Zero Trust in Zero Trust?

Virgil D. Gligor

December 17, 2022

CMU-CyLab-22-002

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Zero Trust in Zero Trust?

Virgil D. Gligor
Carnegie Mellon University
CyLab Technical Report 22-002
December 17, 2022

***Abstract*—We review the standard definitions of *trust, zero trust*, *trusted service*, and *trust establishment*, and show that *zero trust* is unachievable in any enterprise network; i.e., at least one security property is impossible to establish unconditionally with confidence for some devices and many others are impractical to establish for other devices. In fact, zero trust has meaning only as an unreachable limit of trust establishment. Since NIST's *zero-trust architectures* cannot be about *zero trust*, we review their key characteristics and show that their main goal of limiting the effects of security breaches to single trust zones is often unmet. These architectures can never serve as security models nor can they be used to protect critical infrastructures as they cannot counter many common attacks, much less advanced ones. However, mature zero-trust architectures can reduce recovery costs after breaches, but the reduction is lower than provided by some alternate techniques.**

In view of these facts, it seems surprising that a 2021 Presidential Executive Order incorrectly calls NIST's zero-trust architecture a "security model," mandates its adoption, and frequently requires trust establishment, which exclude zero trust. Nevertheless, these architectures are motivated by practical goals. They rely on low-cost security assurance to limit some penetration damage and decrease recovery cost. They aim to detect trust-zone penetrations early by continuous monitoring of network devices. They maintain backward compatibility with existing (insecure) commodity software to facilitate timely deployment. In contrast with low-cost assurance of these architectures, trust establishment encourages flexible cost allocation among security functions and assurances, risk reduction, and adversary deterrence.

I. Introduction

Recent government and industry initiatives have vigorously promoted *zero trust architectures* [1] for network security. Specifically, the first ever Presidential Executive Order on cyber-security mandates use of these architectures throughout the US government [2], [3] and various entities are already planing deployment [4], [5]. Remarkably, a vast majority (i.e., 83%) of security and risk professionals say that zero trust architectures are essential to their organizations and most (i.e., over 80%) plan adoption this year [6]. This unquestionable demand has encouraged vendors to "zero-trust wash" their security products by claiming that all are zero-trust compliant [7]. Undoubtedly, the term "zero trust" is no longer a mere security buzzword as defined in Appendix B. As of now, its Google search count represents approximately 18% of the search results for the term "network security." Zero trust has become a slogan that hyperbolically conveys achieving an impossible security state at a time when cyber-crime is poised to exceed 1% of the global GDP [8] and more in US.

Given their impact, government and industry initiatives promoting zero-trust architectures require some scrutiny to delimit their usefulness. This report does not address key implementation challenges of zero-trust architectures, which are summarized in Section IV-C. We optimistically assume that they will be solved in the next few years. Instead, it focuses on basic problems that remain after resolving these challenges. This enables us to assess the virtues and limitations of these architectures that persist long term.

Basic problems. We focus on three problems. First, after reviewing standard definitions of the terms trust, zero trust, trusted service, and trust establishment (Section II), we show that zero trust is *unachievable* in any enterprise network; i.e., it is impossible to establish a given security property unconditionally with certainty for some network devices and impractical to establish other security properties for other devices (Section III). To gain broader perspective, we give a few examples of *zero trust* failures outside of traditional networks and show that they range from almost zero trust, to conjectured, impractical, and even undesirable zero trust. (Appendix A). We argue that *trust establishment* is a real-life foundation of security and zero trust has meaning only as its *unreachable limit* (Section III-C).

Second, we review the motivation and goals of the *zero trust architecture*, or *zero trust model* [1], [2], [3], since they must differ from the unachievable notion of *zero trust* (Section IV). We show that zero-trust architectures are *unsound* and *inadequate* for protecting critical infrastructures. They are unsound because they fail to meet their primary goal (Section V-B). That is, minimizing trust zones to limit an adversary's "lateral" movement after zone penetration fails to counter cross-zone attacks, which enable such movement. Also, these architectures lack minimization criteria for trust zones where least privilege operation is insufficient; e.g., to separate administrative duties. Furthermore, they fail to limit the number of external zones that need to be trusted by an enterprise network, thereby expanding an adversary's locus of attack against the enterprise without meaningful recourse.

Lack of soundness does *not* mean that these architectures are useless for enterprise networks (Section VII-C). However, it does mean that they can never serve as security models despite US government insistence [1], [2], [3], [4], and cannot be relied upon to protect critical infrastructures; see definitions of these infrastructures at *https://www.cisa.gov/critical-infrastructure-sectors*. Their inadequacy for pervasive government use is manifest: they fail to support many, and prevent implementation of some, government security requirements (Section VI).

Third, zero-trust architectures have demonstrably *low but non-zero defense value* for enterprise networks. Admittedly, adversary penetrations are inevitable [5], and often occur can multiple times per year [9]. These architectures can neither counter nor deter common adversary attacks [10] — much less the advanced ones (Section VII-B). Their virtue is the reduction of penetration-recovery costs. However, the cost reduction is lower than that of other techniques (Section VII-C).

Practical Goals. Zero-trust architectures have three practical goals. They employ only low-cost assurance for commodity software which, despite its low defense value, can limit some attack damage and decrease penetration-recovery cost. They monitor network devices of trust zones continuously to detect inevitable penetrations early. They maintain backward compatibility with existing (insecure) commodity software to enable timely deployment without major re-design.

What justifies these goals? In describing the dismal state of cyber-security before zero-trust architectures, the first US National Cyber Director, Chris Inglis, has been quoted[1] as saying "*it's all offense and no defense*." In view of this, low but non-zero defense, early detection of inevitable penetrations, and reduced attack damage and recovery cost are an improvement over the *status quo ante*.

II. Background: Trust, Zero Trust, Trusted, and Trust Establishment

A. *Basic notions*

We review the definitions of *trust*, *zero trust*, and *trusted* in their general sense, which are accepted in security and beyond. We show how *trust establishment* can reduce the liability of (non-zero) trust and enable a system to become trusted.

Trust. The general definition of *trust* (noun) and its most common interpretation is given by the *Oxford Languages Dictionary* (e.g., via a *Google* search):

1. Firm belief in the reliability, truth, ability, or strength of someone or something.

1.1. Acceptance of the truth of a statement without evidence or investigation.

In security, the noun *trust* refers to beliefs in security properties of a system or network component, or more, *without* verification or monitoring. This implies that trust has the liability that an adversary can control the component operation *without* detection. Hence, the *need to trust* is always detrimental security. If beliefs in a security property can be measured (e.g., by probability, or ordering, metrics), then their liabilities can be demonstrably reduced[2].

Zero trust. The need to trust is reduced by decreasing the number of unjustified beliefs and increasing security-property

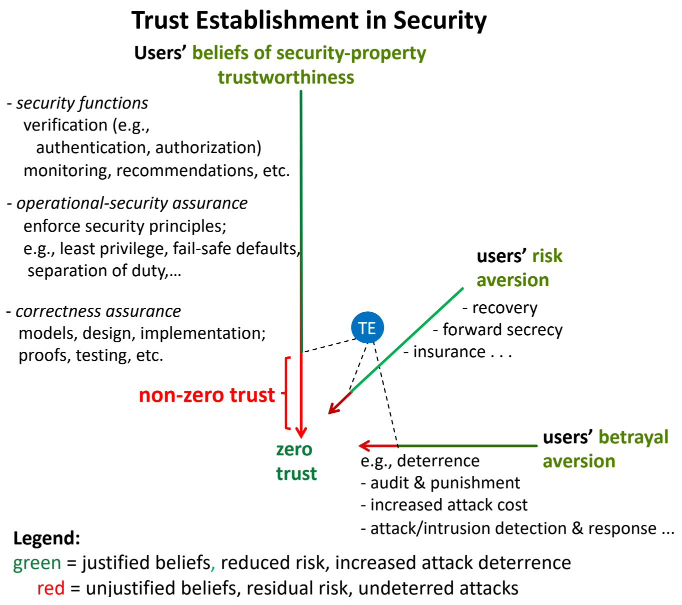

Trust Establishment in Security
Users' beliefs of security-property trustworthiness
- security functions
verification (e.g., authentication, authorization)
monitoring, recommendations, etc.
- operational-security assurance
enforce security principles;
e.g., least privilege, fail-safe defaults, separation of duty,...
- correctness assurance
models, design, implementation; proofs, testing, etc.
users' risk aversion
- recovery
- forward secrecy
- insurance . . .
TE
non-zero trust
zero trust
users' betrayal aversion
e.g., deterrence
- audit & punishment
- increased attack cost
- attack/intrusion detection & response ...
Legend:
green = justified beliefs, reduced risk, increased attack deterrence
red = unjustified beliefs, residual risk, undeterred attacks

Fig. 1: Trust, Zero Trust, and Trust Establishment (TE)

assurance; e.g., by verification and monitoring. Reduction, which is sometimes called *minimization* [1], stops when it is no longer possible or practical. When it's no longer possible, *all* beliefs in any security property of any component are justified *unconditionally with certainty* (i.e., independent of other services and with 100% probability) and there is no security liability left. This ideal limit is called *zero trust*. Most often, however, the reduction stops when the opportunity cost of further security-property justification exceeds a desired limit. In this case, some beliefs are not (unconditionally) justified and some liability remains. Since different minimization-stopping points exist, multiple practical limits appear, which are encompassed by *non-zero trust*.

The vertical axis of Figure 1 illustrates intuitively (i.e., without giving a belief metric) how accumulating evidence of security-function use, security-principle enforcement, and correctness assurance reduces the need to trust security properties and hence liability, and drives them towards the *zero trust* limit. If all beliefs for all security properties of a network component could ever be reached, then *zero trust* is achieved for that component. In Section III below, we show that this ideal limit is unachievable; i.e., it is impossible for some network devices and impractical for others.

Trusted. In standard definition, when the adjective *trusted* qualifies a *service* it implies that the service is reliable and its operation yields truthful results. Similarly, as a verb tense (i.e., past tense, past participle), *trusted* allows someone to have, use, or look after a service of importance or value *with confidence*; see common use in the *Oxford Languages Dictionary*. In security, a *trusted service* means that the beliefs in the service's security properties are backed by *some*, possibly independently evaluated, assurance evidence; e.g., trusted path, trusted recovery, trusted facility management [12] and equivalents [13]. This implies that the need to trust the service is reduced since some assurance evidence exists that justifies

[1]The full quote "If I were to score cyber the way we score soccer, the tally would be 462−452 twenty minutes into the game. In other words, it's all offense and no defense." is given by Nicole Pelroth in *This is how they tell me the world ends*, page 232, Bloomsbury Publishing, New York, NY, 2020.

