OFFICE OF THE CTO REPORT

Consideration of Techniques and Technologies for Adoption of Zero Trust

By: Mudit Tyagi and Ken Arora

Expanding out from the "why"—the guiding principles[1]—of zero trust, the next concern is "how"—the techniques and technologies used to implement those principles.

From the broadest perspective, zero-trust principles can be applied to the entire application development lifecycle, including design of the system, hardware platforms used, and procurement procedures.[2] However, this paper discusses the operational aspects of implementing zero trust for defending applications and data in runtime.

Techniques and Technologies

Broadly speaking, zero trust security uses technologies to achieve one of three distinct goals:

1. Enforce transactional constraints: *Who* can do *What* to *Whom*.
2. Provide situational awareness to the human system operators.
3. Perform more advanced risk-reduction/remediation, based on the machine learning (ML)-assisted suspicion indicators and the risk-reward profile of the transaction(s) in question.

The following graphic depicts this overall zero trust security transactional model, with the following sections diving deeper into each class of technologies.

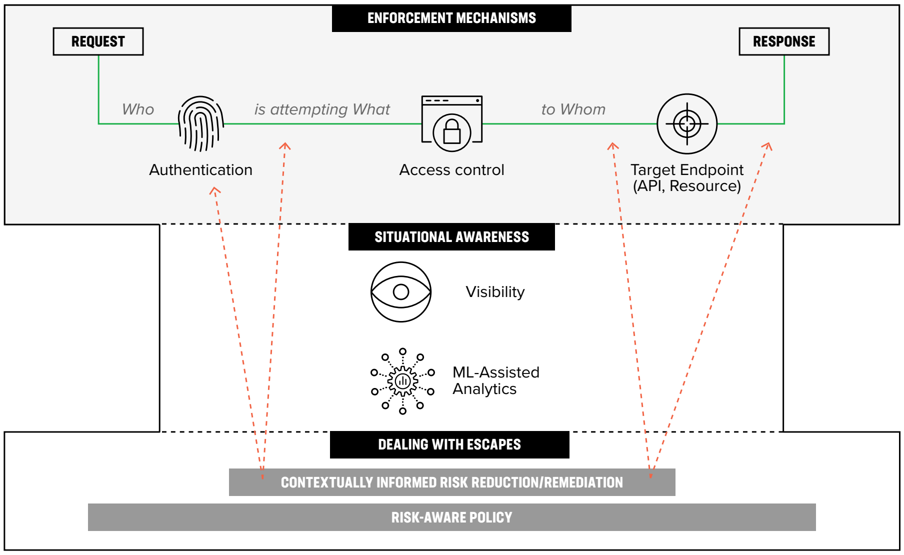


Figure 1: Zero trust security transactional model.

[1] https://www.f5.com/services/resources/white-papers/why-zero-trust-matters-for-more-than-just-access

[2] Zero trust can, and should, be applied even "to the left" of the CI/CD pipeline. Tools such as vulnerability assessment tools, static analysis, CVE databases, open-source code reputation databases, and supply chain integrity monitoring systems are consistent with the zero-trust mindset.

