Streamlining the Threat Detection Development Lifecycle

with SnapAttack

```
function on( elem, types, selector, data, fn, one ) {
  var origFn, type;

  // Types can be a map of types/handlers
  if ( typeof types === "object" ) {

    // ( types-Object, selector, data )
    if ( typeof selector !== "string" ) {

      // ( types-Object, data )
      data = data || selector;
      selector = undefined;
    }
    for ( type in types ) {
      on( elem, type, selector, data, types[ type ], one );
    }
    return elem;
  }

  if ( data == null && fn == null ) {
```

Table of Contents

Streamline Threat Detection

Threat detection teams are either restrained by the limited capabilities of their systems (that were never designed to detect a high volume of complex threats in the first place) or the manual processes their organization has in place (which are nowhere near fast or efficient enough to keep up with the amount of threats they're encountering). And as threats become more complex and constant, SOC teams desperately need a solution that can keep up.

Codified, repeatable workflows allow engineers to follow a streamlined process time and time again as projects arise and needs change. Other industries like software engineering have implemented structured frameworks to adjust to the growing need for solutions that are faster, stronger, and more resource-efficient.

In the threat detection and response space, detection engineering has emerged as a mechanism for security teams to adapt to and resolve the incoming barrage of threats.

What is Detection Engineering?

Detection engineering arose as a flexible, powerful, and repeatable solution to produce high quality detections and alerts. It applies a systems-thinking approach to detection development, providing automation and scale.

Detection engineering has many benefits, including:

- Reduced mean-time-to-detect (MTTD)
- Customizable and adaptable to your environment
- Visibility and confidence in coverage across the attack surface
- Integration of technology and collaboration between teams
- Structured process can yield increased fidelity / fewer false positives
- Operational insight to drive smarter decisions and prioritization

However, detection engineering has not been standardized across the cybersecurity industry just yet, leaving individual companies to develop their own methods. Without a fortified process in place to build detections, over time, security teams will suffer from alert fatigue, low-fidelity detections, and an influx of undetected attacks.

In addition to the lack of standardization across the cybersecurity industry, each organization has its own **challenges to detection engineering**, including:

- Many log analytics and detection systems were **never intended to be threat detection** and response tools
- SIEM solutions require customization, time, and effort to augment and are **unable to scale** across large or decentralized environments
- **Teams are under-resourced** in people and budget, meaning they can't put the time and money into building their own infrastructure to make them effective
- There are several **disparate, complex data sources** to pull from due to legacy tooling, crowdsourced threat intelligence, and a **lack of structure** across the industry
- Manual processes often mean an **absence of automation**, leaving teams to triage the endless stream of alerts by hand

These challenges add up, creating barriers that limit team collaboration, tool utilization, and ultimately lead to a high volume of false positives and negatives and insufficient coverage.

How Can Detection Engineering Improve?

Past approaches to detection engineering, though effective on a small scale, quickly deteriorate into an unstructured, muddled, complicated, wild goose chase.

So…what's the solution?

Threat actors operate under advanced processes with highly sophisticated technology, operationalizing attacks through models such as ransomware-as-a-service (RaaS). To combat them, threat hunters and detection engineers must take a page out of the hacker playbook and follow a structured, repeatable workflow.

Enter: The Detection Development Lifecycle – A Comprehensive Solution

Software engineers employ the software development lifecycle (SDLC) to codify and streamline their software engineering process. By instituting a repeatable process and translatable output through the SDLC, software engineers can develop solutions at a higher quality, lower cost, and in a shorter amount of time.

Applying this school of thought to detection development **gave rise to the detection development lifecycle** as both processes follow a similar flow, equipping code and a documented workflow to maximize efficiency in the detection engineering journey.

Software Development Lifecycle		Detection Development Lifecycle	
1.	Gather and analyze requirements	1.	Research
2.	Design	2.	Design
3.	Implement / code	3.	Write
4.	Test	4.	Validate
5.	Deploy	5.	Deploy
6.	Maintain / monitor	6.	Measure

The detection development lifecycle arms detection engineers with a mature, repeatable framework that streamlines the development and maintenance of detections. Each time a new threat is introduced, less effort is required of teams to move through the otherwise scattered path from ideation to deployment. Viewing detection engineering from a coding perspective **illuminates patterns, priorities, and effective tactics** as detections are written. Additionally, validating detections in this lens creates not only more robust detections, but a more robust threat detection program altogether by **applying one detection to multiple threats**. Under the detection development lifecycle, engineers and their teams benefit from higher quality detections, a greater ability to scale teams and output, and more robust coverage that allows them to prioritize incoming threats.