[2]Belief-justification metrics depend on adversary definitions. For example, *limited* formal code-level verification can yield weaker belief justifications in networks facing an Internet (e.g., nation-state) adversary and stronger in special-purpose systems disconnected from the Internet. Also, some metrics may yield incomparable strengths for different adversaries yielding partial orders of belief justifications [11].

beliefs of trustworthiness in that service.

If *zero trust* in a service were possible, it would assure that the service is *trusted* unconditionally with certainty. In contrast, *non-zero trust* in a service means that not all beliefs of trustworthiness of the service's security properties are fully justified; see the *non-zero trust* region of Figure 1. Then, what additional conditions guarantee that a service *can become trusted*? In other words, how can a service become trusted if *zero-trust* is unachievable? *Trust establishment* [14], [15] answers this sufficiency question as argued below.

B. Trust establishment in security

How to establish trust among humans has been a subject on substantial research in *behavioral economics* [16]. Foundational results are derived by scientific experiments using trust games and neurobiology, and go beyond the well-understood need to decrease individual users' risk aversion. The direct application of these results to trust establishment in networks of human and computers has been explained in some detail [14] and illustrated in Figure 1. In summary, trust establishment in a network service or component that can be attacked by an adversary requires that

1) users' beliefs of trustworthy component operation are justified at some level of confidence; e.g., directly by security functions and assurances or indirectly by trustworthiness recommendations; and

2) users' risk aversion towards using the service or component is decreased; e.g., by risk mitigation measures, such as recovery, forward secrecy, insurance; and

3) users' betrayal (e.g., inequity) aversion caused by component misbehavior when controlled by an adversary is decreased; e.g., by adversary-deterrence measures, such as audit and commensurate punishment, increased attack cost, and attack detection and response.

Example. Trust can be established among strangers in e-commerce protocols. For instance, *eBay* encourages users to engage other users in e-commerce as follows. First, it helps users create beliefs of trustworthiness in others by employing a trustworthiness rank; i.e., an *eBay* recommendation algorithm ranks individual user trustworthiness in various types of transactions. Second, if a user selects some other user for a sale-purchase transaction where the selected user has a high *eBay* recommendation, *eBay* decreases first user's risk aversion by selling transaction insurance. Third, *eBay* employs a deterrence measure: if, after auditing transaction records, it finds that a highly-recommended user cheated an unsuspecting user, it evicts the cheater. Note that decreasing risk aversion without decreasing betrayal aversion, or the other way around, would eventually degrade *eBay*'s business model and render the beliefs of trustworthiness created by the recommendation system insufficient for protocol's security. ■

Although satisfying any proper subset of the three requirements above is insufficient [16], trade-offs in the extent to which each requirement is satisfied are often necessary.

Security of many other social protocols over the Internet are also based on trust establishment [15].

III. Zero trust is unachievable in security

In this section, we show the impossibility of zero trust for a specific security property of "black box" devices, which are used in *all* servers and endpoints of enterprise networks. Then we argue that zero trust is impractical for other security properties of enterprise services and devices. Since zero trust is unachievable, we explain why trust establishment can be a practical foundation of network security, and why zero trust can be viewed as an unreachable limit of trust establishment.

A. Impossibility

Zero trust is unachievable for "black box" devices since at least one security property cannot be justified *unconditionally*[3] and *with certainty*; i.e., with probability 1 in finite time[4].

Black-box devices. The main characteristic of *black-box devices* is that they cannot be opened so that their internal code and data structures can be examined by an external verifier, though all their hardware characteristics are publicly known. The verifier can only interact with a black-box device through its interface, by providing inputs and receiving outputs [17]. For instance, access to the memory of a black box by *any* external program requires code execution inside the black box. Typical black boxes include a variety of device controllers such as network-interface controllers, baseboard management controllers, disk controllers, and USB device controllers. Protected subsystems of endpoint devices (e.g., kernels) can become black boxes for remote servers that scan devices' configurations and monitor their communications.

An unjustifiable property. Security properties of computations performed inside a black-box device exist which cannot be verified or tested unconditionally with certainty or externally monitored; e.g., the property of *absence of malicious software* (aka., malware) in a device controller's firmware. Malware can prevent direct memory access since it mediates all device transfers. Naive attempts to remove it by re-flashing device firmware by an OS kernel driver do not work because malware can hide in firmware areas that do not get updated or disingenuously respond with a prepackaged message; e.g., "update complete" or "already the latest version" [18].

Black-box devices can only be

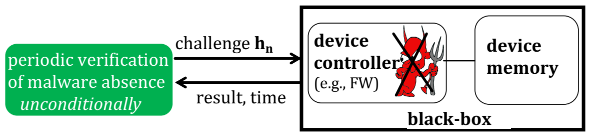
a) externally verified to detect malware absence; and if malware is not detected, they can be invoked repeatedly until the next verification; and

b) continuously monitored to detect malware presence in finite time; e.g., until the next verification.

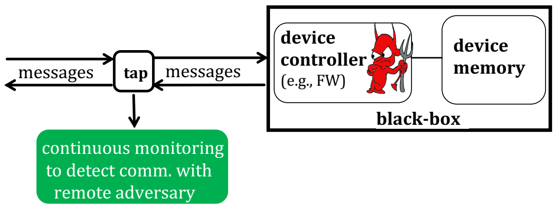
Figure 2 illustrates the impossibility of either alternative. In case a), necessary conditions for unconditional malware absence in device firmware show that the best result that can be obtained is probabilistic [19]. Any demonstrably correct challenge-response verification *falsely* assures that device firmware is malware free with probability at least $p > 0$. The intuition is simple: any malware can always guess the correct

[3]Without secrets, hardware trusted modules, or adversary-power bounds.

[4]Black-box testing/monitoring must be in finite time [17]; otherwise, it could never reach a non-asymptotic conclusion in practice.


periodic verification of malware absence *unconditionally*
challenge h_n
result, time
device controller (e.g., FW)
device memory
black-box

a) ***certainty***: **Pr[*false malware absence*** at **n**-th independent challenge h_n] ≥ p^n


messages
tap
messages
device controller (e.g., FW)
device memory
black-box
continuous monitoring to detect comm. with remote adversary

b) ***certainty***: *communication is covert, context dependent, or inexistent*

problem: *unknown malware has asymmetric advantage over all AI/ML algorithms*

Fig. 2: Impossibility of establishing malware absence

result to be be returned in response to challenge h_i in the verifier's expected *time* with *some* lower-bound probability $p > 0$. Since verification can be repeated n times independently, the probability of *false malware absence* cannot be lower than p^n, and thus it cannot reach *zero* except asymptotically in n; see Figure 2-a. Hence, in black-box devices, the malware-absence property can never be justified with probability 1 in finite time; i.e., $1 - p^n \to 1$ only asymptotically in n. (In contrast, periodic verification yielding high, but possibly different from 1, probability of malware freedom is implied by root of *trust establishment* [19], [20].)

In case b), we note that some malware may not have any (hyper) properties [21] that can be detected by an external observer with certainty. Since this malware has *no externally detectable* properties, the best method to discover its presence in a black-box device is to place a non-intrusive, out-of-band network tap external to the device host, and attempt to detect malware communication with its remote controller by continuous monitoring [22]; see Figure 2-b. This type of monitoring has been called the adversary's "worst nightmare" [23]. Unfortunately, the worst nightmare turns out to be insufficient: use of the most advanced ML/AI monitoring algorithms whose design and implementation are demonstrably correct may still fail to detect malware presence *with certainty*. Why? An adversary who inserts malware into a device has an asymmetric advantage over monitor designers: s/he knows all the properties of the monitor's ML/AI algorithms whereas the algorithm designers cannot rely on any externally visible malware property, including the certainty of communication with its remote controller. Hence, the adversary's malware can circumvent all (finite-time) detection algorithms with non-zero probability.

For instance, covert (e.g., steganographic) communication between malware and its remote controller cannot be detected by an external monitor with certainty, since it can take place infrequently, at unpredictable times, and disguised as legitimate communication. Furthermore, malware activation by its remote controller can be context dependent. Malware in USB devices can be activated in the context of specific hosts when the device is connected to them and remains inactive, and hence undetectable, when connected to all other hosts. This has enabled USB micro-controller malware to breach air gaps [24]. Also, malware activation can be time-triggered, which avoids *all* remote communications with its controller and renders monitoring useless.

Endpoint software such as OS, security, separation, and micro kernels become black boxes after boot, since only their interface can be accessed by remote external monitors. Continuous monitoring of endpoint communication using the best ML/AI algorithms *after* legitimate connection setup, as recently advocated by *Zero Trust* 2.0 [25], can detect a variety of threats. However, they cannot detect the presence of *all* malware *with certainty*, as argued above.

Early warnings. Early warnings implying the impossibility of achieving *zero trust* seem to have been ignored by zero-trust advocates. For example, more than a decade ago, in a deliberately provocative conjecture, Lampson [26] stated that security is *fractal*: "each part is as complex as the whole, and there are always more things to worry about." If security is fractal, clearly zero trust cannot be achieved. Schneider reinforced this conjecture by stating that trust can only be relocated [27]. Damgård observed that assurance obtained by reduction proofs in cryptography [28] only relocate trust to unproven conjectures, and Gollmann [29] re-emphasized this by noting that security reductions often assume that something unknown cannot happen. All imply that unproven security properties prevents *zero trust*, which must be unconditional.

B. *Impracticality*

As illustrated in Figure 1, zero trust requires that *all* security properties of *all* network devices (other than "black boxes") must be justified with *high assurance*; i.e., using sound definitions and formal proofs of code correctness. The long-standing "axioms" of insecurity [30] imply that there will always be only low assurance for *some* security properties of *some* commodity software that yields unjustified, insufficiently, or poorly justified beliefs, and there will always be adversaries to take advantage of them. That is, rapid innovation in the commodity software market (enabled by near-zero cost of entry, no liability, and hardly any regulation) often eschews use of costlier high-assurance methods in favor of developing new functions to meet market demand.