ENFORCEMENT: AUTHENTICATION AND ACCESS CONTROL

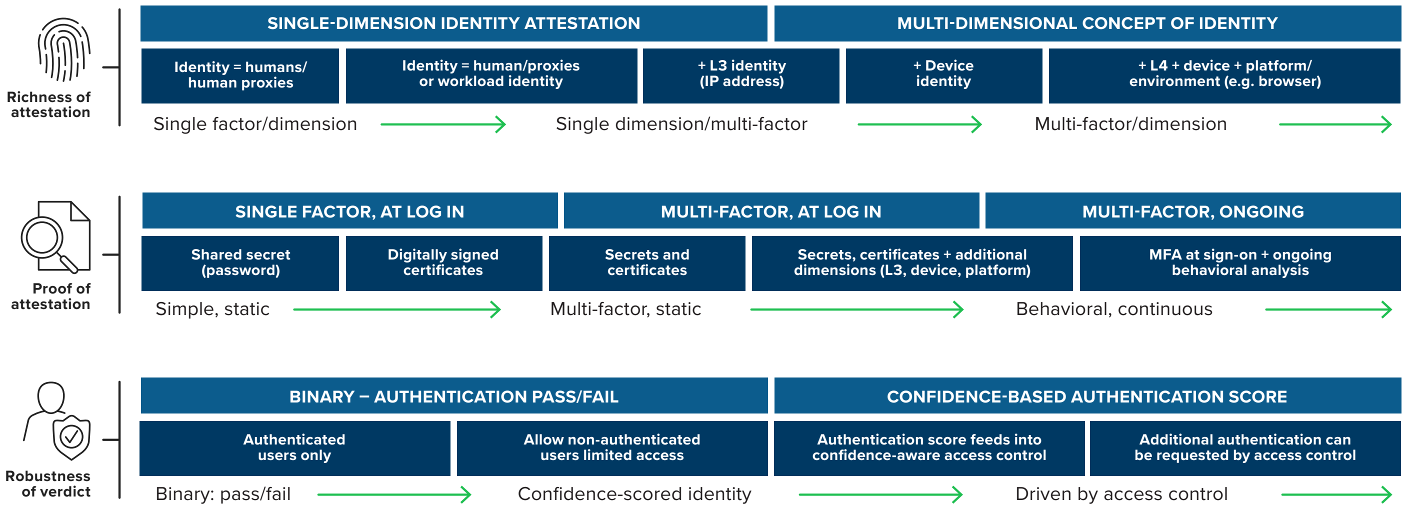
Motivations

The first two technologies—authentication and access control—are closely related and are directly motivated by the principles of “explicitly verify” and “least privilege,” since these technologies are at the core of enforcing “*Who* can do *What*.” More sophisticated implementations of authentication watch the ongoing behavior of an actor, capturing the mindset of “continuously assess.”

Authentication

Authentication technologies are all about building confidence in an attested identity: *Who* is acting in a transaction. The authentication process has three components:

1. There is an attestation, or claim, made by the actor, stating the actor’s identity.
2. There is some data, typically provided by the actor, providing proof of the veracity of the attestation.
3. The system produces a verdict or decision about the likelihood that the attestation is correct—the actor is who they claim to be. The graphic below depicts the different aspects of authentication, the maturity models for each, and is followed by a deeper discussion of each aspect.



Richness of attestation

SINGLE-DIMENSION IDENTITY ATTESTATION | MULTI-DIMENSIONAL CONCEPT OF IDENTITY

Identity = humans/ human proxies | Identity = human/proxies or workload identity | + L3 identity (IP address) | + Device identity | + L4 + device + platform/ environment (e.g. browser)

Single factor/dimension → Single dimension/multi-factor → Multi-factor/dimension →

Proof of attestation

SINGLE FACTOR, AT LOG IN | MULTI-FACTOR, AT LOG IN | MULTI-FACTOR, ONGOING

Shared secret (password) | Digitally signed certificates | Secrets and certificates | Secrets, certificates + additional dimensions (L3, device, platform) | MFA at sign-on + ongoing behavioral analysis

Simple, static → Multi-factor, static → Behavioral, continuous →

Robustness of verdict

BINARY – AUTHENTICATION PASS/FAIL | CONFIDENCE-BASED AUTHENTICATION SCORE

Authenticated users only | Allow non-authenticated users limited access | Authentication score feeds into confidence-aware access control | Additional authentication can be requested by access control

Binary: pass/fail → Confidence-scored identity → Driven by access control →

Figure 2: Zero trust security authentication maturity model

Attestation

The most basic form of attestation is often referred to as a “user”—a human, or agent acting on behalf of a human, that wishes to perform a transaction. However, in the case of zero trust used within an application an actor might be a workload (such as a process, service, or container), so the generalized concept of identity should include such actors. In other cases, the notion of *Who* includes not just the human or workload, but additional considerations or dimensions of identity. From that perspective, additional dimensions of identity might include the device or platform of the user/workload, or the ecosystem being used for the interaction or the location of the agent. For example, a user “Alice” may be on a PC tagged as “ABC-0001” using a specific, fingerprinted browser instance, sourced from IPv4 address 10.11.12.13.

Proof of Identity

Some systems allow unauthenticated users, sometimes referred to as “guests” or “anonymous” users, to perform a limited set of transactions. For such systems, the additional steps of proving identity and the system rendering a verdict is not relevant. However, for any specific attested identity, the following methods are commonly used to support that attestation:

- Knowledge of a shared secret, such as a password.
- Some sort of credential from a trusted third party, such as a signed certificate.
- Interrogation—either automated (such as device fingerprinting) or active (such as captcha challenge)—of the user, workload, or platform.
- Metadata related to the actor, such as the geolocation, IP reputation, or time of access.
- Behavioral analysis, such as passive biometric analysis or workflow patterns of application interaction.

Often, if a high degree of confidence is required, multiple methods are used. This is evidenced in the Google BeyondCorp model[3], which requires multi-factor authentication (MFA) before allowing higher value transactions. The more sophisticated authentication solutions associate a “confidence” with each identity and specify a minimum confidence level for each type of transaction, based on the value and risk of the transaction.

Static vs. Dynamic Authentication

Finally, note that some of these methods are not static, one-shot actions but can and should be ongoing as per the principle of “continuously assess.” In such cases, the confidence score assigned to the identity attestation can change up or down over time. For example, the browser fingerprint or IP address may change within a single user session, which could

[3] https://cloud.google.com/beyondcorp-enterprise/docs/quickstart

be viewed as suspicious, reducing confidence; or as more data is collected on the actor's behavior in a session, the confidence score may either increase or decrease depending on how the current behavior compares to past observations.

