Evadere Classifications

posts.specterops.io/evadere-classifications-8851a429c94b

Jonathan Johnson June 1, 2021

Jonathan Johnson
Follow
Jun 1, 2021

.

10 min read

Introduction

The term evasion is derived from the Latin word “evadere” which means — “To escape, to get away.” The defines evasion as — “The process whereby isolated personnel avoid capture with the goal of successfully returning to areas under friendly control.”

Fairly often I come across a post on Twitter or a new blog that talks about a new or technique. Whether explicitly stated or not. I’ve seen posts that discuss UAC Bypass, new methods of process injection, or LSASS dumping that will bypass certain sensors or even precise detections that are released (at first this doesn’t sound like it fits into an evasion technique, but this will be explained later within the blog).

This made me think — what does or truly mean? Are there different categories that these evasion techniques fit into? Lastly, if these techniques are to fit into categories — how can detection engineers leverage these for engagements?

Capturing the type of evasion that was executed would help identify the gaps within the environment, which in turn would help engineers understand if that gap is something they can fix or if it is something that is out of their control. Once these gaps are identified, prioritization can be created.

For example: say a red-team engagement is performed and the red-teamer the sensor, there are 3 scenarios that could be in effect:

1. The sensor doesn’t collect the correct data to correlate with the event
2. The sensor has a built-in detection that the red-teamer evaded (engineers don’t have insight into what the detection logic is)
3. The sensor collects the correct telemetry to provide visibility into the activity, but the engineer now has to create a custom detection.

I discuss this in the next section, but a priority could be set once the correct scenario above is identified.

For scenarios 2 & 3, if the data is there then setting a priority for creating the detection should be high.

For scenario 1, if the sensor doesn't collect data to correlate with the event then addressing this issue should be a priority.

Throughout this blog post, I will use the term to reference both and terms because their definitions are interchangeable.

Types of Evasions

Note: Throughout this write-up, I will be referring to terminology that is applied to the , a mapping created by Jared Atkinson. I would highly suggest reading his piece on that methodology before moving forward, as well as his talk at SO-CON — which goes into how to classify False Positives/False Negatives by leveraging base conditions.

Initially, when diving into this methodology I wanted to make sure I was applying these opportunities to a mapping that was already created to help with comprehension. To accomplish this I chose to tie each evasion opportunity to a piece of the Funnel of Fidelity. This allowed me to categorize these methods more easily while helping with how to prioritize each method. For example, if an environment is more focused on their detection processes, they would want to focus on the evasion opportunities that pertain to that piece of the funnel. However; regardless of which piece of the funnel an organization wants to focus on I would suggest focusing on the opportunities that pertain to the data collection piece first, as this flows through the whole funnel. This is where I chose to start my focus when it comes to this methodology.

When setting my focus on the data collection portion of the funnel I decided to spend a lot of time on data/ sensor tools. Often times I have seen someone state "this method of this attack bypasses sensor x", my question then becomes "does it bypass the sensor's logging capabilities, the sensor's/EDR's (Endpoint Detection and Response tool) default detections, or the custom detections you have set into place?" While breaking those things out, 2 different evasion techniques are identified along with 2 sub-techniques:

is broken up into two different sub-techniques: and . There is value in separating these two sub-techniques even though they fall under the same evasion category. One method is controlled by the customer/analyst: . The data is there for you to collect, enabling just needs to take place. The other method: is something the customer/analyst has zero control over. Neither one can control the sensor is capable of collecting, this falls on the developers of that tool. The breakdown is as follows:

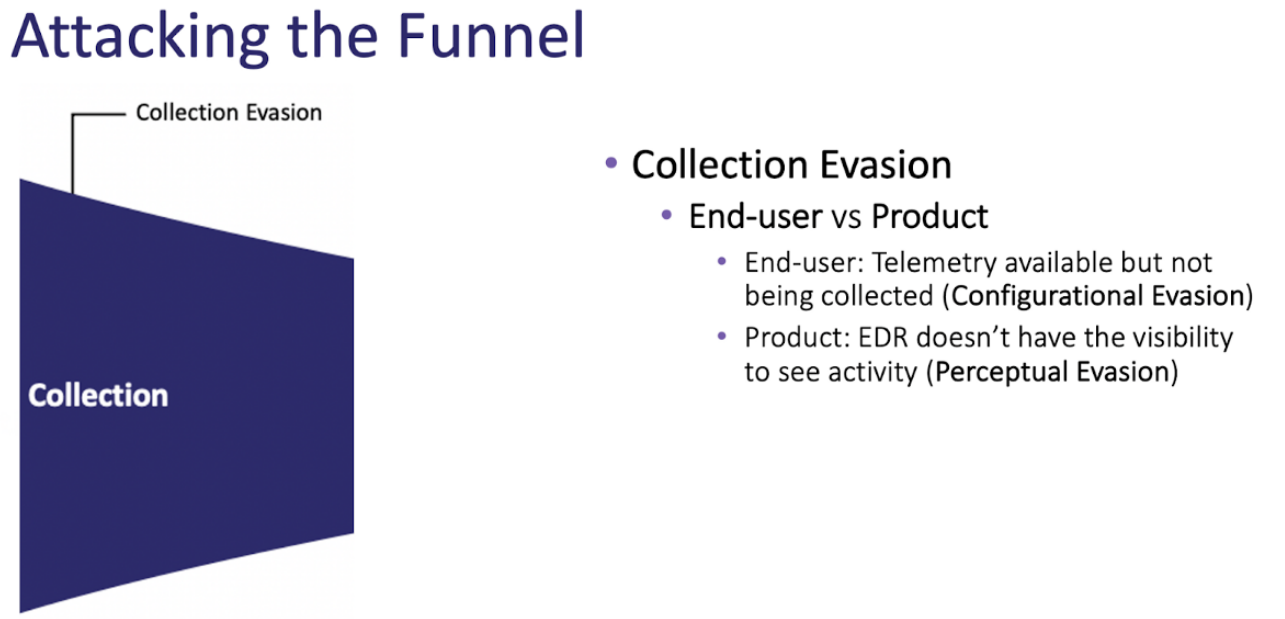
End-User vs Product:

End-User: Telemetry available but not being collected ()

> **Scenario**: sensor has data that provides visibility into behavior, but the current logging configuration is preventing coverage

Product: sensor doesn't have the visibility to see activity ()

> **Scenario**: An action leverages a low-level syscall to perform a behavior but the sensor is hooking on Win32 APIs for logging capability

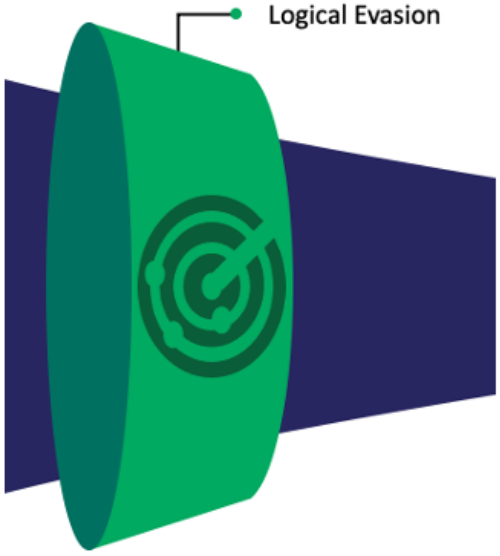


After identifying the initial evasion methods, I asked myself — "What if a sensor's built-in detection logic or a detection engineer's custom detection logic is evaded?" This topic is interesting because many times analysts don't have insight into an EDR vendor's built-in detection logic. Without having insight into this, how do engineers know what is covered within the environment? Since the engineer has little to no insight into this strategy, that is something they can't control. However; that doesn't mean that those strategies don't provide value to the organization, but when applied a gap is created. Identifying those gaps to the best of our ability is beneficial when creating custom detections. Custom detections are something the engineers can control. These detections are built upon a certain strategy or methodology. Is the detection precise where it is looking at command line parameters, certain binaries? Is it broader where it is looking for specific APIs that an attacker can leverage? There are a lot of strategies, but it is important to note — no detection is bulletproof. They all have gaps. These gaps are evasion opportunities for the attacker. This evasion opportunity will be classified as and is broken out below:

Logical Evasion:

Attacker evades the analytical logic implemented by a detection

> **Scenario: Intentional** — adversary finds/exploits the logic set in place
>
> **Scenario: Unintentional** — the detection engineer implemented a detection with gaps that were not covered, allowing for this evasion effort by the attacker



- Logical Evasion
  - Attacker evades the analytical logic implemented by a detection
    - **Intentional** - adversary finds/exploits the logic set in place
    - **Unintentional** - the detection engineer implemented a detection with gaps that were not covered, allowing for this evasion effort by the attacker

The past couple of evasion techniques have fallen under either data collection or detection engineering, but are there evasion opportunities within other parts of the detection and response pipeline? Say — Triage.

Before diving into the evasion opportunities that can be applied to Triage, let's first define what I mean by this. Within the detection and response pipeline, triage is meant to add context to the datasets that were captured by the base condition within detection. These contextual pieces get applied to the dataset, given the different attributes that were applied a priority is assigned per dataset, then this is passed to the investigation so that the analyst has a pre-applied knowledge base attached to the alert they are given. This allows the analyst to respond to an alert quicker. However; there is an evasion technique that could be applied to this Triage piece. This technique applies when context can't be applied to the dataset or when it's difficult to distinguish this dataset is different from "normal" behavior. This evasion technique is called: and is broken down below:

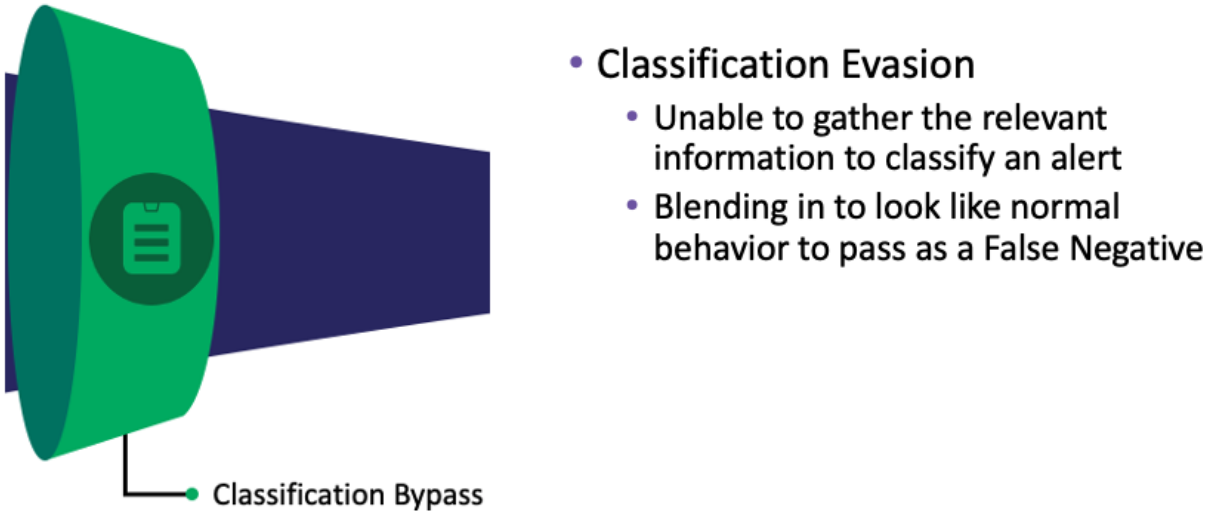
Classification Evasion:

Inability to gather the relevant information to classify an alert

> **Scenario**: Wanting to see if a service was created remotely, but no network data is being collected

- Blending in to look like normal behavior to pass as a False Negative

- This is an organization-specific example



Once classification is complete, imagine a scenario where a high number of alerts are triggered and the analyst experiences alert fatigue? Is there an opportunity where the adversary can either clean up their malicious activity or have enough time to perform another behavior allowing them to migrate further into the environment? This is something the Funnel of Fidelity tries to identify, how to take 1000000 events and funnel them down to 10s of leads. Except that isn't always going to apply. An alert's logic might be really broad causing a high volume of alerts to trigger, which after past through triage the analyst has a lot of data to look through, and not only can alert fatigue happen but data analysis paralysis can. When performed properly, the investigation process takes time. The analyst has to ingest the base dataset, the added contextual pieces applied by triage, and then if all the data they need isn't there to make a decision they have to go and grab that data to make the decision to escalate this alert or to close the alert. The evasion technique that applies to this scenario is: and is broken down below:

Temporal Evasion:

- The attacker has time to clean up their activity and move on due to the high volume of alerts
- The attackers' ability to complete a task is faster than the defenders' ability to remediate the task

> **Scenario:** The defender detected the attack, but couldn't act quick enough to stop the attacker from being successful

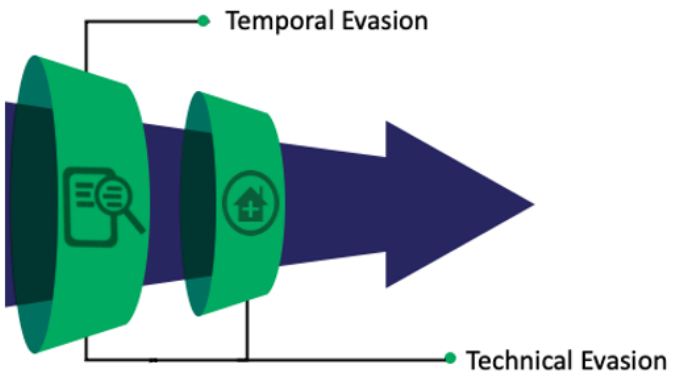
Next let's talk about a situation where an alert fires, context is added to the dataset, and the analyst successfully receives this incident. What happens if the analyst doesn't have the technical skills to effectively comprehend the alert which in turn affects their ability to respond properly? This isn't to mean these skills can't be learned, but this is something that must be taken into consideration when identifying possible evasion opportunities. The lack of

technical comprehension could come from the absence of knowledge within the detection document or the current status of the analyst's technical depth. This evasion opportunity can be applied to both the Investigation and the Response. This opportunity can be defined as and is broken down below:

Technical Evasion:

Analyst lacks the knowledge or skill set to respond to an alert correctly

> **Scenario:** Suspicious binary drops on a host, but the analyst doesn't have the proper skillset to reverse the binary to determine if it's malware



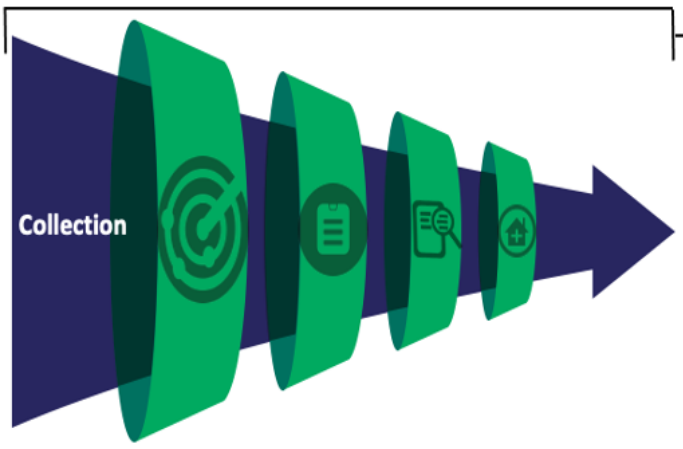
- Temporal Evasion
  - Attacker has time to clean up their activity and move on due to the high volume of alerts
- Technical Evasion
  - Analyst lacks the knowledge or skillset to respond to alert correctly