Zero trust requires high assurance since it must always prevent security-property violations by adversaries; e.g., system and network software must be penetration free and can never be breached. This is equivalent to reaching perfection, which is always impractical in commodity software [26]. However, in the real world, rational defenders accept low assurance and inevitable security-property breaches by adversaries who could not be deterred, and focus on reducing breach-recovery costs [31]. Hence, weak belief justification by low assurance always precludes achieving *zero trust*. It follows that eschewing high assurance and its higher opportunity cost precludes achieving minimum trust, and hence *zero trust*, as illustrated intuitively on the vertical axis of Figure 1.

C. *Zero trust is an unreachable limit of trust establishment*

Trust establishment in practice. Why is trust establishment necessary in practice? Trust establishment fuels electronic commerce and safe social protocols over computer networks [15]. If zero trust were required instead of trust establishment, *all* these protocols would fail in a finite (e.g., small) number of steps since zero trust is unachievable in any of these protocols *unconditionally* and *with certainty*.

Why is trust establishment useful in practice? Trust establishment does not require any specific assurance technique for justifying beliefs in security properties in the presence of an adversary. Instead, it requires that unjustified, poorly justified, or insufficiently justified beliefs be compensated by risk decrease and deterrence increase, as deemed appropriate in specific network applications. This implies that it can allocate development and operation costs in a flexible manner. For example, instead of incurring uniformly high-assurance cost for all security properties of network components, trust establishment allows lower-assurance costs for some or all components' properties plus some additional risk-mitigation and deterrence-support costs for them. This allows cost-allocation trade-offs that are unavailable elsewhere.

A limit of trust establishment. Trust establishment is intended to counter liabilities caused by adversaries who can exploit security properties when zero trust cannot be achieved. For instance, *some* security properties of a network component cannot be assured using simple methods and tools, and hence only a few beliefs of trustworthiness can be justified. Since trust is *not* minimized in this case, it cannot be zero, and trust establishment becomes necessary to compensate for the missing evidence of trustworthiness; see Figure 1. Hence, if trust establishment is necessary for *some* security properties of a network component, the *zero trust* limit cannot be reached for an entire network. Conversely, if hypothetically the zero-trust limits could be reached for *all* security properties of *all* network components, trust establishment would be unnecessary: *all* beliefs in all security properties would be fully justified, there would be no residual risk or need for adversary deterrence, since all network components would be completely resistant to all adversary attacks. Since this is impossible in any network, it follows that zero trust has meaning only as an *unreachable limit* of trust establishment.

IV. Review of zero-trust architectures

The question of this report's title can be answered with certainty now: *there is no zero trust in "zero trust."* Hence what the US Government and industry call a *zero trust architecture* or a *zero trust model* [1], [2], [3] must be explained since it must have other virtues than zero trust.

Implicit Trust Zones. To understand the motivation for *zero-trust architectures*, one must recall the danger of exclusive reliance on single-perimeter protection in enterprise networks. A typical enterprise operates several internal networks, each comprising commodity devices, software, and applications at remote locations, has mobile users, and uses cloud services [1]. In network-perimeter protection, an external user's access to devices, services, and data objects located inside a network perimeter is verified upon every perimeter entry and never thereafter. Hence, a protection perimeter defines an *implicit trust zone*: upon perimeter entry, the user's computations (e.g., processes) become trusted to maintain object secrecy and integrity, by default. If an adversary can penetrate an implicit trust zone (e.g., by credentials theft, by exploiting incorrect perimeter-entry verification), the adversary's computations become automatically trusted, and can move "laterally" throughout the zone. This compromises the implicitly trusted zone and increases the complexity and cost of recovery.

Motivation. Zero-trust architectures eliminate exclusive reliance on any *single* perimeter by partitioning enterprise-network resources into *multiple* micro-segments, each comprising network devices, services, and data objects [1]. Each micro-segment represents an implicit trust zone that can be entered only after a separate verification check on the zone's identity and accessor's credentials (e.g., users' identities, roles, permissions, privileges, access levels) passes. Since a minimum set of credentials can be assigned to a zone's devices, services, and programs to accomplish their tasks, the zone can operate with least privileges. Furthermore, a zone's operations can be monitored continuously and its programs' behavioral patterns an be checked when granting access. Repeated partitioning of large implicit trust zones by micro-segmentation reaches a *minimum* when trust zones comprise single uniquely identified devices; see Figure 5-b of Section IV-A below.

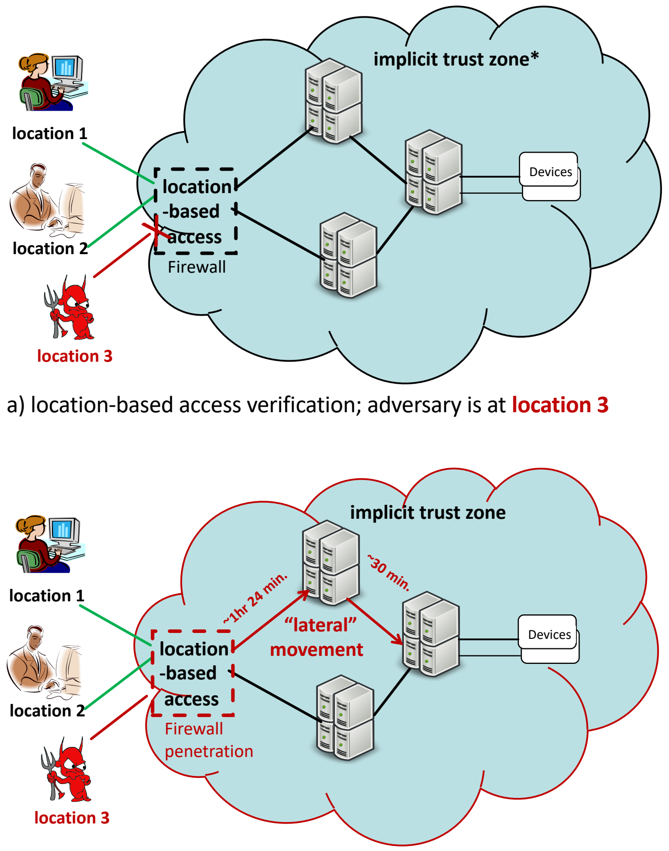
Main Goal. Zero-trust architectures partition a network into multiple trust zones and minimize them. Minimization aims to limit an adversary's "lateral" movement to a single penetrated zone. This can limit zone-penetration damage instead of preventing zone penetration, which would require costlier high-assurance methods for commodity software security.

Other goals. An additional goal is to detect security breaches early, by continuous monitoring of devises, systems, applications and communications of a network micro-segment (i.e., of an implicit trust zone), thereby decreasing the delay to penetration recovery. Equally important, is to maintain backward compatibility with existing (insecure) network, system, and application software, which enables timely deployment without major re-design and disruption.

A. *Key characteristics*

Although zero trust in unachievable in network security, it is important to understand what the US Government and industry label as a "zero-trust architecture" [1]. We give a brief but hypothetical example of network micro-segmentation into implicit trust zones and describe the conditions under which their minimization to single-device zones can be achieved.

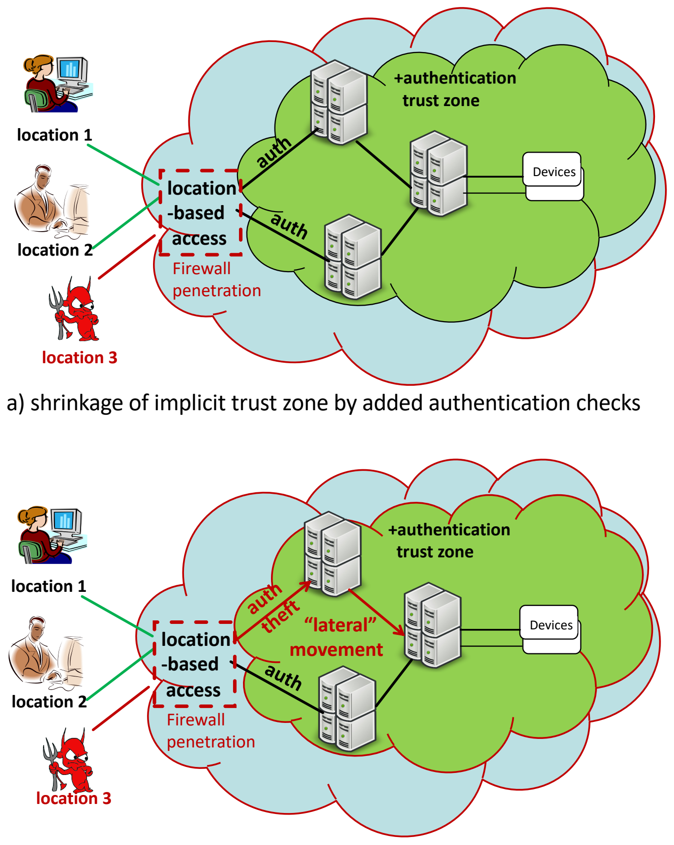
A network perimeter is protected by firewalls that check the location of users attempting access; see the *light-blue* zone of Figure 3-a. If an adversary at location 3 penetrates



a) location-based access verification; adversary is at location 3

b) lateral movement after perimeter penetration by adversary

Fig. 3: Implicit trust zone and lateral adversary movement



a) shrinkage of implicit trust zone by added authentication checks

b) adversary's (at location 3) lateral movement after credential theft

Fig. 4: Shrinkage of implicit trust zone by authentication checks and shrinkage failure after identity-credential theft

a perimeter firewall[5], the adversary's computation becomes automatically trusted and can move "laterally" throughout the *light-blue* zone, as illustrated in Figure 3-b. Recent industry reports show that, on the average, it takes 1 hour and 24 minutes for an adversary to move from the initial point of compromise to other servers within a zone, and that about a third of these attacks enables "lateral" movement in under 30 minutes [33]. As illustrated below, if additional verification is performed inside a zone to restrict objects/devices access, then the adversary's "lateral" movement can be limited.