Dynamic authentication can work hand in hand with access control in more advanced systems. As the first level of this interaction, the access control policy can specify a minimum confidence score for different classes of transactions, as mentioned earlier. The next level of the interaction allows the access control subsystem to provide feedback to the authentication subsystem, typically asking for additional authentication to increase the confidence score to the minimum threshold.

Access Control

After using authentication techniques to ascertain *Who* is acting in a transaction, the next questions are: *What* is that actor allowed to do? And to *Whom*? This is the purview of access control technologies.

To take a physical security analogy, imagine you wanted to visit a military base. After the guards confidently determine whether you are a civilian, politician, or soldier, they would use that determination to decide which buildings you could enter and whether you could bring a camera into each building that you might be allowed to enter. The policy governing those choices might be very coarse and apply to all buildings (for example, "politicians can enter any building") or might be more fine-grained (such as "politicians can only enter building <A> and <B> but can only bring cameras into <A>").

Applied to the cybersecurity context, access control techniques should embody the zero trust principle of "least privilege." In other words, the optimal access control policy would only allow exactly those privileges that the actor requires and disallow all other privileges. Additionally, an ideal robust policy would be conditional on a specific minimum level of confidence in the authenticity of the actor's identity, with the confidence threshold specified at the granularity of each allowed privilege.

Therefore, value of an access control solution can be judged by how closely it aligns to these ideals. Specifically, a zero trust security solution must include access control and should evaluate the access control technology along the dimensions depicted below and described thereafter.

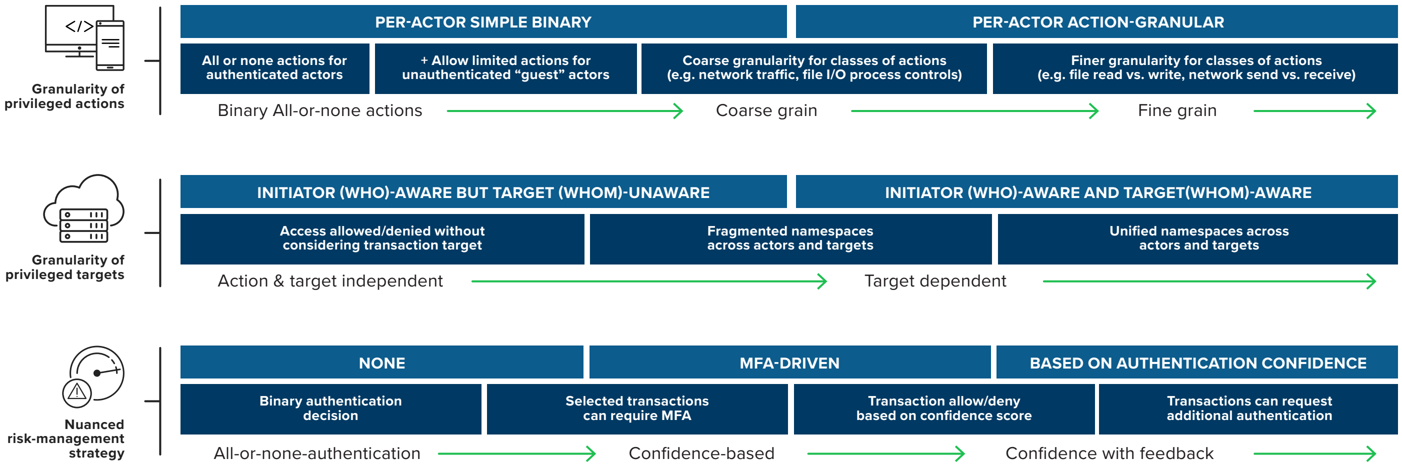

Granularity of privileged actions
PER-ACTOR SIMPLE BINARY
PER-ACTOR ACTION-GRANULAR
All or none actions for authenticated actors
+ Allow limited actions for unauthenticated "guest" actors
Coarse granularity for classes of actions (e.g. network traffic, file I/O process controls)
Finer granularity for classes of actions (e.g. file read vs. write, network send vs. receive)
Binary All-or-none actions
Coarse grain
Fine grain
Granularity of privileged targets
INITIATOR (WHO)-AWARE BUT TARGET (WHOM)-UNAWARE
INITIATOR (WHO)-AWARE AND TARGET(WHOM)-AWARE
Access allowed/denied without considering transaction target
Fragmented namespaces across actors and targets
Unified namespaces across actors and targets
Action & target independent
Target dependent
Nuanced risk-management strategy
NONE
MFA-DRIVEN
BASED ON AUTHENTICATION CONFIDENCE
Binary authentication decision
Selected transactions can require MFA
Transaction allow/deny based on confidence score
Transactions can request additional authentication
All-or-none-authentication
Confidence-based
Confidence with feedback

