >= other.experiment
127:
128:     def __add__(self, other):
129:         """
130:         Add the experiment to another
131:         """
132:         return Experiment(self.experiment, self.observations + other.observations, self.control)
133:
134:     def __sub__(self, other):
135:         """
136:         Subtract the experiment from another
137:         """
138:         return Experiment(self.experiment, self.observations - other.observations, self.control)
139:
140:     def __mul__(self, other):
141:         """
142:         Multiply the experiment by another
143:         """
144:         return Experiment(self.experiment, self.observations * other.observations, self.control)
145:
146:     def __div__(self, other):
147:         """
148:         Divide the experiment by another
149:         """
150:         return Experiment(self.experiment, self.observations / other.observations, self.control)
151:
152:     def __mod__(self, other):
153:         """
154:         Modulo the experiment by another
155:         """
156:         return Experiment(self.experiment, self.observations % other.observations, self.control)
157:
158:     def __divmod__(self, other):
159:         """
160:         Divmod the experiment by another
161:         """
162:         return (Experiment(self.experiment, self.observations / other.observations, self.control), Experiment(self.experiment, self.observations % other.observations, self.control))
163:
164:     def __pow__(self, other):
165:         """
166:         Power the experiment by another
167:         """
168:         return Experiment(self.experiment, self.observations ** other.observations, self.control)
169:
170:     def __radd__(self, other):
171:         """
172:         Right add the experiment to another
173:         """
174:         return Experiment(self.experiment, other.observations + self.observations, self.control)
175:
176:     def __rsub__(self, other):
177:         """
178:         Right subtract the experiment from another
179:         """
180:         return Experiment(self.experiment, other.observations - self.observations, self.control)
181:
182:     def __rmul__(self, other):
183:         """
184:         Right multiply the experiment by another
185:         """
186:         return Experiment(self.experiment, other.observations * self.observations, self.control)
187:
188:     def __rdiv__(self, other):
189:         """
190:         Right divide the experiment by another
191:         """
192:         return Experiment(self.experiment, other.observations / self.observations, self.control)
193:
194:     def __rmod__(self, other):
195:         """
196:         Right modulo the experiment by another
197:         """
198:         return Experiment(self.experiment, other.observations % self.observations, self.control)
199:
200:     def __rdivmod__(self, other):
201:         """
202:         Right divmod the experiment by another
203:         """
204:         return (Experiment(self.experiment, other.observations / self.observations, self.control), Experiment(self.experiment, other.observations % self.observations, self.control))
205:
206:     def __rpow__(self, other):
207:         """
208:         Right power the experiment by another
209:         """
210:         return Experiment(self.experiment, other.observations ** self.observations, self.control)
211:
212:     def __iadd__(self, other):
213:         """
214:         Inplace add the experiment to another
215:         """
216:         self.observations = self.observations + other.observations
217:         return self
218:
219:     def __isub__(self, other):
220:         """
221:         Inplace subtract the experiment from another
222:         """
223:         self.observations = self.observations - other.observations
224:         return self
225:
226:     def __imul__(self, other):
227:         """
228:         Inplace multiply the experiment by another
229:         """
230:         self.observations = self.observations * other.observations
231:         return self
232:
233:     def __idiv__(self, other):
234:         """
235:         Inplace divide the experiment by another
236:         """
237:         self.observations = self.observations / other.observations
238:         return self
239:
240:     def __imod__(self, other):
241:         """
242:         Inplace modulo the experiment by another
243:         """
244:         self.observations = self.observations % other.observations
245:         return self
246:
247:     def __iaddi__(self, other):
248:         """
249:         Inplace add the experiment to another
250:         """
251:         self.observations = self.observations + other.observations
252:         return self
253:
254:     def __isubi__(self, other):
255:         """
256:         Inplace subtract the experiment from another
257:         """
258:         self.observations = self.observations - other.observations
259:         return self
260:
261:     def __imuli__(self, other):
262:         """
263:         Inplace multiply the experiment by another
264:         """
265:         self.observations = self.observations * other.observations
266:         return self
267:
268:     def __idivi__(self, other):
269:         """
270:         Inplace divide the experiment by another
271:         """
272:         self.observations = self.observations / other.observations
273:         return self
274:
275:     def __imodi__(self, other):
276:         """
277:         Inplace modulo the experiment by another
278:         """
279:         self.observations = self.observations % other.observations
280:         return self
281:
282:     def __iaddi__(self, other):
283:         """
284:         Inplace add the experiment to another
285:         """