Zero-trust architectures [1] aim to eliminate exclusive reliance on (e.g., location-based) protection perimeters, in order to limit an adversary's "lateral" movement in a compromised network. This requires micro-segmentation of a zone's resources (e.g., network devices, services, and data objects) to create smaller implicit trust zones, and granting access to resources based on:

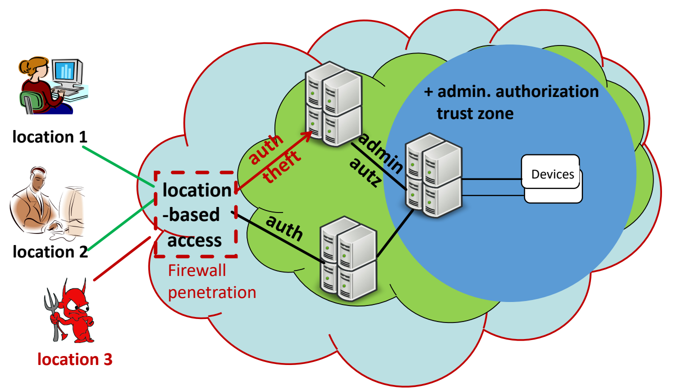
(i) continuous verification of a user's attributes, such as unique identity, credentials (e.g., roles, permissions, access levels), and monitoring behavioral patterns in granting access. This explicitly rejects verification based on *verify-once-trust-on-every-access* until the next verification implied by perimeter protection, in favor of the *never-trust-and-always-verify-every-access* approach.

(ii) enforcement of the least privilege principle, fail-safe defaults, and auditing a user's behavior after access [34]. This aims to limit the effect of perimeter penetration and prevent "lateral" movement of an adversary through the compromised network. The adversary is always *assumed to be the network* [5].

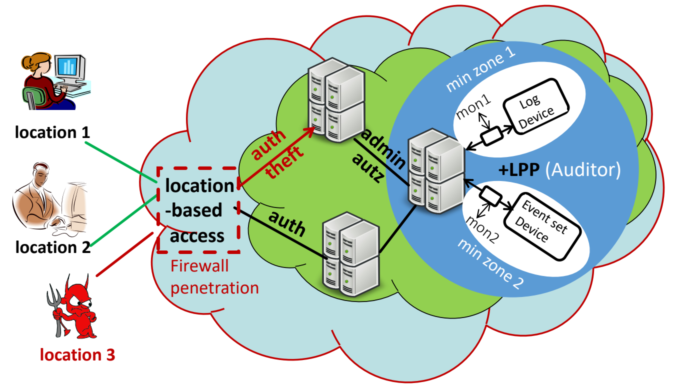
(iii) shrinkage of implicit trust zones by repeated application of (i) and (ii). Shrinkage reaches the minimum when the protection perimeter is a *single device*.

Zero-trust architectures recognize that any network micro-segmentation and corresponding verification of access attributes upon entry creates a new, smaller implicit trust zone. Figure 4-a illustrates a smaller implicit trust (*green*) zone created by the added authentication (*auth*) checks performed by the servers of the *light-blue* zone. Note that these servers can be accessed only after the location-based access check of the *light-blue* zone is passed. After penetrating the firewall, the adversary at location 3 must pass the authentication check of the *green* zone to access any objects in that zone. However, Figure 4-b shows that theft of one of the authentication (*auth*) credentials by the adversary at location 3 can re-enable "lateral" movement.

[5]The main reason for firewall penetration is misconfiguration, which is more prevalent than firewall OS, applications, and insider-attack vulnerabilities. Misconfiguration is often caused by *invisibility of endpoint IP addresses* and reliance on DHCP, and consequent *inability to automate* all firewall settings; e.g., ACL settings in remote virtual private clouds [32].



a) further shrinkage of trust zone by added admin. authorization checks



b) trust-zone minimization by using LPP principle and device monitoring

Fig. 5: Repeated trust-zone shrinkage until minimization using least privilege principle and device monitoring

Figure 5-a illustrates the shrinkage of the implicit authentication (*green*) trust zone by the addition of authorization (*admin autz*) checks for a network-administration zone whenever these checks *do not depend*[6] on the stolen authentication (*auth*) credential in Figure 4-b. This creates a new smaller trust (*blue*) zone and prevents "lateral" movement to it after firewall penetration *and* a single authentication credential theft.

Figure 5-b shows that additional enforcement of the least-privilege principle (LPP) can minimize the *blue* trust zone by shrinking it to *single-device* zones. For example, if these devices are accessible to an auditor, LPP implies that the privilege of reading a *Log Device* is granted only to the log-analysis application but not to the audit-event (re)setting application, and *Event-Set Device* access is granted only to the audit-event (re)setting application but not to the log-application. This means that cross-zone attacks between the audit-log and audit-event (re)setting applications can be prevented. Furthermore, continuous monitoring of these devices by *mon1* and *mon2* can detect attack patterns of a penetrated device against services of other external zones. Hence,"lateral" movement of an adversary that penetrated a minimized zone can be limited to that zone.

[6]Zero-trust architectures do not identify any type of functional or policy dependencies among verification checks [13]. For example, the "trust algorithms" recommended by NIST's zero-trust architecture [1], Section 3.3, fail to identify any such dependencies.

B. *Conditions for achieving the main goal*

Zero-trust architectures *cannot* and *do not* claim that implicit trust zones are resistant to penetration [5], [10].

The explanation of Figure 5-b shows that their main goal of limiting an adversary's "lateral" movement in a penetrated trust zone can be achieved by zone minimization
(i) *only if* continuous verification checks and application of least privilege principle (i.e., *location+ auth + autz + LPP*) *prevent* cross-zone attacks; and
(ii) *only if* continuous monitoring of devices' behavior in minimized zones *detects* cross-zone attacks.

If cross-zone attacks *cannot* be prevented or detected, then an adversary's "lateral" movement cannot be limited, and the main goal of zero-trust architectures cannot be achieved. Section V below illustrates this fact.

C. *Optimistic implementation assumptions*

Our hypothetical example of repeated micro-segmentation of Figures 4-a and 5 optimistically assumes that all user endpoint devices have unique identifiers, device locations are known, network topology and configuration – including high-value data objects – are easily determined, and all access to sensitive network APIs is accounted for, authenticated, and controlled. In short, assignment of the minimum set of permissions to each micro-segment is possible and, in principle, least privilege operation could be fully supported.

In practice, however, network micro-segmentation often fails for several reasons. First, many enterprises suffer from endpoint-device sprawl[7] which diminishes network-topology visibility [35]. Most user devices do not have unique identifiers, which makes control of their access to remote private clouds difficult to implement. Use of the dynamic host configuration protocol (DHCP) for remote bring-your-own devices further increases network insecurity [36]. Furthermore, incomplete location discovery and movement of high-value data objects in hybrid clouds [36] further complicates their assignment to individual micro-segments.

Second, large enterprises can use over ten thousand APIs to create new software services, applications, and platforms across their networks. However, few are able to discover and account for all APIs and when they do, fewer authenticate and authorize of all access to these APIs, leaving some exposed to external attacks[8]. Micro-segmentation aims to control access to APIs within implicit trust zones, yet few enterprises implement correct API access authentication and access control based on strong policies (e.g., role or attribute based access controls)

[7]A typical enterprise manages about 135,000 endpoint devices with nearly half (48%) of them remaining undetected on their networks [35].

[8]API attacks represent one of the most frequently used attack vectors with roughly 20% of enterprises surveyed [37] being breached during the past year. The recent attack against FBI's InfraGuard attack exfiltrated personal data of about 80K private sector stakeholders (see https://www.pcmag.com/news/fbis-infragard-us-critical-infrastructure-intelligence-portal-hacked) via an externally *exposed* API, whereas the breach against Optus communication service in Australia exploited an *unauthenticated* API. Weak authentication protocols allowing man-in-the-middle attacks to replay tokens/key to access APIs also contribute to trust zone compromise.

rather than merely using web application firewalls [37], [38]. Some also employ continuous external monitoring to detect – rather than prevent via high assurance methods – API breaches.

Third, enforcing least-privilege operation for micro-segments in large enterprises posses substantial technical challenges. In such an enterprise, ML-based tools must establish least-privilege micro-segments based on learned hardware and software configurations and applications' behavior in multiple networks. The learning phase of these tools may easily exceed a couple months; e.g., all software and applications' behavior must be learned before micro-segmentation can be performed and credentials/permissions assigned. At least two challenges could arise during this phase[9]. Configuration updates, patches, and business software releases in agile environments may take place, and these could cause continuous execution *without ending* this phase. Also, limited time learning may miss infrequently used enterprise software, such as year-end operations and once-a-year disaster recovery testing, and fail to assign its required permissions. This could cause this software to fail due to lack of permissions.

Absence of ML tool integration with Internet intelligence feeds may lead to learning malicious micro-segment connections to compromised third-party software suppliers and even to foreign servers controlled by adversaries. Malicious connections may be long lasting before discovery; e.g., over two-hundred eighty days on average [10]. Hence, micro-segmentation of large enterprise networks to achieve least privilege operation is often a coarse approximation of reality.

V. Zero-trust architectures are unsound

A. Unsoundness

Definition. An implication "**A** ⇒ **B**" is *unsound* if it is *invalid* or if the premise **A** does not hold in cases of interest. The implication is *invalid* if there exist cases when **A** is *true* and **B** isn't.

Recall that part (ii) of the zero-trust main goal discussed in Section IV-B represents the implication
"*minimization of a implicit trust zone limits lateral adversary movement only if continuous monitoring of devices' behavior in minimized zones detects cross-zone attacks.*"
This implication is invalid, and hence unsound, since there exist cases of minimal implicit trust zones (i.e., single devices) for which continuous monitoring cannot detect cross-zone attacks. For example, continuous monitoring fails to detect cross-zone attacks by some malicious firmware. In Section III-A case b) and Figure 2, we showed that continuous monitoring of a black-box device's behavior in a minimized zone cannot detect the presence of an adversary's malware in the device controller's firmware. Also, in **Example 2** below we show several cases where the adversary's malware can launch cross-zone attacks that are undetectable by external monitoring.

[9]Personal communication by Daniel Stieger of UBS, Zurich, on November 1, 2022.

Part (i) of the zero-trust main goal discussed in Section IV-B represents the implication
"*minimization of a implicit trust zone limits lateral adversary movement only if continuous verification checks for granting access and application of least privilege principle prevent cross-zone attacks.*"
In this section, we show that this implication is unsound since it is both invalid and its premise fails in cases of interest. In the first three examples below, we assume that the premise holds (i.e., all implicit trust zones can be minimized), and illustrate why the implication is invalid. The fourth and fifth examples show that the premise fails for critical trust zones, which cannot be minimized by zero-trust architectures. We illustrate unsoundness primarily with examples from NIST's special publication [1], though these examples apply to *all* zero-trust architectures.