Figure 3: Zero trust security access control maturity model

1. How fine-grained are the access control privileges?

   a. What is the granularity of the action—the *What* in the transaction? Is it:

      i. Binary: Allow "all" transactions or "none." In the military base analogy, this would correspond to "all soldiers can enter any building and can do anything they want" and "civilians cannot enter any building."

      ii. Coarse: Granular to the API endpoint or "type" of transaction (such as file I/O, network communication, and process controls). In our example, this would allow a level of nuance around the allowed action, such as "politicians can enter buildings but not take pictures."

      iii. Fine: At the sub-API (that is, dependent on the parameters or the version of the API) or "sub-type" of transaction (such as file I/O read-only or TCP receive-only). Using our analogy one more time, this would allow finer-grain controls, such as "civilians can enter buildings only if accompanied by a soldier."

   b. Is there granularity as to the target of the action—the *Whom* in the transaction? Is it:

      i. Not target-granular: The access control policy does not consider the target of the action. In our example, this approach would map to the granularity of allowing entry to some buildings, but not others—buildings being the target of the "enter" action.

ii. Target-granular, infrastructure level: The access control policy can include the target of the action, but uses some infrastructure-specific tag/label, such as the IP address, DNS name, or Kubernetes (K8S) container name for the target. This would allow the policy to be building-granular, but every military base might have a different naming convention for buildings.

iii. Target-granular, identity-aware: The access control policy can include the target of the action, identifying the target using the same identity name space (for example, SPIFFE) used for the actor (the *Who*). The benefit over the prior approach is that all military bases would use a consistent scheme for how buildings are identified, making the policy more portable.

2. How does the access control solution deal with the concept of less versus more risky actions?

a. Not risk-aware: The access control solution treats all access control privileges identically, in terms of the decision to allow or disallow the action.

b. Risk management via MFA: The access control solution manages risk by allowing the policy to specify that some permitted transactions still be required to use Multi-Factor Authentication, based on the actor (*Who*), action (*What*), or target (*Whom*) involved in the transaction.

c. Risk management via confidence: The access control solution manages risk by allowing the policy to specify that any transaction may still require a minimum *confidence* level in the actor's authenticity, based on the action (*What*) and target (*Whom*) in the transaction. MFA may increase confidence, though other means may be used; in other cases, the transaction may not be allowed at all if the transaction is high value and the actor's authenticity is sufficiently uncertain.

Noting the principle of "continuously assess (and reassess)," any belief in the authenticity of the actor should adjust over time. In a simple solution it may simply be a timeout; in more sophisticated systems the confidence could vary based on observations of the actor's behavior over time.

SITUATIONAL AWARENESS: VISIBILITY AND ML-ASSISTED CONTEXTUAL ANALYSIS MOTIVATIONS

If authentication and access control are implementations of the “always verify” and “least privilege” mindset, then visibility and contextual analysis are foundational to the “continuously assess” and “assume breach” principles.

Visibility is the necessary precursor to analysis—a system cannot mitigate what it cannot see. Thus, the efficacy of the zero trust security solution will be directly proportional to the depth and breadth of telemetry that can be gathered from system operations and outside context. However, a modern visibility infrastructure will be capable of providing much more potentially useful data, metadata, and context than any reasonable unassisted human will be able to deal with in a timely manner. As a result of desires for both more data and the ability to distill that data into insights more quicky, a key requirement is machine assistance for the human operators.

This assistance is typically implemented using automated algorithms that span the spectrum from rule-based analysis to statistical methods to advanced machine learning algorithms. These algorithms are responsible for translating the fire hose of raw data into consumable and operationalized situational awareness that can be used by the human operators to assess and, if necessary, to remediate. For this reason, ML-assisted analysis goes hand in hand with visibility.

The generalized pipeline from raw data (visibility) to action (remediation) is shown below:

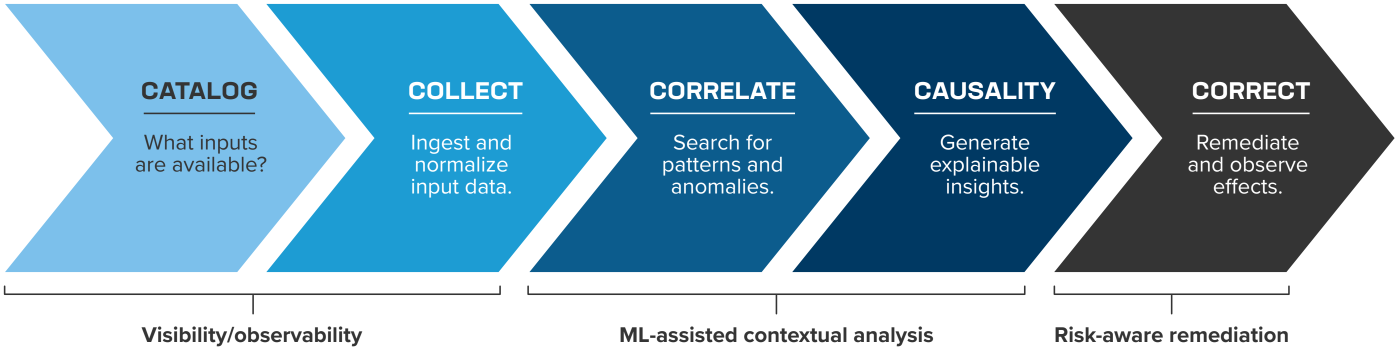

CATALOG
What inputs are available?
COLLECT
Ingest and normalize input data.
CORRELATE
Search for patterns and anomalies.
CAUSALITY
Generate explainable insights.
CORRECT
Remediate and observe effects.
Visibility/observability
ML-assisted contextual analysis
Risk-aware remediation

Figure 4: Zero trust visibility to remediation pipeline

Visibility

Visibility is the implementation—the “how”—of the “continuously assess” zero trust principle. It includes keeping an inventory of available data inputs (Catalog) and real-time telemetry plus historical data retention (Collect).

The maturity of a zero trust visibility implementation should consider four factors:

1. Latency: *How near to real-time is the data?*
   The latency provides a lower bound to how quickly a potential threat can be responded to. A zero trust solution’s latency should be measured in seconds or less; otherwise, it is quite likely any analysis—no matter how accurate—will be too late to prevent the impact of the exploit, such as data exfiltration/encryption or unavailability due to resource exhaustion. More sophisticated systems may allow both synchronous and asynchronous mitigations. Synchronous mitigation would inhibit completion of the transaction until full visibility and analysis are completed. Because synchronous mitigation is likely to add latency to the transaction, this mode of operation would be reserved for particularly anomalous or risky transactions, while allowing all other transactions to send telemetry and be analyzed asynchronously.

2. Consumability: *How readily can the data be consumed?*
   This concern is relevant if data arrives from multiple sources or types of data sensors, which is a common scenario. This factor typically breaks down into two sub-concerns.

   - At the syntactic level, how the data can be extracted and represented. For example, if a source IP reputation arrives as text field (such as “malicious,” “suspect,” “unknown,” etc.) in a syslog message from one source, and as a numeric, binary-encoded field in data file from another, then a canonical representation should be identified. Data serialization will be required to extract and transform the incoming data into that representation.
   - At the semantic level, different sources may not only vary in their syntax and transport, but also on the semantics. Using the IP reputation example again, one provider may provide a threat score as number, whereas another provider may classify the IP into a bucket such as “TOR node,” “DNS Control,” or “Phishing.” The visibility solution must also provide a mechanism for normalizing the incoming data into a consistent syntax.

3. <u>Completeness:</u> *What is the breadth and depth of available data?*
   One key value derived from a high-quality visibility solution is the ability to discover suspicious activities as an indicator of possible breach. To do so effectively the solution must receive telemetry across all the relevant “layers” of application delivery: the application itself, of course, but also the application infrastructure, the network infrastructure, any services applied to or used by the application, and even the events on the client device. For example, identifying a user coming in from a new device, never seen before, may be slightly suspicious on its own; but when combined with network information (such as GeoIP mapping from a foreign country) the suspicion level goes up higher. This suspicion level is manifested as a lower confidence score in the identity of the user. In the context of a zero trust security policy, when this actor attempts a high-value transaction (such as transfer of funds to a foreign account), the access control solution can choose to block the transaction, based on the low confidence.

   As it relates to zero trust mindset, the deeper and more complete the visibility solution is, the more effective the system can be in appropriately limiting transactions and detecting breaches.

4. <u>Governance:</u> *How well are statutory and license-based data use requirements supported?*
   Finally, any collection of data must be compliant with statutory and licensing requirements relating to the security, retention, and use of data. Therefore, a robust visibility solution must address each of these needs. Understanding the constraints on data use implied by governance must be factored into a zero trust visibility solution. For example, if an IP is considered Personally Identifiable Information (PII), then the use and long-term retention of IP addresses for analysis must cater to permissible use of the IP addresses.

Contextual Analysis with Machine Assistance

In addition to visibility, the other machinery required to implement “continuously assess” is the analytical tooling required to perform meaningful assessment; that is, to have assessment that can be operationalized by a zero trust solution.

One consideration for analysis is the scope and breadth of the input data. The inputs to the analysis algorithms can be limited to a single stream of data from a single source, or can look across multiple streams, including from various data sources and all layers of the infrastructure and application.