The key to the success of the detection development lifecycle lies in two core components: (1) a well-defined process interlaced with (2) adaptable, codified tactics known as detection-as-code. Together, these elements remove the barriers currently inhibiting detection engineers from creating quality detections and manifest into a flexible yet repeatable structure.

> Structured Workflow

Many organizations lack the infrastructure and workflow to implement a streamlined process to follow each time they need a new detection. Without strong, reliable processes in place, threat detection teams exhaust themselves and all resources just trying to get a detection out the door for each threat they face. Not only is this inefficient, but it can also create lower-quality detections as engineers scramble to protect their organization on an expedited timeline with limited resources. Developing a strategy for detection engineering can give threat hunters and detection engineers a procedure to follow, lessening the strain on them to figure out a new solution each time a vulnerability is introduced and empowering individuals and teams to collaborate more effectively.

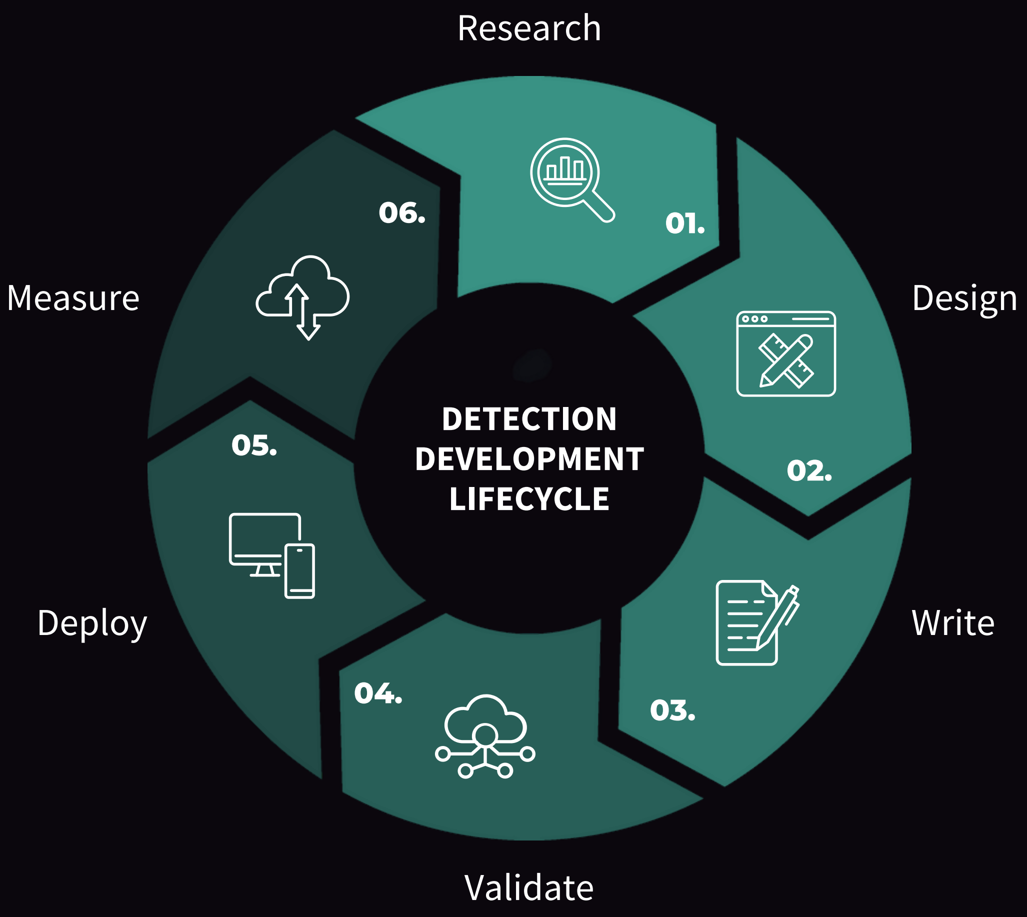
> Detection-as-Code

The complex threat environment spanning security operations today calls for a disciplined approach to developing, deploying, and managing detections. Sharing IOCs is fairly common and simple, but sharing more advanced information such as TTPs is considerably more complicated.