I quickly realized after moving through this methodology that there are other opportunities for an attacker to evade the current defenses, whether that be data collection, alert logic, triage, investigation, or response. Simply by one of these processes being broken, a control is misconfigured, a machine is offline, tampering with a sensor, or the organization is simply not performing one of these procedures in a robust manner. All of these possibilities lead to the failure of identifying a threat. This will apply to the whole funnel and can be classified as , which is broken down below:

Procedural Evasion:

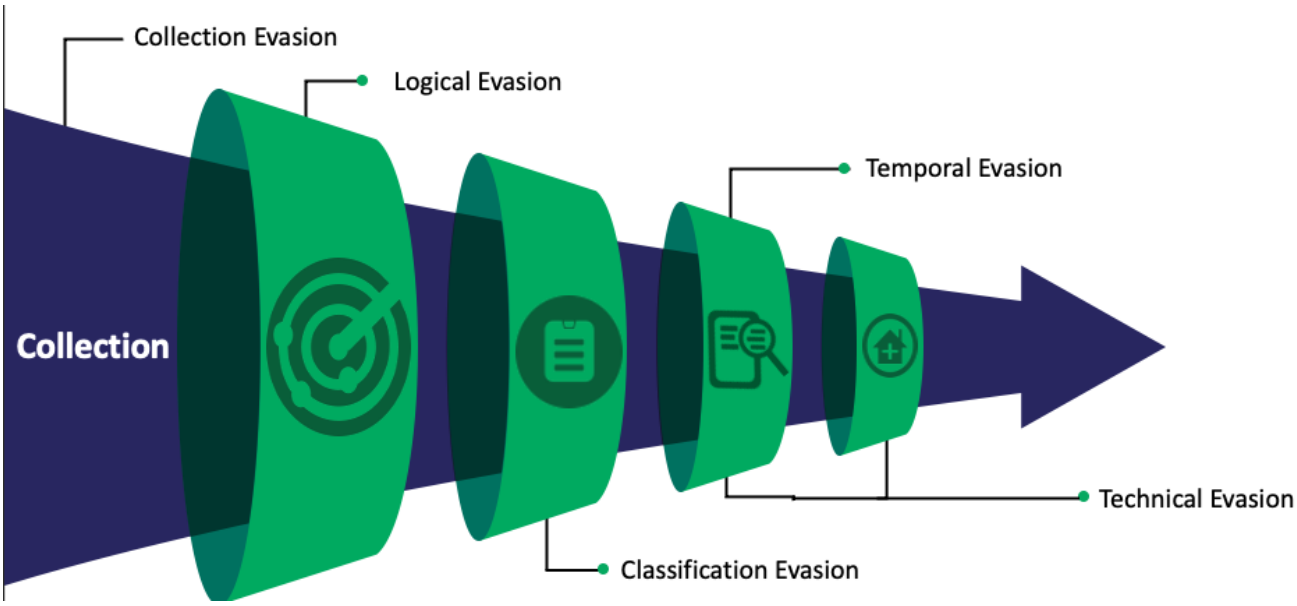
A control, procedure, process, or machine that fails to identify a threat