Recall that unsoundness of the implications above means that the main goal of zero-trust architectures cannot be achieved; i.e., an adversary's damage cannot be limited to the (albeit minimized) implicit trust zone it penetrates. We stress that this does *not* mean that these architectures are useless for enterprise networks. However, it does mean that these architectures *cannot* serve as security models, as they fail to counter major security exposures, and *should not* be exclusively relied upon to protect any network for critical applications.

B. Five Examples of Unsoundness

Sandboxing. All zero-trust architectures support application "sandboxing" (aka., isolation) on a host using a variety of constructs, such as containers, enclaves, or isolated execution environments with hardware support[10]. Sandboxes protect applications from each other and other compromised applications, and represent a host's micro-segmentation into separate minimized trust zones. Hence, malware in one zone cannot infect the other zone. The basic problem is that cross-zone attacks between two minimized trust zones cannot be prevented when both depend on a large implicit trust zone that can be penetrated.

Example 1. *Dependency-enabled cross-zone attacks.*
Sandboxed applications use input-output functions, including network access and/or trusted path [39], which make them dependent on an I/O subsystem comprising millions of lines of code within an untrusted host OS. Figure 6-a illustrates an isolated sandbox 1 with exclusive access to I/O device 1 and an isolated sandbox 2 with exclusive access to device 2. The two sandboxed applications may contain separate drivers for their I/O devices; see Figure 6-b. In both cases, malware in one trust zone can manipulate the I/O hardware of the underlying host to breach the security of the other trust zone.

The common I/O vulnerabilities and attacks against isolated applications shown in Section II of reference [40], illustrate

[10]NIST's special publication [1], Section 3.2.4, Figure 6, page 17, illustrates two sandboxed applications in two isolated trust zones of an untrusted host. Both trust zones depend on a common host OS and host-device firmware.

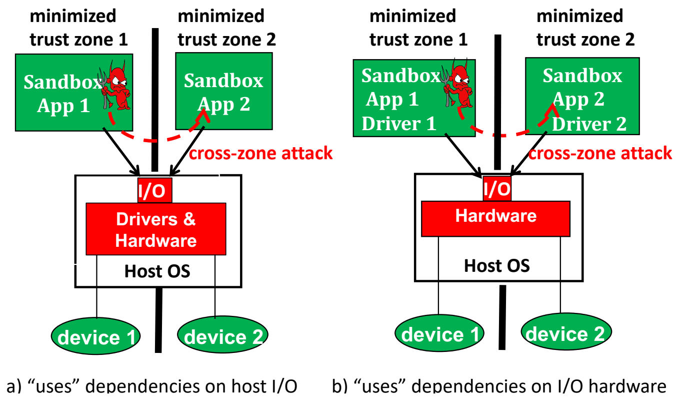


Fig. 6: Cross-zone attack enabled by lack of I/O separation

this simple case of unsoundness. For example, some I/O hardware fails to enforce exclusive association of I/O devices with their sandboxed application, and to selectively separate read-write permissions for I/O transfers. Other I/O hardware can selectively associate *only* I/O bus controllers with sandboxed applications but *not* individual devices, and separate read-write permissions *only* for bus transfers. Although the most advanced I/O hardware (e.g., IOMMU, PCIe, and ACS) can exclusively associate individual devices with their sandboxed applications and selectively authorize read-write transfers, their use frequently trades-off correctness assurance of I/O transfer separation for added performance, particularly in commodity OSes [40]. Vulnerabilities caused by malware manipulation of I/O hardware in a penetrated trust zone cause cross-zone attacks that easily lead to applications' compromise despite sandboxing strength.

Other cross-zone attacks, which are enabled by "uses" dependencies [41], appear between implicit trust zones that are *cyclicly dependent* even if these zones are minimized; see *Example 4* and Figure 9 below. Some of these attacks disappear after *cyclic dependencies* are removed; see Figure 10-b and c. ■

Bring-your-own-device (BYOD). All zero-trust architectures allow use of remote bring-your-own-device (BYOD) endpoints in an enterprise network [1], Section 2.2. These devices can represent minimized implicit trust zones if they have unique identifiers that cannot be corrupted (e.g., by malware in their firmware or OS) *and* persistent cryptographic channels to their external suppliers; e.g., for secure firmware (FW) updates and new unique-identifier provisioning on device-ownership changes, as shown in Figure 7. Assume these conditions can be satisfied by BYOD endpoints. The Trusted Computing Group (TCG) has shown how to provide a unique device identity [42] and maintain persistent cryptographic channels for updating device firmware, including the device-update engine itself [43], and later attest to the update completion [44], [45]. Microsoft's Cider system [46] and its underpinnings [47] illustrate how to implement TCG's specifications. For instance, they show how to establish a cryptographic channel from a BYOD firmware to its external supplier, though this must assume some local firmware provisioning[11]. Similar cryptographic channels can be established between uniquely identified OSes and applications and their external suppliers; see Figure 7.

Physical and insider attacks. A significant problem is that BYODs are subject to *physical* and *insider attacks* that can corrupt their firmware. For example, some USB devices allow an adversary's direct access to the firmware of the device's micro-controller [50]. An "evil innkeeper" can access a guest's device in the guest's absence, and a second-hand reseller has possession of a future owner's device before its delivery; both can access motherboard firmware and modify the unified extensible firmware interface [51]. A physical man-in-the-middle can interdict a device and modify its disk-controller firmware before the device is delivered to a legitimate owner [52], [53]. In all cases, device firmware can be re-provisioned with malicious code and public keys of the remote adversary's command-and-control servers. This illustrates Stajano's long-standing *big stick principle: "whoever has physical access to a device is allowed to take it over"* [54]. However, the recent Presidential Executive Order [2] assures us that "[i]f a device is compromised, zero trust can assure that the damage is contained;" see page 26646. The next two examples show that this is *not* the case even without any physical device compromise.

Example 2. *Cross-zone attacks by malicious firmware.* Assume that a fully minimized implicit trust zone comprises a single BYOD. Malicious BYOD firmware can subvert verification of access attributes when entering a trust zone, thereby illustrating a more advanced case of unsoundness. Examples of malicious-firmware use include advanced persistent threats (APTs), which allow *remote* adversaries to exploit the lack of device-firmware integrity to circumvent access controls without having to exploit any physical device access; e.g., see[12] APT28 (Fancy Bear with the *LoJax* attack), APT29 (Cozy Bear theft of Covid-19 research results), APT41 (*Double Dragon*'s large-scale espionage and the recent *MoonBounce* attack). How could BYOD firmware behave maliciously? Recall that non-malicious firmware updates by trusted suppliers [46], [47] do *not* guarantee absence of zero-day vulnerabilities in the updated firmware that can be exploited remotely. OS kernel code updates do not guarantee that security flaws do not exist among the millions of lines kernel code which can enable firmware-vulnerability exploits and malware insertion into

[11]Local provisioning of BYOD firmware is always necessary [19]. For example, suppose that a supplier attempts to deliver a device whose firmware is pre-provisioned with the supplier's cryptographic channel to a remote user. However, an adversary can interdict device delivery before it reaches the user, change the supplier's (e.g., public) cryptographic key, and – at the very least – enable a *cuckoo attack* [48]. This makes the device communicate securely with the adversary instead of the supplier's firmware-update server. Similarly, changes of device ownership [49] and key replacement after loss also requires local device-firmware provisioning.

[12]APT28: https://www.cyberscoop.com/lojax-russia-apt28-eset-firmware
APT29: https://www.cyberscoop.com/coronavirus-vaccine-hacking-cozy-bear-apt29
APT41: https://www.zdnet.com/article/chinese-apt-deploy-moonbounce-malware-in-uefi-firmware.

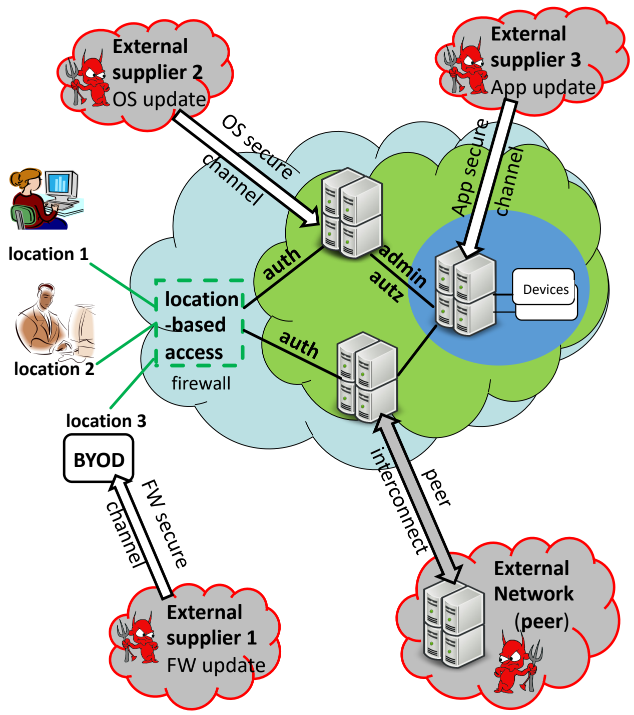


Fig. 7: Cross-zone attacks by compromised external suppliers and peer networks

firmware. Also, continuous external monitoring of BYODs can fail to detect malware presence (as shown in Section III) and thus cannot prevent possible cross-zone attacks. ■

Example 3. *Expanded locus of external cross-zone attack.* Assume that each autonomously administered external (e.g., supplier, peer) network implements a zero-trust architecture with minimized trust zones; see the four (gray) minimized trust zones of Figure 7. Compromised suppliers' zones can deliver malware to minimized zones of a customer network illustrating unsoundness without customer recourse. For example, in the *ShadowHammer* (2019) attack, an adversary compromised a server running ASUS' *live update* service and obtained a valid signature on a malicious firmware update[13] for ASUS' customers. Supply-chain attacks can scale in the number of compromised suppliers. For instance, a recent ethical hack against supply chains breached over thirty-five suppliers[14].