Security teams are burdened with a hefty duplication of effort when writing and rewriting detections in various vendor and query languages, frequently needing to apply detections to several different environments.

Detection-as-code democratizes detections by allowing for the flexibility of detection content from one threat to another, abstracting the syntax needed by a given platform. Codifying detections brought a solution to the table that made detections shareable and repeatable, standardizing the format in which they were built around behavior rather than specific threats. When vendor languages are no longer a challenge obstructing development, **engineers can pivot their focus to more critical elements** like pure threat intelligence, methodologies, and tactics regardless of the language in which they're writing.

Detection Development Lifecycle



Research

Before beginning to develop a detection, teams must ask several questions and consult many resources to determine needs and priorities.

1. What is the attack?

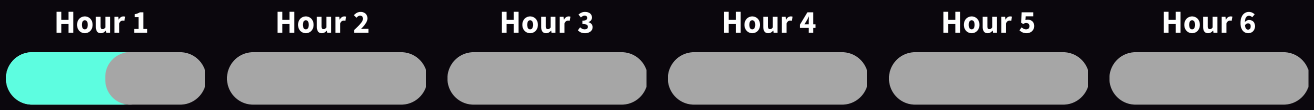
Detection engineering starts with what appears to be a simple question, but getting to the bottom of that question is a different story.

Pinning a threat down is about as straightforward as finding a needle in a haystack. With a vast array of resources, threat hunters turn to online forums and resources like:

- Hacking news sites
- Social media
- Reddit
- Twitter
- Blog posts

The aim of this step is to learn anything and everything they can about each hacker they research. Other methods to research the attacker are reviewing premium intelligence feeds or SME intelligence services (which many organizations don't have) and asking questions of peers (who aren't always willing to share).

There is no single source of truth for threat intelligence at this stage: threat hunters must discern within this plethora of information what's accurate, what's up-to-date, and what's useful in developing a detection for their organization.

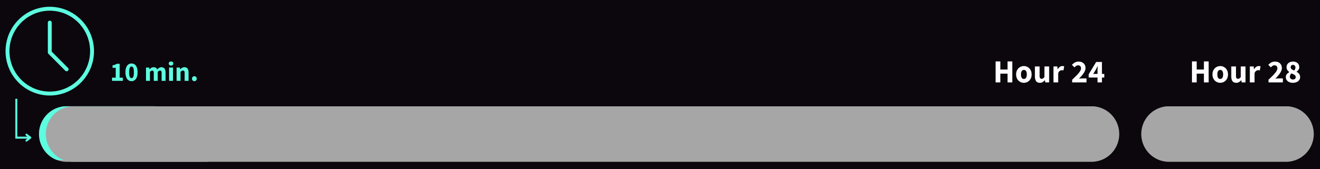


WITH SnapAttack	**WITHOUT SnapAttack**
~30 minutes Take threat intelligence from anywhere: write your own, use SnapAttack's, or bring it from another source.	**6 hours** Do all of your own research with no guidance, direction, or knowledge if you're looking in the right places.

2. Is this relevant to us?

Before moving forward with the development of a detection, analysts have to assess their findings to determine whether or not the threat at hand is relevant or dangerous to their environment. Not all threats will apply to all organizations, so threat hunters need visibility and confidence in the current standing of their coverage.

At this stage, techniques like threat modeling and further vulnerability research are useful in determining how applicable a threat is to an organization.



WITH SnapAttack	WITHOUT SnapAttack
10 minutes Determine existing defenses and model threats against the MITRE ATT&CK Matrix, all from the SnapAttack platform.	**28 hours** • Threat modeling • Vulnerability research • Asset management review

3. Which TTPs are they using and which should I prioritize?

Past frameworks for threat detection have honed in on indicators of compromise (IOCs) such as IP addresses, domain names, and hash values – but as adversaries have become more advanced, evasive, and tactical, these values are no longer enough as they are easily changeable.

Behavioral tactics, techniques, and procedures (TTPs) are exceedingly more valuable because attackers must reconfigure their tooling and approach to evade notice. This gives threat hunters the chance to catch them earlier in the kill chain before they're able to move laterally across the threat environment. Therefore, detection engineers and their teams should focus their research on relevant TTPs to ultimately build higher-fidelity detections.