- The most basic contextualization is within the scope of a single “type” or “stream” of data (such as a stream of “user login information with time and Source IP address” events), typically with historical context and/or data across multiple users. This analysis can result in some basic insights, such as “this user <X> has never logged in during this time of day” or “this user <X> appears to always login immediately after other user <Y>.” The former may reduce confidence in identity; the latter may be indicative of some higher-level pattern, perhaps malicious.
- A more advanced contextualization considers the time-based and multi-instance scope not just for a single “type” of data, but also performs time-correlation across different “types” or “streams” of data. An example would be correlating the user login data with specific actions at the application layer (such as money transfer or email contact updates) or at the network layer (such as an IP-based geolocation lookup).

A second particularly relevant aspect of analysis in the zero trust framework is dealing with the volume and rate of data ingested, which will exceed the capability of any human to digest. Therefore, some sort of machine assistance to form human digestible insights is required. Once again, the sophistication of the assist can be described as a progression:

- **Visualization only:** The first step is bare presentation of statistics and events, typically via dashboards available on a management portal. The data presented is based upon collected data streams (usually either a time-series or a longitudinal cross-section across users/transactions within a fixed time window). More advanced dashboards offer the capability for sorting and grouping of the data, as well as drill-downs via filtering. In this model, the human does all the data exploration, event prioritization, analysis, and remediation, with the automation helping primarily with data exploration.
- **Visualization plus configurable static rules:** The most common next extension of visualization is the ability to allow the human to define statically specified rules, either for alerting or remediation. Examples of such rules would be “notify human <*John*> if <*CPU load exceeds 80% for a 5-minute period*>” or “require <*MFA via email*> if <*API [Y] is invoked and country is not [United States]*>”. These rules may be limited to pre-defined ones specified by the solution provider, or more advanced rule-based subsystems may also allow custom rules written by a SecOps engineer. In either case, any applied rules are typically controlled (and defined, if needed) via configuration.

- **Visualization plus ML assistance:** The next level uses more advanced techniques that find behavioral anomalies and/or transactions that match the patterns of previously identified malicious transactions. Although there are many techniques used (and a deep discussion of the technical and business tradeoffs is outside the scope of this paper) there are some key points to consider:
    - *Human effort required:* Some ML-techniques require "training" (aka supervised learning), which means that some reasonably sized corpus of data must be pre-categorized using a trusted, reliable "classifier." Often, to have trusted categorization, the "classifier" ends up being a human or set of humans. In those cases, the human effort required should be a consideration. Other techniques (aka unsupervised learning) detect anomalies and/or recognize patterns on their own, without human effort.
    - *Explainability:* The output(s) of a machine learning system results from evaluating the input data against a model. This model can be based on a series of simple easy-to-answer questions such as, "Has this user been seen on this device in the past month?" It could also be based on an easily explained similarity, such as "this user falls into the general pattern I have seen of users who never log in during working hours." In other cases, the model may be based on applying a complex set of vector algebra operations on the input data, with no easily discernable high-level logic. This last case is clearly less explainable—more of a "black box"—than the prior two examples. In some cases, explainability is an important factor for human understanding or auditability. For those instances, an explainable model will be strongly preferred over an inscrutable one.
    - *Confidence score availability:* As mentioned earlier, a metric indicating how confident the model is for a decision—in other words, a sense of the relative likelihood of a false positive or false negative—is often very useful. Some algorithms are such that a confidence metric output naturally falls out; for others, such as rule-based, a confidence level would need to be explicitly specified as one of the outputs. In either case, algorithms that can provide a confidence score are more robust than those that cannot.

As with the rules-based approach, ML assistance can be for detection only or it can be tied to automatic remediation. Additionally, ML assistance can be used in conjunction with a rules-based system, where the ML "verdict" (or opinion or confidence) can be used as an input into a rule, such as "do action <X> if <ML evaluator [bot_detector_A] reports bot with confidence greater than 90%>."

HANDLING ESCAPES: AUTOMATED RISK-BASED REMEDIATION

The final tenet of the zero rust mindset is to “assume breach.” To be clear and provide perspective, properly implemented authentication and access control methods are effective at preventing the overwhelming majority of malicious transactions. However, one should, out of an abundance of paranoia, assume that the enforcement mechanisms of authentication and access control will be defeated by some sufficiently motivated or lucky adversary. Detection of breaches, necessary for responding to these escapes in a timely manner, requires visibility and machine assisted analysis. Therefore, it is because the other enforcement mechanisms *will* be defeated on occasion that the technologies of visibility feeding ML-assisted contextual analysis are a critical need to feed the zero trust security backstop solution of risk-based remediation.