286:         self.observations = self.observations + other.observations
287:         return self
288:
289:     def __isubi__(self, other):
290:         """
291:         Inplace subtract the experiment from another
292:         """
293:         self.observations = self.observations - other.observations
294:         return self
295:
296:     def __imuli__(self, other):
297:         """
298:         Inplace multiply the experiment by another
299:         """
300:         self.observations = self.observations * other.observations
301:         return self
302:
303:     def __idivi__(self, other):
304:         """
305:         Inplace divide the experiment by another
306:         """
307:         self.observations = self.observations / other.observations
308:         return self
309:
310:     def __imodi__(self, other):
311:         """
312:         Inplace modulo the experiment by another
313:         """
314:         self.observations = self.observations % other.observations
315:         return self
316:
317:     def __iaddi__(self, other):
318:         """
319:         Inplace add the experiment to another
320:         """
321:         self.observations = self.observations + other.observations
322:         return self
323:
324:     def __isubi__(self, other):
325:         """
326:         Inplace subtract the experiment from another
327:         """
328:         self.observations = self.observations - other.observations
329:         return self
330:
331:     def __imuli__(self, other):
332:         """
333:         Inplace multiply the experiment by another
334:         """
335:         self.observations = self.observations * other.observations
336:         return self
337:
338:     def __idivi__(self, other):
339:         """
340:         Inplace divide the experiment by another
341:         """
342:         self.observations = self.observations / other.observations
343:         return self
344:
345:     def __imodi__(self, other):
346:         """
347:         Inplace modulo the experiment by another
348:         """
349:         self.observations = self.observations % other.observations
350:         return self
351:
352:     def __iaddi__(self, other):
353:         """
354:         Inplace add the experiment to another
355:         """
356:         self.observations = self.observations + other.observations
357:         return self
358:
359:     def __isubi__(self, other):
360:         """
361:         Inplace subtract the experiment from another
362:         """
363:         self.observations = self.observations - other.observations
364:         return self
365:
366:     def __imuli__(self, other):
367:         """
368:         Inplace multiply the experiment by another
369:         """
370:         self.observations = self.observations * other.observations
371:         return self
372:
373:     def __idivi__(self, other):
374:         """
375:         Inplace divide the experiment by another
376:         """
377:         self.observations = self.observations / other.observations
378:         return self
379:
380:     def __imodi__(self, other):
381:         """
382:         Inplace modulo the experiment by another
383:         """
384:         self.observations = self.observations % other.observations
385:         return self
386:
387:     def __iaddi__(self, other):
388:         """
389:         Inplace add the experiment to another
390:         """
391:         self.observations = self.observations + other.observations
392:         return self
393:
394:     def __isubi__(self, other):
395:         """
396:         Inplace subtract the experiment from another
397:         """
398:         self.observations = self.observations - other.observations
399:         return self
400:
401:     def __imuli__(self, other):
402:        
```



## 4.3.1 Pick a Tactic and Technique

### Step #1 Pick a tactic and technique

In this step, we review attack sources and pick a technique. As mentioned earlier, we will utilize the MITRE ATT&CK framework in this course, so that is our source.

As an example, let's pick technique [T1502](https://attack.mitre.org/techniques/T1502/) - Parent PID Spoofing under "Privilege Escalation".

## 4.3.1 Pick a Tactic and Technique

### Parent PID Spoofing:

“Adversaries may spoof the parent process identifier (PPID) of a new process to evade process-monitoring defenses or to elevate privileges. ... Adversaries may abuse these mechanisms to evade defenses, such as those blocking processes spawning directly from Office documents and analysis targeting unusual / potentially malicious parent-child process relationships ... Explicitly assigning the PPID may also enable Privilege Escalation ...”

## 4.3.2 Identify Associated Procedure(s)

### Step #2 Identify associated procedure(s)

ATT&CK will also present us with the procedure for the technique. In our case, it has the following:

#### Procedure Examples

Name	Description
Cobalt Strike	Cobalt Strike can spawn processes with alternate PIDs. <sup>[6]</sup>

## 4.3.2 Identify Associated Procedure(s)

We recommend additional research in this phase, as reports and blog posts often have developed tools and different implementations of the technique, which may not be present in ATT&CK. Do your own research to really understand the procedures, including their prerequisites, requirements, and outcome if successfully executed.

The following posts provide additional context to our example of PPID Spoofing, which you can check out [here](#) and [here](#).

## 4.3.2 Identify Associated Procedure(s)

From the blog posts, we can tell:

- **Prerequisite** – the attacker must have already compromised the box and can execute commands.
- **Requirements** – process path, path to injected assembly, and fake parent process name.
- **Outcome** – new processes are started.