5-10 min.
Hour 24
Hour 32

WITH SnapAttack	WITHOUT SnapAttack
5-10 minutes • Review SnapAttack Attack Sessions for TTPs catalogued in-app or spin up the sandbox for adversary emulation • Map coverage against the MITRE ATT&CK Matrix • Determine confidence level of existing detections • If no existing attack session, spin up sandbox and emulate the adversary	**24-32 hours** • Threat modeling • Adversary emulation • Prioritize

Design

Once sufficient research has been conducted, engineers must reflect their findings through the design of the detection.

4. How do I write a detection that will work here? What controls will help me identify this activity?

Looking back to the inspiration for the DDLC, a secure SDLC often involves incorporating security testing and other tasks into an already-existing development process. Examples include doing an architecture risk analysis at the design stage of the SDLC and establishing security requirements alongside functional requirements. This method is intended to produce software with fewer design flaws, producing less vulnerabilities and therefore less exploit opportunities.

Once in the design stage, engineers begin looking into the vulnerability further to build out the infrastructure for adversary emulation if their organization has the capability to do so. Like the SDLC, aligning priorities, capabilities, and needs sets the foundation for a strong detection. The exploit or TTP must be tested in a realistic environment so the engineer can review logs and determine the detection strategy. Continuous testing is critical as the logic is built out so the engineer can streamline the development process and create a deeply robust detection.

10-15 min.
Hour 21
Hour 29

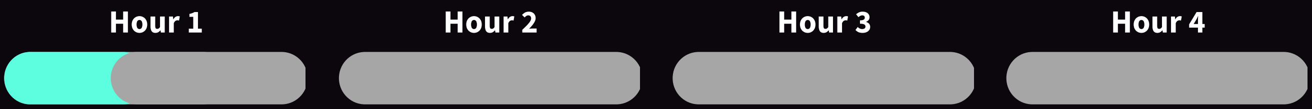
WITH SnapAttack	WITHOUT SnapAttack
10-15 minutes • Review telemetry in the Attack Capture Lab to understand relevant forensic artifacts • If the attack isn't already in the library, develop detection strategy by emulating the Attack in the Attack Capture Lab	**21-29 hours** • Research of vulnerability / exploit / adversary • Build out infrastructure for adversary emulation • Testing of exploit / TTP • Reviewing logs • Developing detection strategy • Design testing logic

Write

Once the engineer is secure in the detection logic and strategy, they have a clearer vision of what exactly will be required to protect against the threat - meaning they're ready to find or write the most applicable, effective detection as possible.

5. How do I find / build a detection that will defend against it?

After sufficient research, testing, and analysis of the detection strategy and logic, the engineer can confidently build or find a detection to comprehensively defend against the threat at hand. Once again, they often turn to public resources and forums like repositories and social media to determine whether a detection yet exists. If not, they must build the detection themselves, testing and validating along the way that it is working as desired and adhering to the relevant query language / tooling of their environment.

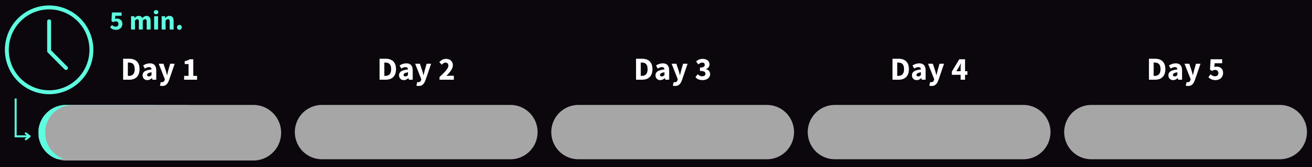


WITH SnapAttack	WITHOUT SnapAttack
20 minutes • Determine if existing detection exists in Detection Repo • If not, build a new one in No-Code Detection Builder • Peek into production data to determine performance - tune / refine as needed	**4 hours** • See if detection exists in public repositories, blog posts, tweets, etc. • Build detection - need coding knowledge and expertise • Hope detection works

Validate

6. How can I validate that my detection is working properly / will block the attack?

Just like a software developer must test applications for bugs prior to deployment, it's critical for detection engineers to test and validate detections to ensure each one performs as desired. Historical testing involves checking the detection against past attacks and data, and real-time testing can be accomplished through adversary emulation in a lab environment or by looking thoroughly into production data. Both are helpful in gaining a holistic perspective of how the detection will perform in the real world.