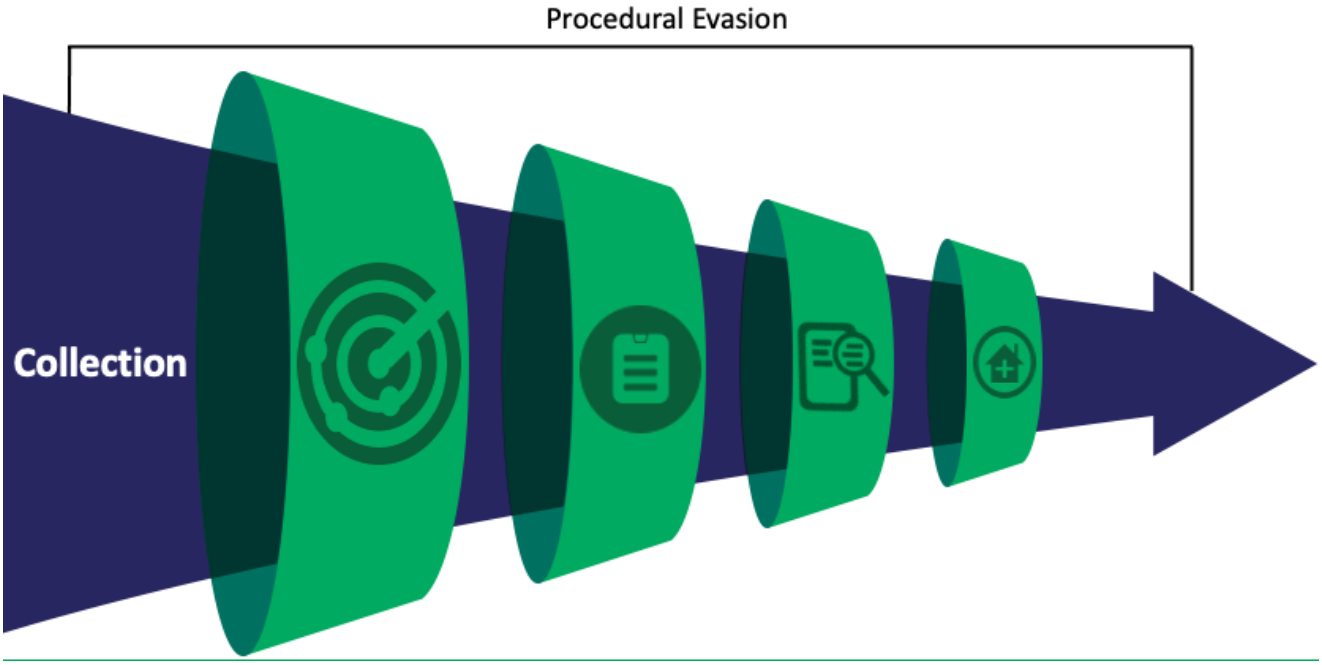
> **Scenario 1:** Sensor is turned off, preventing data to be collected
>
> **Scenario 2:** SACL was misconfigured for when a specific registry key is queried, the key is queried, but no insight is collected (MSHTA)
>
> **Scenario 3:** Unrecognized detection gaps, leading to a false sense of security related to technique coverage



- Procedural Evasion
  - A control, procedure, process, or machine that fails to identify a threat

Once we combine all of these different evasion opportunities the correlation to the funnel can be visualized as:





Conclusion

I think as time progresses, certain terms start to lose their meaning due to redefinition or misuse. We see this with a lot of terminologies, even outside of InfoSec. Evasion and bypass have seemed to fall under that category. Maybe it's not that they have lost their meaning, but more specific classification has to be applied when talking about the subject. This is what piqued my interest when it comes to this topic. Not only — how do we classify these evasion methods so that detection engineers can leverage them better, but also setting a common terminology that can be leveraged by the SOC and all sub-teams (threat research, red-team, detection engineers, etc).

It's important to remember that regardless if someone is part of a red-team, detection engineer, threat hunting team, etc the goal should always be to output something to the SOC. If terminology is misconstrued and everyone has their disparate definitions of a word, then the expectation of the output potentially gets lost.

I'm sure as time moves forward there will be more to add to this, but that is the exciting part of this. Everything evolves and is meant to.

Acknowledgments

A thank you to Jared Atkinson for helping with these classification naming conventions, as well as discussing this methodology with me.