Zero-trust architectures also allow *pairwise interconnection* of autonomously administered networks; e.g., see Section 4.4 of reference [1]. These networks may be connected as *peers*, as illustrated in Figure 7, allowing their local users and services to access remote objects and services located in their peer networks. Authentication trust in peer-interconnected networks [55] expands an implicit trust zone with the administrative zones of all its peers and some of these peers' networks. An adversary who impersonates a legitimate user of an external network by compromising the user's credentials can launch a cross-zone attack against a peer network. The only recourse available to a peer network is to lower *all* remote users' (and hence adversary's) permissions to its local objects and services proactively, since it cannot distinguish legitimate from compromised remote users [1]. However, this cannot prevent *new* cross-zone attacks, since the adversary impersonating a legitimate user will get *some* illegitimate access to objects and services of the peer network, illustrating unsoundness. Lowered permissions will also deny *some* remote legitimate users' access in the peer network illustrating incompleteness.

How large is the locus-of-attack expansion? When all external network configurations are known, the expansion is linear in the number of external suppliers, and quadratic in the number of pairwise-interconnected peers[15]. ■

Administrative perimeter. All enterprise networks require the protection of an administrative network perimeter that sets security policies and initializes access control decisions. For example, Section 3, page 4, Figure 2, of the NIST's zero-trust architecture [1] recognizes the need for an administrative network perimeter in all network security decisions; viz., its policy decision point. However, many network architectures fail to recognize the critical importance of minimizing this perimeter's large implicit trust zone: penetration of any administrative perimeter allows unfettered "lateral" adversary access to network resources without recourse. For example, past penetrations of poorly protected MS Windows administrative perimeter (e.g., by exploiting the long-time vulnerable "pass the hash" mechanism[16]) fully compromised US Office of Personnel Management and other government sensitive databases in 2013. The *Solar Winds* attack of 2019 has vividly illustrated the magnitude of the security problem faced by not minimizing administrative network perimeters.

Example 4. *Failure to minimize critical administrative zones.*

Function separation is not micro-segmentation. Past separation of administrative functions into implicit trust zones distinguishes security administrator and auditor functions and places them in a checks-and-balances relationship, which minimizes trust [56], [57]. This differs from micro-segmentation of administrative functions since enforcement of the least-privilege principle is insufficient for implementing checks-and-balances relationships or other Separation of Duty policies (SoD) [58]; see Figure 8. Although this principle helps minimize trust zones (see auditor example of Section IV), it isn't always necessary for administrative-function separation.

Administrative-function separation creates "uses" dependencies [41] among *security functions and policies*[17]. For instance, system-administrator functions are separated from certificate authority functions, which are partitioned into separate

[13]https://eclypsium.com/2019/04/23/shadowhammer-and-the-firmware-supply-chain/

[14]Elisabeth Montalbano. Supply-Chain Hack Breaches 35 Companies Including PayPal, Microsoft, Apple. In *TreatPost*, Feb. 2021. https://threatpost.com/supply-chain-hack-paypal-microsoft-apple/163814/

[15]In Section IV-C we optimistically assumed that micro-segmentation can address endpoint-device and API sprawl. However, if a *peer* network includes unaccounted for devices and APIs, their external suppliers are unknown, and defense against *all* supply-chain attacks [2], [3] becomes impossible.

[16]Microsoft's "pass-the-hash" mechanism predates the LanMan 1.4 design in 1991. Microsoft finally deprecated this mechanism over two decades later.

[17]See their definitions in the *Common Criteria* [13].

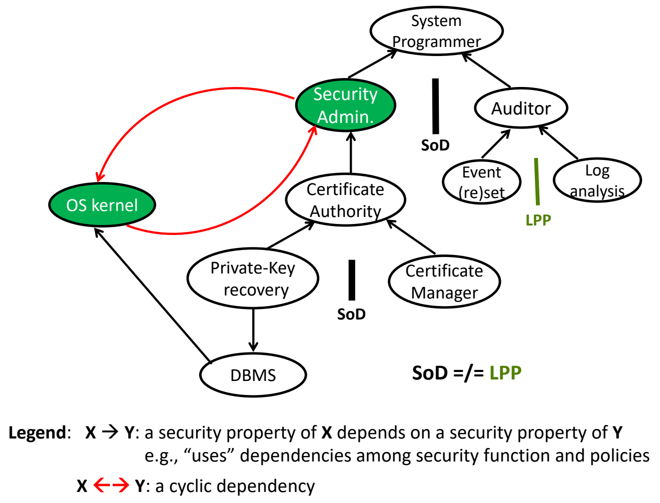

System Programmer
Security Admin.
SoD
Auditor
OS kernel
Certificate Authority
Event (re)set
Log analysis
LPP
Private-Key recovery
SoD
Certificate Manager
DBMS
SoD =/= LPP
Legend: X → Y: a security property of X depends on a security property of Y
e.g., "uses" dependencies among security function and policies
X ↔ Y: a cyclic dependency

Fig. 8: Dependency graph among administrative zones

certificate manager and private-key recovery functions. This separation establishes "uses" dependencies of certificate manager and private-key recovery functions on the certificate authority and of the certificate authority on security administrator functions. The private-key recovery functions also depend on a database management system (DBMS) for private key archival and recovery, which in turn depends on the system's OS-kernel for setting its permissions. When the source code implementing these functions is compiled/assembled/linked, a new administrative function – a trusted systems programmer [56], [57] – is required to assure that binary tools are malware free[18]. All other administrative functions depend on system programmer functions. Since each separate administrative function defines an implicit trust zone, a security function and policy *dependency graph* is defined on these zones, as shown in Figure 8.

Trust minimization on dependencies graphs. Administrative trust-zone minimization has to be performed on the dependency graphs. However, neither trust-zone separation nor minimization on dependency graphs are defined by any zero-trust architectures, including NIST's [1]. Hence, the implicit-trust-zone-minimization premise of zero-trust architectures cannot hold for critical administrative zones.

Cyclic-dependency removal. An additional complexity introduced by trust minimization on dependency graphs is that dependencies can be cyclic; e.g., a security property of trust zone A depends on a security property of trust zone B and (perhaps another) security property of B depends on some security property of A; see Figure 9-a. For instance, a cyclic dependency arises between an implicitly trusted SA zone and a server's OS kernel zone: the OS kernel depends on SA for the settings of its permissions and SA depends on the OS kernel for enforcing its permissions; see Figure 8. Also, note that a single-zone penetration can compromise both zones A and B; see Figure 9-b.

[18]Systems programmer can use diverse double compilation [59] and simple binary-comparison tools to detect malware-compromised compiler, assembler, or linker code [60].

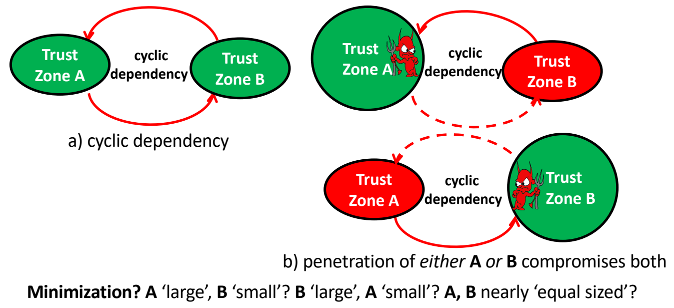


Fig. 9: Cyclic dependency and zone-crossing attacks

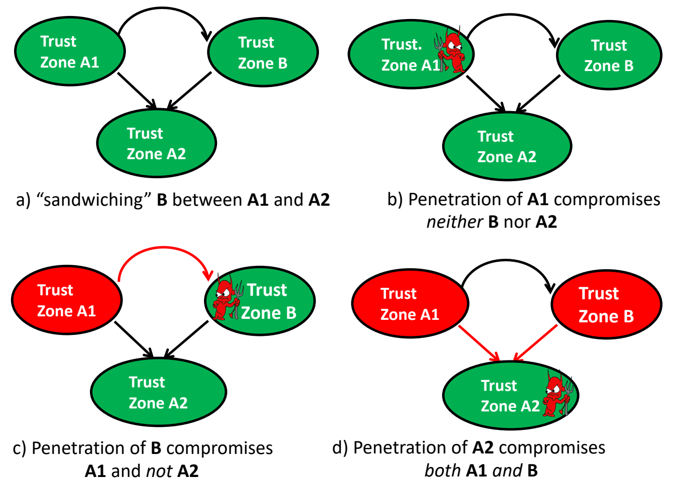

Trust Zone A1
Trust Zone B
Trust Zone A2
a) "sandwiching" B between A1 and A2
Trust. Zone A1
Trust Zone B
Trust Zone A2
b) Penetration of A1 compromises neither B nor A2
Trust Zone A1
Trust Zone B
Trust Zone A2
c) Penetration of B compromises A1 and not A2
Trust Zone A1
Trust Zone B
Trust Zone A2
d) Penetration of A2 compromises both A1 and B

Fig. 10: Cyclic-dependency removal & cross-zone attacks

Removal of cyclic dependencies is important because it can decrease the number of cross-zone attacks; see Figure 10-b and c. This is accomplished by re-design that converts dependency *cycles* into *directed acyclic graphs* of dependencies. Classic software engineering techniques, such as "sandwiching,"[19] have been used to re-design services and convert dependency cycles into *directed acyclic dependency graphs*; as illustrated in Figure 10-a. Whether implicit trust zones are minimized or not, re-designing trust zones to convert cyclic trust-zone dependencies into directed acyclic graphs can prevent some cross-zone attacks; see Figure 10-b and c, but not d. Hence, such conversion decreases the number of cross-zone attacks.