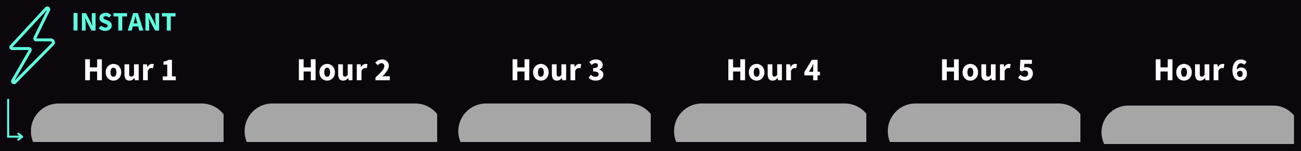


WITH SnapAttack	WITHOUT SnapAttack
5 minutes • Look at production data • Replay attack in production lab or sandbox in SnapAttack	**1-5 days** • Adversary emulation in production • Political wrangling to do bad things in production

Deploy

7. How do I deploy my detection to work across my environment?

Once the engineer is completely secure in the performance of their detection, it's time to deploy it to their environment. Deploying the detection entails deploying it across the production SIEM, EDR, and data lakes.



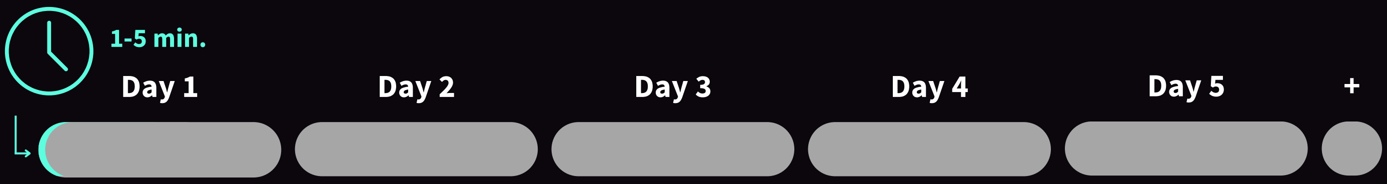
WITH SnapAttack	**WITHOUT SnapAttack**
INSTANT Translate to whatever language you need and click "deploy" (to multiple tools or for multiple detections) in at most two clicks.	**6 hours** • Deploy detection code to production SIEM • Deploy detection code to production EDR • Deploy detection to production datalake

Measure

With the detection deployed, it's critical to continuously monitor its performance against incoming attacks.

8. How confident are we that we can now defend against the attack?

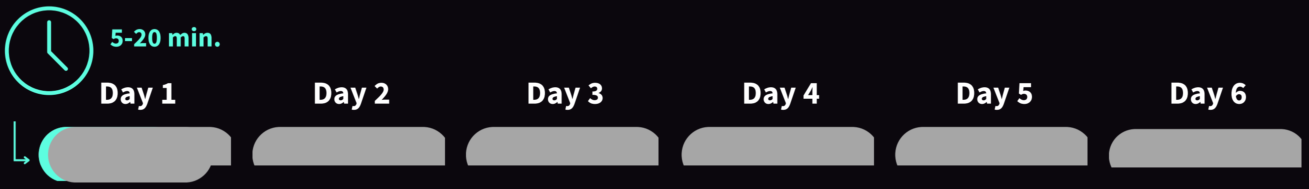
Continuous testing should be maintained even after deployment in case a threat does come through before analysts have realized the detection is inadequate. Engineers should look out for false positives, bugs, or excessive noise of any kind - if the detection is not functioning properly, it can be tweaked, tuned, and decommissioned if necessary.



WITH SnapAttack	WITHOUT SnapAttack
1-5 minutes • Review detection's confidence score in the Detection Repo • Review coverage of TTPs against existing defenses with the MITRE ATT&CK Matrix • Re-run attack validation regularly to ensure detections haven't failed due to environment / infrastructure changes	**5+ days** Wait and hope for the best.

9. How can we improve that confidence level?

After deployment, there is always room for improvement as adversaries will continue to hone their craft, invest in stronger tooling, and enhance their attack strategy. Additionally, new information or tactics surrounding the threat may arise, expanding or changing what is needed for the detection to remain effective. Monitoring and regularly reviewing the performance of false positives helps guide engineers towards any bugs or weaknesses which they can then remediate. Without a centralized tool to track, monitor, and tune the detection's performance, this responsibility rests on the shoulders of the detection engineer and their security team.