```
34 def __init__(self, process_path, assembly_path, parent_name):
35     self.process_path = process_path
36     self.assembly_path = assembly_path
37     self.parent_name = parent_name
38     self.observations = []
39     self.control = None
40     self.candidates = []
41     self.create_conditions()
42
43     # Initialize the experiment
44     self.initialize(experiment, observations = [], control = None)
45
46     # Initialize the experiment
47     @experiment
48     @observations
49     @control
50     @candidates
51     @create_conditions
52
53     # Initialize the experiment's context
54     def context:
55         experiment.context
56
57     # Initialize the name of the experiment
58     def experiment_name:
59         experiment.name
60
61     # Initialize the result of the experiment
62     def result:
63         experiment.result
64
65     # Initialize the result of the experiment
66     def result:
67         experiment.result
```

## 4.3.3 Perform an Attack Simulation

### Step #3 Perform an attack simulation

Although not always necessary (based on what you have gathered previously), this step replicates the procedure(s) of the technique. During the replication process, you can observe what data and logs are generated, and in the next step, build your detection based on the identified behaviors.

It can be very time consuming, as it requires you to have the systems and monitoring capabilities equivalent to the ones in your environment.



## 4.3.4 Identify Evidence to Collect

### Step #4 Identify evidence to collect

Once executed in our controlled environment, we should start investigating places that may contain artifacts of interest; this could be on disk, in-memory, network traffic, registry, etc.

The goal of this step is to identify behaviors we can use to detect this activity in the future.



## 4.3.4 Identify Evidence to Collect

**Some of the things to look for are:**

- Deviations from baselines
- Attempts to appear normal (e.g., svchost.exe running from user directory)
- Unexpected encryption
- Odd frequency of occurrence (e.g., too many connections or user activity at an abnormal time)

## 4.3.4 Identify Evidence to Collect

Other than identifying detection behaviors, you will likely find some false positives, which may be unique to your environment. Having the ability to filter these out at an early stage is beneficial during the actual hunt.

To avoid confusion at this point (due to concepts not yet explained), we will return to the PPID Spoofing detection at a later stage during a lab.

## 4.3.5 Set Scope

### Step #5 Set Scope

Finally, now that we know everything about the technique and its procedure(s) and having identified behaviors after the simulation, we are almost ready to begin proving/disproving our hypothesis of whether a specific activity has occurred or not.

The last thing left is to define the hunting scope.

## 4.3.5 Set Scope

The main things we should define in the scope are:

- Hunt Duration
- What data and where to collect from

We recommend that the duration of a hunt is at least a week.

```
20 with_reader(reader)
21
22 # @context: Scope of the hunt
23
24 # @experiment: the experiment this result is for
25 # @observations: an array of Observations, in @context
26 # @control: the control Observation
27
28 def initialize(experiment, observations = [], control = null)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = observations - [control]
33   evaluate_candidates
34
35   freeze
36 end
37
38 # @RANDO: the experiment's context
39 def context
40   @context
41 end
42
43 # @NAME: the name of the experiment
44 def experiment_name
45   @experiment.name
46 end
47
48 # @MATCH: the result a match between all
49 def match?
50   @match
51 end
52
53 # @SCIENTIST: the result a match between all
54 def scientist(result)
55   1
56 end
```



## 4.3.5 Set Scope

As for the scope, there may be other external factors that do not allow us to capture all data from all devices. This could be due to network bandwidth or analysis capability in the defined timeframe (too many logs to analyze).

In some hunts, you may want to focus only on Business-Critical Systems, while on others only on certain data sources.



## 4.3.5 Set Scope

You should outline the assumptions and limitations set in the scope, which may have an impact on future decisions (about tools, architecture, future hunts, etc.).

For our example, of PPID spoofing, we define it as:

- Time – 1 week
- Collection – All Windows devices sending Event Tracing for Windows (ETW) logs. We will discuss ETW in a later module; for now, think of it as a certain type of Windows logs.

## 4.3.5 Set Scope

Remember to take notes during the hunts of any activity that was unsolved, as these may be used at a later stage or during another hunt.

Use a tool that you are comfortable with, and you will want to ensure a consistent structure to help you identify notes from previous hunts.





# Hunting Metrics



## 4.4 Hunting Metrics

Hunting metrics are a way for us to track our progress and report back to management about our hunting expeditions.

But how do we define metrics that actually make sense?  
Are we good at hunting only if we find malicious activity?  
No, simply because there may be nothing to find.

## 4.4 Hunting Metrics

You may be thinking that ongoing simulated malicious activity may be good to show that your hunting expeditions can detect that activity, but simulating malicious activity in production environments is not advised.