We note that zero-trust architecture, including NIST's [1], lack criteria for both minimization of cyclically dependent trust zones and cyclic dependency removal, further illustrating why the implicit-trust-zone-minimization premise can fail. ■

Example 5. *Failure to minimize critical application zones*. Zero-trust architectures fail to recognize that many implicit trust zones of network applications require minimization principles that differ from the least privilege principle of *zero*

[19]Parnas [41] introduced "sandwiching" to convert cyclic dependencies into directed acyclic graphs in software design. Whenever a module A depends on module B and B depends on A, module A can be redesigned to comprise two sides of a "sandwich," modules A1 and A2, such that A1 depends on A2 and B, and B depends on A2; see Figure 10-a. Sandwiching has been used in secure system design beginning with the Multics kernel design project [61].

trust. For example, traditional business applications define implicit trust zones and minimize trust by implementing SoD principles and policies [58]; e.g., by implementing checks-and-balances between implicit trust zones, multi-zone agreement for taking critical decisions, and exclusion of trust zones to avoid conflicts of interest. Other applications minimize trust by performing secure multi-zone computations in face of active adversaries. The trust-zone minimization premise of zero-trust architectures fails for many business applications. ■

It is unsurprising that zero-trust architectures cannot minimize critical application zones when minimization principles required differ from those of zero-trust architectures. However, it is surprising that the Presidential Executive Order and its Office of Management and Budget (OMB) implementation [2], [3] miss requiring these principles since they have been widely used in government applications for many decades; e.g., SoD principles and policies. However, they emphasize *zero trust* use of only lower-level access controls, such as role- and attribute-based access controls, which do *not* require SoD use, for instance.

VI. Zero-trust architectures are inadequate for pervasive government use

The question of whether zero-trust architectures are adequate for pervasive use in network security arises naturally. In particular, can these architectures satisfy the requirements of the recent Presidential Executive Order 14028 [2]? This document (i.e., Section 10, paragraph (k)) relies on NIST's zero-trust architecture [1], calls it a "security model," and mandates its adoption for government (see page 26636). The accompanying OMB memorandum M-22-09 elaborates the adoption requirements by the end of FY 2024 [3]. While a complete analysis of which requirements of the Executive Order and OMB memorandum can be supported or are precluded is beyond the scope of this report, it is clear that zero-trust architectures are inadequate for the government mandate, for two reasons. They fail to fully satisfy the government mandate, and hence new requirements must be introduced; and they preclude implementing some others requirements of the mandate. We illustrate a few of these inadequacies.

The following five requirements *cannot be satisfied*.

Requirement 1: *isolate computing environments*.

Zero-trust enforcement implies that computing environments can be isolated only if they are in different implicit trust zone. *Examples 1) – 3)* above show that even if different implicit trust zones are minimized they cannot prevent cross-zone attacks that violate computing environment isolation.

Requirement 2: *if a device is compromised, zero trust can assure that the damage is contained.*

Zero-trust enforcement implies that, to contain the damage caused by a device compromise, the implicit trust zone of a device must be fully minimized; i.e., it must contain that single device and nothing else. However, *Examples 2* and *3* above show that cross-zone attacks can be launched by compromised devices even if their trust zones are fully minimized.

Requirement 3: *understand devices' operation and their security posture when granting access to resources.*

Zero-trust enforcement can only guarantee that a device belongs to its fully minimized trust zone. However, device operation and its security posture cannot be known and understood when granting access to resources on that device, since the device's firmware could have been compromised by undetected malware. *Examples 2* and *3* above show that malware freedom of device firmware cannot be established by implicit trust zone minimization.

Requirement 4: *audit of trust relationships.*

Zero-trust enforcement cannot audit any trust relationships beyond those implied by implicit trust zone minimization; see Section IV. Other relationships among critical trust zones cannot be identified and hence cannot be audited; e,g., *Examples 4* and *5* above show that "uses" dependencies that are essential to minimizing critical trust zones are undefined and hence are non-auditable.

Requirement 5: *mandate that users log in directly to applications to enable a bare minimum of access needed to perform their job.*

Other than prevention single sign-on to multiple applications of an enterprise, which may be a user inconvenience, this mandate fails to satisfy its intended purpose of least-privilege access. Although zero-trust enforcement can guarantee direct application login by users (e.g., via multi-factor authentication), it cannot "enable a bare minimum of access needed to perform their job." This is because it fails to support *trusted path* to applications *after* login. That is, a remote adversary can attack a user's endpoint system *after* the user's secure application login, and surreptitiously modify application commands to corrupt user inputs and display unauthentic outputs. This can trigger unnecessary (e.g., maximum) users' access while performing their job, despite secure application login. Moreover, *without trusted path there is no accountability* of "who, what, when, and where" performed an access [2].

It is astonishing that all zero-trust architectures, including NIST's, and the Presidential Executive Order [2], [3] miss the fundamental need for *trusted path* after four decades since its introduction in US Government security standards[20] and enforcement in past commercial-product evaluations.

Enforcing zero-trust *precludes* implementing the following three *trust-establishment* requirements (i.e., 6 – 8 below), among others. The basic reason for this is that trust establishment *excludes* zero trust; see Section III-C.

Requirement 6: *establish "security and integrity of 'critical software' – software that performs functions critical to trust."*

Zero-trust enforcement implies that functions critical to trust must be in some implicit trust zones. Even if trust zone isolation were possible, the security and integrity of functions within a zone cannot possibly be (e.g., formally) proven once and trusted afterwards, since this is ruled out

[20]*Trusted path* was introduced by the NSA for commercial systems evaluations in 1983 [12]. Lack of *trusted path* has legitimately been called the "ultimate insult" to end-to-end network security [39], [62].

by the continuous verification requirement (i) of Section IV. The approach of (formally) *verify-once-and-trust-on-every-access* afterwards, which has been used to assure security-, separation-, and micro-kernels for decades, is explicitly rejected by zero-trust architectures[21]. This rejection is further emphasized by the justified assumption that any trust zone is penetrable. Thus, an adversary can penetrate any trust zone that encapsulates functions critical to trust and corrupt the security and integrity of "critical software" implementing them.

Requirement 7: *establish "trusted source code supply chains."*

Zero-trust enforcement in an enterprise network cannot establish that any of the enterprise's external supply chains are trusted. This is clearly a trust-establishment requirement which *excludes* zero trust.

Requirement 8: *ensure and attest, to the extent practicable, to the "integrity and provenance of open-source software."*

Zero-trust enforcement in an enterprise network rules out satisfying this requirement. This is because ensuring the integrity of open-source software requires source-code (e.g., formal) proofs, and provenance attestation requires signature-verification checks that must be isolated in a trust zone *and* (e.g., formally) proven; see similar requirement for attesting firmware-update provenance in Appendix B. However, both requirements imply correctness assurance of some software content of an implicit trust zone, which is explicitly ruled out by zero-trust implementations; see *Requirement 6* above.

Despite other inadequacies that are similar to the eight noted above, the Presidential Executive Order [2] and OMB memorandum [3] mandate *zero trust*, which excludes *trust establishment*. Nevertheless, these documents attempt to remedy some *zero-trust* inadequacies while missing many others, some of which are identified herein. These inadequacies are also relevant to non-government enterprises.

VII. Low defense and good recovery value

A. Addressing known weaknesses

Zero-trust initiatives have addressed some glaring weaknesses of enterprise-network security. These include user identification and authentication via multi-factor authentication, removal of single corporate VPN perimeters and replacement with tailored remote desktop protocols (e.g., of virtual desktop infrastructures) for access to specific corporate resources. Protection perimeters and their firewalls are removed, network zones are merged and then micro-segmented to define implicit trust zones with decreased, least-privilege access. All these improvements have been known long before the *zero trust* buzzword was coined, and yet their belated deployment has *not* prevented many common security breaches, much less advanced persistent threats.

B. Low but non-zero defense

A recent IBM-Ponemon Institute survey [10] examined 537 breaches in 17 countries and 17 world regions, covering

[21]In contrast, trust establishment (Section II-B), which excludes zero trust, relies on different levels of assurance for security properties.

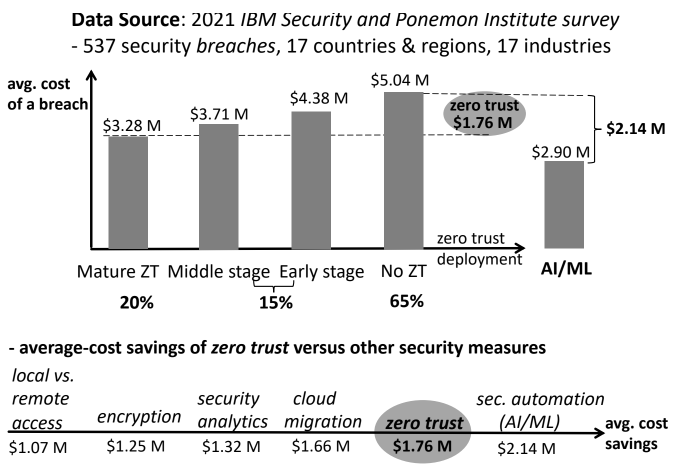


Fig. 11: Reduced cost of recovery from security breaches

17 major industries. They ranged from the most common (e.g., exploits of compromised credentials, phishing, cloud misconfiguration, and vulnerable third-party software) to the most costly; e.g., email compromises, insider attacks, social engineering attacks and data/device losses. *None* were prevented by mature *zero-trust* implementations and their average recovery costs were substantial; i.e., $3.28 M average cost per breach; see Figure 11. The consequences of these findings are predictable. Much like an incomplete patch to a known software vulnerability that does not cover some possible exploits, zero-trust architectures encourage an adversary to take advantage of vulnerabilities they fail to remove.

We stress that zero-trust architectures do *not* claim to and *cannot* provide assurance of penetration resistance for any implicit trust zones even if these zones are correctly defined and minimized, as optimistically assumed in Section IV-C. This suggests that future zero-trust architectures will continue to offer only low-security value. Nevertheless zero-trust initiatives have increased industry security awareness and about 83% of security and risk professionals agree that zero-trust architecture is essential to their organizations [6]. This is unsurprising given the elevation of the *zero-trust* buzzword to an industry and government slogan: a Google search on the "zero trust" phrase identified millions of citations. However, the projected investments in zero-trust initiatives appear to paint a more realistic picture. The estimated investment in these initiatives by 2025 is $1.6B compared to $233B of projected expenditures in information security and risk management [6]. Thus, the zero-trust investment estimate represents a very small fraction (i.e., 0.68%) of these expenditures, reflecting the limited value of the *zero-trust* initiatives.