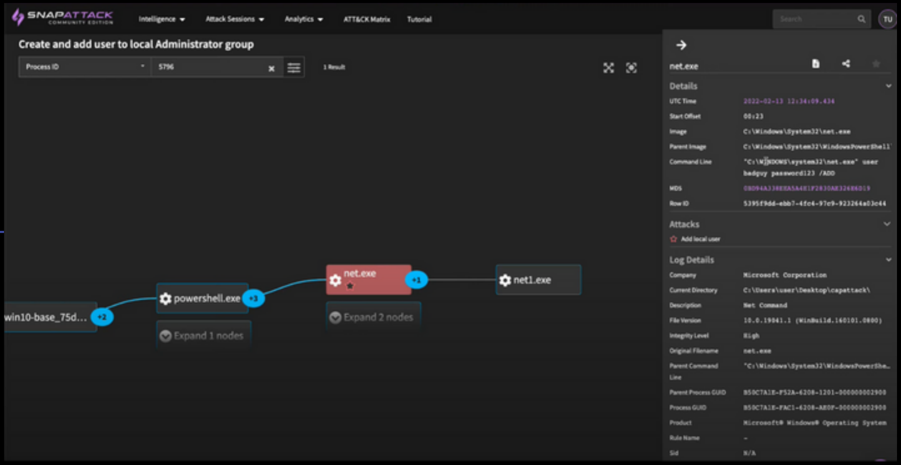
WITH SnapAttack	WITHOUT SnapAttack
5-20 minutes • Review confidence, false positives, and TTPs • Tune rule as needed	**1-6 days** • Wait to see results of SOC / great hunter • Hope it worked • Triage incidents when it doesn't work • Review and tune

Building Detections with SnapAttack Case Study

Researching, writing, validating, and deploying a detection can take the effort of several highly-skilled engineers, days of sophisticated work and triage, and many failed attempts before a successful detection is deployed. But with SnapAttack, that process is cut down to minutes and can all be done in one simple, centralized platform.

In comparison to traditional frameworks, SnapAttack can create a detection for a Log4J or behavior de-jour using our no-code detection builder. SecOps teams are then able to test detections to see if they work properly and deploy them directly into their security tools with just one click.

This enables proactive threat hunting, removes barriers to detection-as-code, and advocates for purple teaming – all in one integrated platform. SnapAttack enables hunters (and even uplevels junior analysts) to spend more time on the hunt and less time researching and developing detections.



Create and add user to local Administrator group

net.exe

Details

UTC Time	2022-02-13 12:34:09.434
Start Offset	00:23
Image	C:\Windows\System32\net.exe
Parent Image	C:\Windows\System32\WindowsPowerShell\
Command Line	"C:\WINDOWS\system32\net.exe" user badguy password123 /ADD
MD5	0BD94A338EEA5A4E1F2830AE3248D19
Row ID	5395f9dd-ebb7-4fc4-97c9-923264a03c44