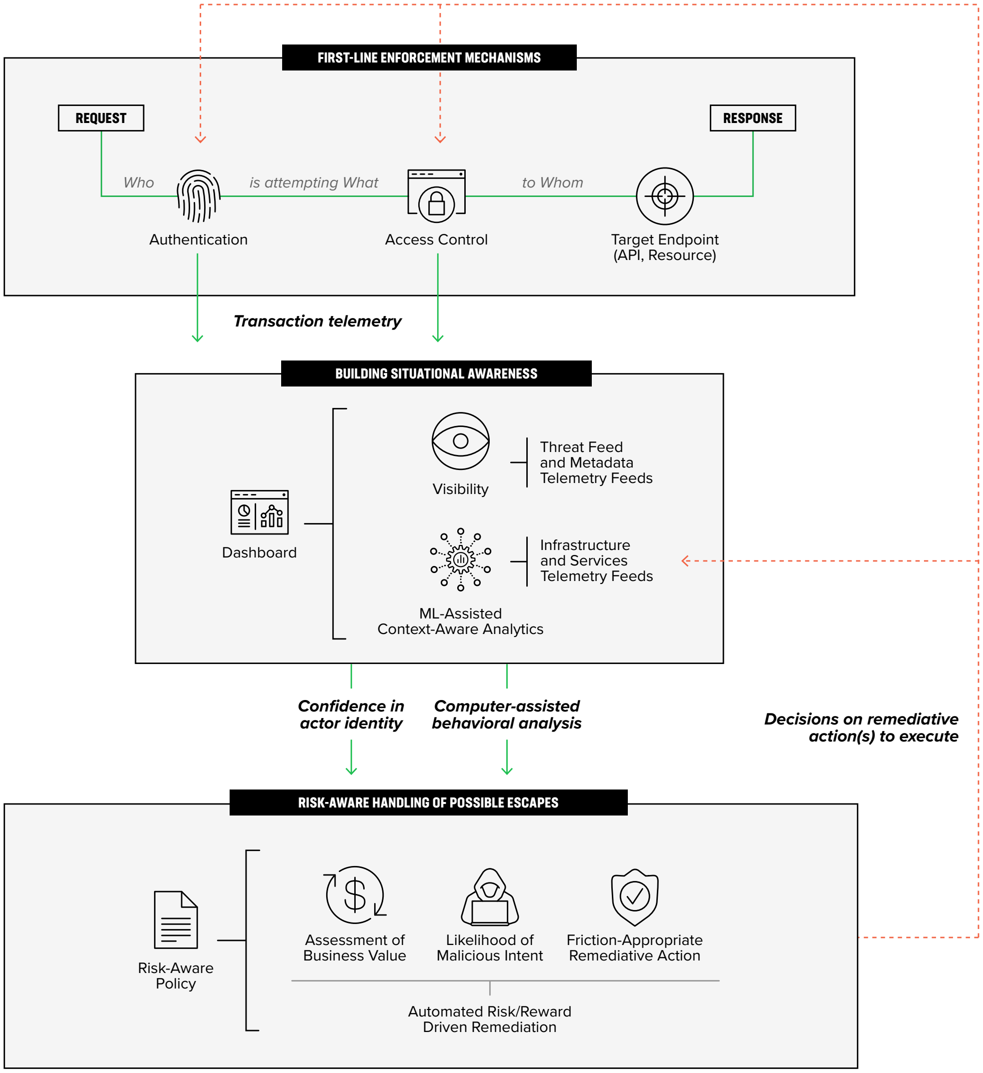

FIRST-LINE ENFORCEMENT MECHANISMS
REQUEST
RESPONSE
Who
is attempting What
to Whom
Authentication
Access Control
Target Endpoint (API, Resource)
Transaction telemetry
BUILDING SITUATIONAL AWARENESS
Dashboard
Visibility
Threat Feed and Metadata Telemetry Feeds
Infrastructure and Services Telemetry Feeds
ML-Assisted Context-Aware Analytics
Confidence in actor identity
Computer-assisted behavioral analysis
Decisions on remediative action(s) to execute
RISK-AWARE HANDLING OF POSSIBLE ESCAPES
Risk-Aware Policy
Assessment of Business Value
Likelihood of Malicious Intent
Friction-Appropriate Remediative Action
Automated Risk/Reward Driven Remediation

Figure 5: Zero trust risk aware remediation

For the “false negative” cases where an actual malicious transaction did defeat authentication and access control, the mechanism of automated risk-based remediation should be used as a backstop. But because this technology is applied as a backstop against transactions that passed the prior enforcement checks, there is a higher concern around incorrectly flagging what was, in truth, a “true negative” (a valid, desirable transaction) into a “false positive” (incorrectly flagged as malicious transaction). To mitigate this concern, any remediation actions triggered by a belief in possible maliciousness, that somehow was not caught by authentication or access control, should be based on the following three factors[4]:

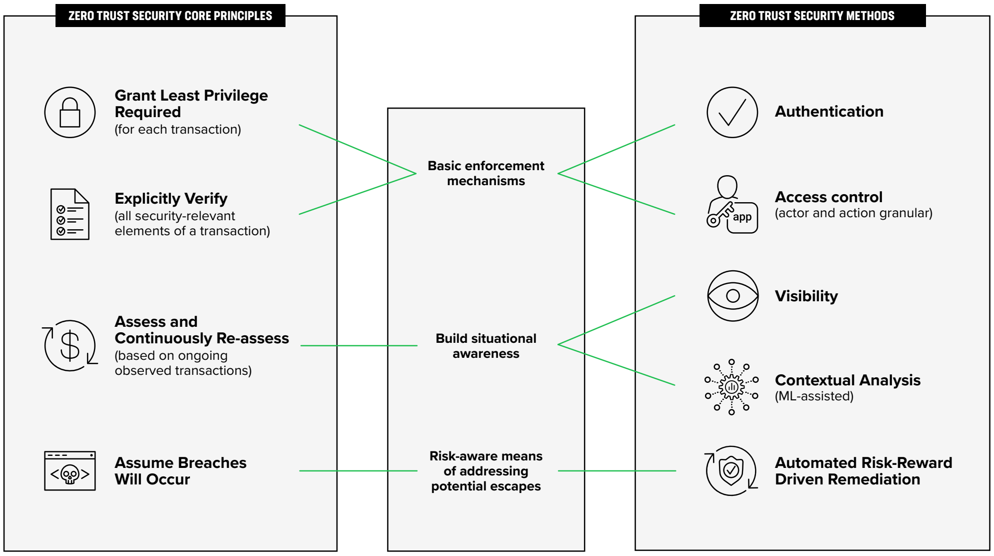
- **Business value of the transaction:** Different transactions will have differing levels of risk and reward. For example, a transaction viewing a bank account is less risky than a transaction that transfers money into a foreign account. Similarly, for an airline, purchasing a ticket is likely to be a higher business reward transaction, relative to a transaction that lists current flight delays. The *business value* is the benefit of the potential business reward weighed against the risk profile of the transaction. It is more acceptable to tolerate any “false positive” effects from speculative remediation to low-reward/high-risk transactions, as compared to high-reward/low-risk transactions that have high business value.
- **Confidence in the “maliciousness” belief:** Of course, not all suggestions for speculative remediation are equal. Specifically, in addition to the risk-reward mentioned above, the other key factor is confidence in the beliefs around that transaction, such as the confidence in the attested identity of the actor. Roughly speaking, the likelihood of “false negative”—that is, the probability that enforcement mechanisms did not trigger, but should have—is inversely proportional to confidence in the actor. And because reduction of “false negatives” is the goal of risk-based remediation, the lower the confidence, the more biased the system should be towards applying remediation.
- **Impact of the remediation:** Finally, not all remediations are binary decisions—allow or block. In fact, a variety of risk-reduction techniques are possible, ranging from some that are user-visible (for example, asking for additional credentials/MFA, providing a challenge such as a captcha), to others that are mostly invisible to the user (performing deeper traffic inspection such as DLP, or doing a more advanced/compute-intensive analysis). Therefore, the impact of performing these additional forms of risk-reduction—as measured by friction experienced by the user and operational costs for the application (such as for DLP or compute-heavy analysis)—is the third factor to consider. Low impact, customer-transparent remediations are preferred to higher impact

[4] Note that the line between contextual, risk-aware access control and the general topic of risk-aware remediation is a fuzzy one, and some overlap does exist.

customer-visible challenges, and outright blocking of the transaction is the least attractive option. However, blocking may be appropriate when confidence is high, or for risky transactions that cannot sufficiently increase confidence via the other risk-reduction mechanisms.

Conclusions

Zero trust security is a more modern take on prior approaches to security such as defense in depth, extending the prior art by taking a transaction-centric view on security—*Who* is attempting to do *What* to *Whom*. This approach enables securing not only external access to an application but is applicable to protecting the application internals as well.[5] Given this foundational transactional view, zero trust security is rooted in a set of core principles that are used to defend applications within today's more complex and challenging environment, with the principles then mapped to a set of subsystem-level solutions, or methods, that embody those principles. The core principles and how they map to solution methods are summarized below.



ZERO TRUST SECURITY CORE PRINCIPLES

Grant Least Privilege Required
(for each transaction)

Explicitly Verify
(all security-relevant elements of a transaction)

Assess and Continuously Re-assess
(based on ongoing observed transactions)

Assume Breaches Will Occur

Basic enforcement mechanisms

Build situational awareness

Risk-aware means of addressing potential escapes

ZERO TRUST SECURITY METHODS

Authentication

Access control
(actor and action granular)

Visibility

Contextual Analysis
(ML-assisted)

Automated Risk-Reward Driven Remediation

Figure 6: Zero trust security core principles and security methods

These tools—the methods of authentication, access control, visibility, contextual analysis, and risk-aware remediation—are necessary and sufficient to prevent a wide variety of attack types.

[5] Often referred to as "East-West" intra-app protection, as opposed to "North-South" to-the-app protection.