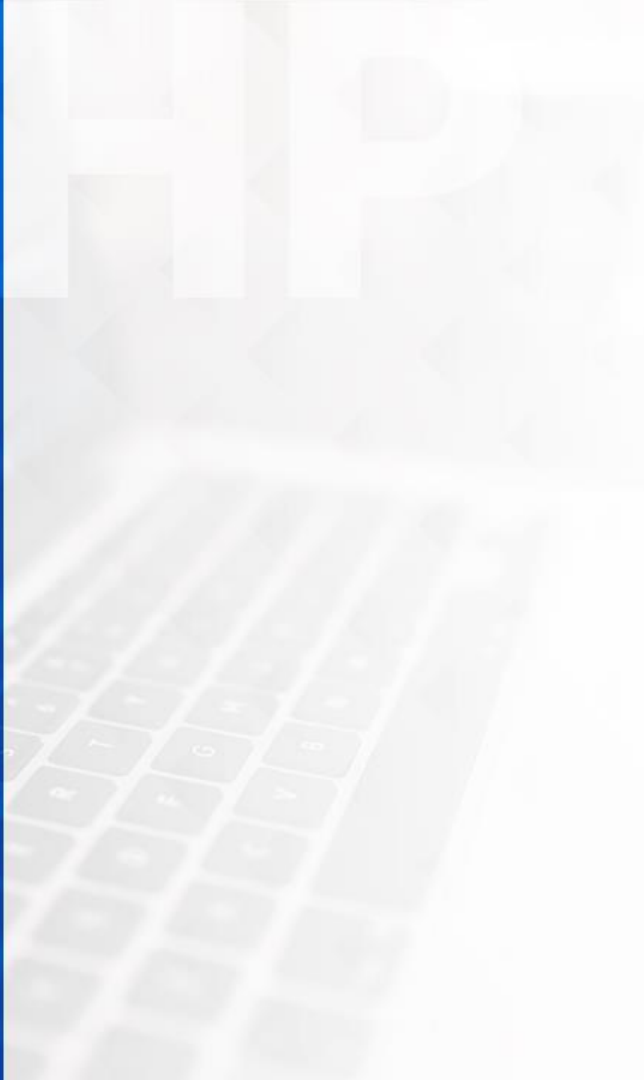
Instead, coordination with the penetration testing team can be formed to test the detection capabilities.

## 4.4 Hunting Metrics

Some things that we as hunters have control over are great to show the work we perform, such as:

- How often we hunt
- Technique coverage of ATT&CK
- Procedure coverage of certain groups relevant for your type of business
- Hunting coverage of the network
- Historic logging capability that facilitates our hunts (how far back can we go)

```
21 # @param: context - the experiment's context
22 # @param: observations - an array of Observations, or Observations
23 # @param: control - the control observation
24
25 def initialize(experiment, observations = [], control = null)
26   @experiment = experiment
27   @observations = observations
28   @control = control
29
30   freeze
31 end
32
33 # @param: the experiment's context
34 def context
35   @experiment.context
36 end
37
38 # @param: the name of the experiment
39 def experiment_name
40   @experiment.name
41 end
42
43 # @param: result - the result of the experiment
44 def result
45   @result
46 end
47
48 # @param: result - the result of the experiment
49 def result
50   @result
51 end
```



# References



# References

## [Report Template for Threat Intelligence and Incident Response](https://zeltser.com/cyber-threat-intel-and-ir-report-template/)

<https://zeltser.com/cyber-threat-intel-and-ir-report-template/>

## [MITRE ATT&CK](https://attack.mitre.org/)

<https://attack.mitre.org/>

## [MITRE ATT&CK™: Design and Philosophy](https://www.mitre.org/sites/default/files/publications/pr-18-0944-11-mitre-attack-design-and-philosophy.pdf)

<https://www.mitre.org/sites/default/files/publications/pr-18-0944-11-mitre-attack-design-and-philosophy.pdf>

## [MITRE ATT&CK Framework For Threat Hunting](https://www.youtube.com/watch?v=tmW60vC0tHE)

<https://www.youtube.com/watch?v=tmW60vC0tHE>

# References

## [What is data governance and why does it matter?](https://searchdatamanagement.techtarget.com/definition/data-governance)

<https://searchdatamanagement.techtarget.com/definition/data-governance>

## [Parent PID Spoofing](https://attack.mitre.org/techniques/T1502/)

<https://attack.mitre.org/techniques/T1502/>

## [Parent PID Spoofing](https://medium.com/@r3n_hat/parent-pid-spoofing-b0b17317168e)

[https://medium.com/@r3n\\_hat/parent-pid-spoofing-b0b17317168e](https://medium.com/@r3n_hat/parent-pid-spoofing-b0b17317168e)

## [Detecting Parent PID Spoofing](https://blog.f-secure.com/detecting-parent-pid-spoofing/)

<https://blog.f-secure.com/detecting-parent-pid-spoofing/>







## [T1208 - Kerberoasting](https://attack.mitre.org/techniques/T1208/)

<https://attack.mitre.org/techniques/T1208/>

# References