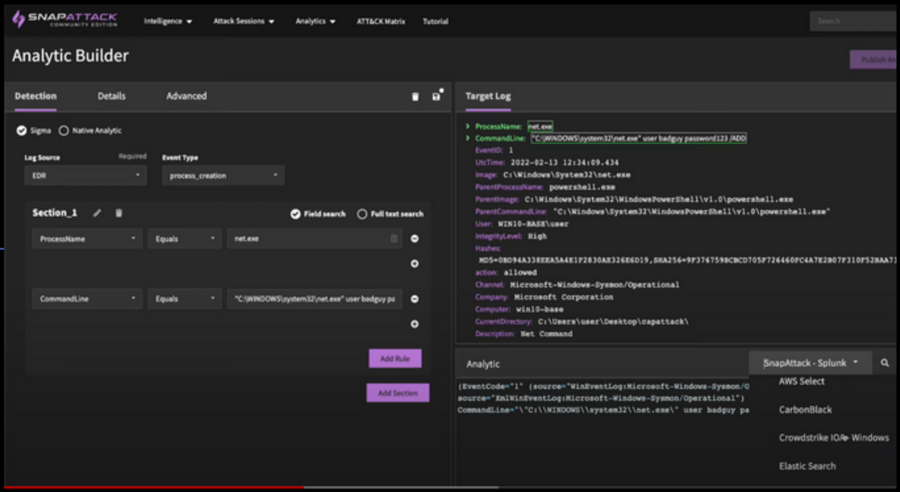
Attacks

Add local user

Log Details

Company	Microsoft Corporation
Current Directory	C:\Users\user\Desktop\capattack\
Description	Net Command
File Version	10.0.19041.1 (WinBuild.160101.0800)
Integrity Level	High
Original Filename	net.exe
Parent Command Line	"C:\Windows\System32\WindowsPowerShe…
Parent Process GUID	B50C7A1E-F52A-6208-1201-000000002900
Process GUID	B50C7A1E-FAC1-6208-AE0F-000000002900
Product	Microsoft® Windows® Operating System
Rule Name	-
Sid	N/A

View the attack from start to finish.



Analytic Builder

Sigma [✓] Native Analytic

Log Source: EDR (Required)
Event Type: process_creation

Section_1 — Field search [✓] Full text search

ProcessName	Equals	net.exe
CommandLine	Equals	"C:\WINDOWS\system32\net.exe" user badguy pa

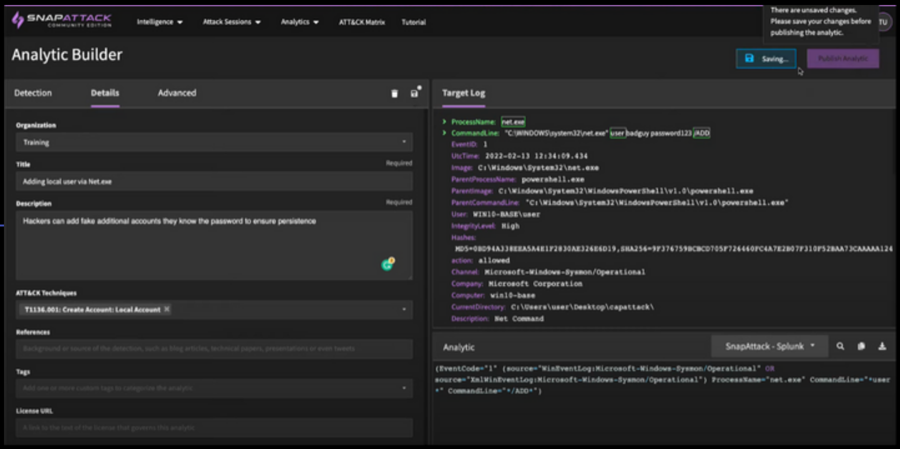
Target Log

```
> ProcessName: net.exe
> CommandLine: "C:\WINDOWS\system32\net.exe" user badguy password123 /ADD
  EventID: 1
  UtcTime: 2022-02-13 12:34:09.434
  Image: C:\Windows\System32\net.exe
  ParentProcessName: powershell.exe
  ParentImage: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
  ParentCommandLine: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
  User: WIN10-BASE\user
  IntegrityLevel: High
  Hashes:
   MD5=08D94A338EEA5A4E1F2830AE326E6D19,SHA256=9F376759BCBCD705F726460FC4A7E2B07F310F52BAA7
  action: allowed
  Channel: Microsoft-Windows-Sysmon/Operational
  Company: Microsoft Corporation
  Computer: win10-base
  CurrentDirectory: C:\Users\user\Desktop\capattack\
  Description: Net Command
```

Analytic — SnapAttack - Splunk

```
(EventCode="1" (source="WinEventLog:Microsoft-Windows-Sysmon/O
source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational")
CommandLine="\"C:\\WINDOWS\\system32\\net.exe\" user badguy pa
```

Select detection logic parameters in the point-and-click detection builder.



There are unsaved changes. Please save your changes before publishing the analytic.

Analytic Builder

Organization: Training

Title (Required): Adding local user via Net.exe

Description (Required): Hackers can add fake additional accounts they know the password to ensure persistence

ATT&CK Techniques: T1136.001: Create Account: Local Account

References: Background or source of the detection, such as blog articles, technical papers, presentations or even tweets

Tags: Add one or more custom tags to categorize the analytic

License URL: A link to the text of the license that governs this analytic

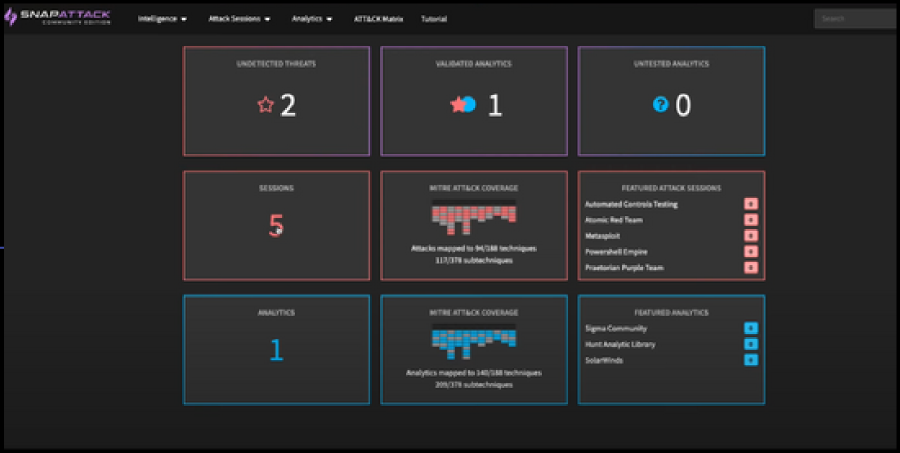
Target Log

```
> ProcessName: net.exe
> CommandLine: "C:\WINDOWS\system32\net.exe" user badguy password123 /ADD
  EventID: 1
  UtcTime: 2022-02-13 12:34:09.434
  Image: C:\Windows\System32\net.exe
  ParentProcessName: powershell.exe
  ParentImage: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
  ParentCommandLine: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
  User: WIN10-BASE\user
  IntegrityLevel: High
  Hashes:
   MD5=08D94A338EEA5A4E1F2830AE326E6D19,SHA256=9F376759BCBCD705F726460FC4A7E2B07F310F52BAA73CAAAAA124
  action: allowed
  Channel: Microsoft-Windows-Sysmon/Operational
  Company: Microsoft Corporation
  Computer: win10-base
  CurrentDirectory: C:\Users\user\Desktop\capattack\
  Description: Net Command
```

Analytic — SnapAttack - Splunk

```
(EventCode="1" (source="WinEventLog:Microsoft-Windows-Sysmon/Operational" OR
source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational") ProcessName="net.exe" CommandLine="*user
*" CommandLine="*/ADD*")
```

Test, save, and publish powerful, accurate, low-noise detections in minutes.



UNDETECTED THREATS 2

VALIDATED ANALYTICS 1

UNTESTED ANALYTICS 0

SESSIONS 5

MITRE ATT&CK COVERAGE — Attacks mapped to 94/188 techniques, 117/378 subtechniques

FEATURED ATTACK SESSIONS: Automated Controls Testing; Atomic Red Team; Metasploit; Powershell Empire; Praetorian Purple Team

ANALYTICS 1

MITRE ATT&CK COVERAGE — Analytics mapped to 140/188 techniques, 209/378 subtechniques

FEATURED ANALYTICS: Sigma Community; Hunt Analytic Library; SolarWinds

In under 30 minutes, deploy high-fidelity detections across query languages and security landscapes.

Conclusion

An established detection development lifecycle provides a powerful backbone to an organization looking to rapidly research, write, validate, and deploy high-fidelity detections in a repeatable framework. Whether developing their own or partnering with a third-party vendor, security teams should align themselves under an organized, repeatable process that enables collaboration, centralization, and a structured flow to build detections.

Clearly, detection development is a highly involved process that requires excessive effort, time, and personnel to achieve. A comprehensive threat detection tool can reduce the strain on teams by consolidating capabilities into one accessible, collaborative location.

SnapAttack provides a platform to manage each step of the detection development lifecycle and reduces the time it takes from research to deployment from days to minutes, with every tool and resource detection engineers could need all in one place. If you'd like to see SnapAttack in action or learn more, book a demo at snapattack.com/contact.

About SnapAttack
SnapAttack is the enterprise-ready platform that helps security leaders answer their most pressing question: "Are we protected?"

By rolling intel, adversary emulation, detection engineering, threat hunting, and purple teaming into a single, easy-to-use product with a no-code interface, SnapAttack enables you to get more from your technologies, more from your teams, and makes staying ahead of the threat not only possible - but also achievable.

Whether you're an analyst or a CISO, a red teamer or a blue teamer, SnapAttack unlocks the potential of your security operations and enhances existing toolsets.