C. Reduced recovery cost after network compromise

According to the same IBM-Ponemon Institute survey [10], the significant advantage of zero-trust deployment is in reducing recovery costs from security breaches. For example, Figure 11 shows that the average cost of a security breach after "mature" zero-trust deployment (i.e., $3.28M) is much lower than those incurred when zero trust is not deployed at all (i.e., $5.04M). However, this average-cost reduction (i.e.,

\$1.76M) is significantly lower than that obtained by security automation and artificial intelligence (AI)/machine learning (ML) measures in containing security incidents and intrusion attempts (i.e., \$2.14M). Although this cost reduction is only marginally higher than migrating to hybrid-cloud security (\$1.66M), it is substantially higher than that obtained by using standard security analytics (i.e., \$1.32M) and encryption (\$1.25M). Also, the recovery-cost savings of mature zero trust are much higher than those realized by local-only network access; i.e., the cost reduction of local access is \$1.07M higher than than of remote access.

VIII. Summary

With millions Google Search references to date, *zero trust* has become a government and industry slogan, which conveys an achievable security goal. However, scrutiny of trust and zero trust definitions shows that "zero trust architectures" have nothing to do with the unachievable notion of *zero trust* in security. Ignoring its hyperbolic use of the *zero trust* qualifier, a zero-trust architecture is an improvement over the "no defense" state of network security before its introduction. However, the improvement is relatively small. All available evidence shows that zero-trust architectures have low defense value, and their goal of limiting effects of inevitable security compromise is often unmet. Admittedly these architectures do *not* offer *any* correctness evidence for their verification and monitoring claims, which naturally lowers development and operational costs. Zero-trust architectures could never serve as *security models* as they fail to meet their mail goal and inadequate for pervasive use in critical networks, not just government's. Nevertheless, they can reduce recovery costs after inevitable network compromises.

Mandating adoption of these architectures in *government* networks seems surprising, particularly since trust establishment measures, which exclude zero trust, are required to compensate for some basic *zero trust* inadequacies, while missing the ones identified here. However, mandating these architectures has some practical advantages. They maintain backward compatibility with existing (insecure) network, system, and application software, which is intended facilitate deployment without major disruption. They rely on continuous monitoring of implicitly trusted zones aiming to detect security breaches early thereby decreasing the delay to begin recovery.

Adoption of zero-trust architectures reinforces a past market preference: low-assurance networks that incur *recurrent* recovery costs after multiple penetrations have been commercially preferable to high-assurance alternatives that could incur *one-time* high development costs but prevent most penetrations. This may change in the future if trust establishment measures are adopted since they provides a better cost-benefit balance. Trust establishment could decrease one-time development costs by applying high assurance only to *selected* components while adding some recurrent risk-mitigation and deterrence costs. This would allow more flexible cost allocation.

Acknowledgments

Earlier versions of this report have received sympathetic readings and useful suggestions from David Basin, Mads Dam, Kevin Foltz, Tim Grance, Brent Kang, Rick Kuhn, Adrian Perrig, Vyas Sekar, William Simpson, and Nickolai Zeldovich. Whatever clarity this report has is due in no small measure to comments received from Frank Stajano, Bryan Parno, and Miao Yu. However, responsibility for all its content rests with the author.

References

[1] National Institute of Standards and Technology, "Zero Trust Architecture," 2020.

[2] White House, "Executive Order No. 14028 – Improving the Nation's Cybersecurity," May 2021. [Online]. Available: https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[3] S. Young, "Moving the U.S. Government Toward Zero Trust Cybersecurity Principles," Jan 2022. [Online]. Available: https://www.whitehouse.gov/wp-content/uploads/2022/01/M-22-09.pdf

[4] Department of Defense, "DoD Zero Trust Reference Architecture," Tech. Rep., March 2021. [Online]. Available: https://dodcio.defense.gov/Portals/0/Documents/Library/(U)ZT_RA_v2.0(U)_Sep22.pdf

[5] National Security Agency, "Embracing a Zero Trust Security Model," 2021. [Online]. Available: https://media.defense.gov/2021/Feb/25/2002588479/-1/1/0/CSI_EMBRACING_ZT_SECURITY_MODEL_UOO115131-21.PDF

[6] L. Columbus, "Zero-Trust Trends for 2022," in *VentureBeat*, January 20, 2022. [Online]. Available: https://venturebeat.com/2022/01/20/zero-trust-trends-for-2022

[7] ——, "How cybersecurity vendors are misrepresenting zero trust," in *VentureBeat*, August 22, 2022. [Online]. Available: https://venturebeat.com/security/how-cybersecurity-vendors-are-misrepresenting-zero-trust/

[8] Z. Malekos, E. Lostri, and J. Lewis, "The hidden costs of cybercrime," Dec 2020. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-hidden-costs-of-cybercrime.pdf

[9] VentureBeat Staff, "Report: US businesses experience 42 cyberattacks per year." Sept. 20 2022. [Online]. Available: https://venturebeat.com/security/report-u-s-businesses-experience-42-cyberattacks-per-year/

[10] IBM Security and Ponemon Institute, "Cost of a data breach report," 2022. [Online]. Available: https://www.ibm.com/security/data-breach

[11] V. D. Gligor, "Dancing with the adversary: A tale of wimps and giants (article and transcript of discussion)," in *Proc. of the 2014 Security Protocols Workshop, Cambridge, UK*, no. LNCS 8809. Springer, 2014.

[12] NSA, National Computer Security Center, "Trusted computer system evaluation criteria (The Orange Book)," 1983, DoD 5200.28-STD. [Online]. Available: http://csrc.nist.gov/publications/history/dod85.pdf

[13] Common Criteria, "Common criteria for information technology security evaluation part 2: Security functional components," pp. 1–321, 2009. [Online]. Available: https://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R3-markedchanges.pdf

[14] V. Gligor and J. Wing, "On the foundations of trust in networks of humans and computers (also see transcript of discussion)," in *Proc. of the 2011 Security Protocols Workshop, Cambridge, UK*, no. LNCS 7114. Springer, 2012.

[15] V. D. Gligor and F. Stajano, "Assuring safety of asymmetric social protocols (also see transcript of discussion)," in *Proc. of the 2017 Security Protocols Workshop, Cambridge, UK*, no. LNCS 10476. Springer, 2017.

[16] E. Fehr, "The economics and biology of trust," *Journal of the European Economics Association*, vol. 7, April-May 2009.

[17] M. T. Dashti and D. A. Basin, "Tests and Refutation," in *Proc. of Automated Technology for Verification and Analysis - ATVA*, ser. LNCS 10482. Springer, 2017.

[18] C. Raiu, "Commentary in Equation: The Death Star of the Malware Galaxy," in *Kaspersky Lab*, Feb 2015. [Online]. Available: https://securelist.com/equation-the-death-star-of-malware-galaxy/68750/

[19] V. Gligor, "What's Necessary to Establish Malware Freedom Unconditionally? (long version)," in *Presentation at the IEEE Foundations of Computer Systems Workshop, Boston, Mass.*, June 2020.

[20] V. D. Gligor and M. Woo, "Establishing Software Root of Trust Unconditionally," in *Proceedings of the NDSS, San Diego, CA*. Springer, Feb 2019.

[21] M. R. Clarkson and F. B. Schneider, "Hyperproperties," *Journal of Computer Security*, vol. 18, no. 6, pp. 1157–1210, 2010. [Online]. Available: http://dx.doi.org/10.3233/JCS-2009-0393

[22] C. Raiu, "Where are all the A's in APT?" in *Virus Bulletin*, 2018. [Online]. Available: https://www.virusbulletin.com/blog/2018/09/where-are-all-apt

[23] R. Joyce, "Disrupting nation state hackers," USENIX Enigma Conference, January 2016. [Online]. Available: https://www.youtube.com/watch?v=bDJb8WOJYdA

[24] M. Guri and Y. Elovici, "Bridgeware: The Air-Gap Malware," in *Comm. ACM*, vol. 61, April 2018.

[25] L. Columbus, "De-mistifying zero-trust network access 2.0," in *VentureBeat*, July 8, 2022. [Online]. Available: https://venturebeat.com/2022/07/08/demystifying-zero-trust-network-access-2-0/

[26] B. Lampson, "Usable Security: How to Get It," in *Comm. ACM*, Nov 2009.

[27] F. B. Schneider, "Beyond hacking: An SOS!" in *2010 ACM/IEEE 32nd International Conference on Software Engineering*, vol. 1, 2010, pp. 2–2.

[28] I. Ivan Damgård, "A "proof-reading" of Some Issues in Cryptography," in *Proceedings of International Coll. on Automata, Languages and Programming (ICALP)*, ser. LNCS, vol. 4596. Springer, 2007. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-12400-1_12

[29] D. Gollmann, "Commentary in Transcript of Discussion: Dancing with the Adversary - A Tale of Wimps and Giants," in *Proceedings of the 2014 International Workshop on Security Protocols, Cambridge, UK*, ser. LNCS, vol. 8809. Springer, 2014.

[30] V. D. Gligor, "On the security limitations of virtualization and how to overcome them (also see transcript of discussion)," in *Proc. of the 2010 Security Protocols Workshop, Cambridge, UK*, no. LNCS 7061. Springer, 2014.

[31] Butler W. Lampson, "Computer security in the real world," in *Proc. of 16th Annual Computer Security Applications Conference*, Dec 2000. [Online]. Available: https://www.acsac.org/2000/papers/lampson.pdf

[32] D. Burton, "What are the most common causes of firewall misconfigurations?" in *Akamai Blog*, November 16, 2020. [Online]. Available: https://www.akamai.com/blog/security/the-dangers-of-firewall-misconfigurations-and-how-to-avoid-them

[33] L. Columbus, "How zero trust can help battle identities under siege," in *VentureBeat*, September 14, 2022. [Online]. Available: https://venturebeat.com/security/how-zero-trust-can-help-battle-identities-under-siege/

[34] J. Saltzer and M. Schroeder, "The protection of information in computer systems," in *Proceedings of IEEE*, Sept 1975.

[35] L. Columbus, "Struggling with endpoint security? How to get it right," in *VentureBeat*, July 13, 2022. [Online]. Available: https://venturebeat.com/2022/07/13/struggling-with-endpoint-security-how-to-get-it-right/

[36] ——, "Forrester's best practices for micro-segmentation," in *VentureBeat*, July 20, 2022. [Online]. Available: https://venturebeat.com/2022/07/20/forresters-best-practices-for-zero-trust-microsegmentation/

[37] T. Keary, "94% of survey respondents experienced API security incidents in 2021," in *VentureBeat*, August 22, 2022. [Online]. Available: https://venturebeat.com/security/api-security-incidents/

[38] M. Isbitski, in *MythBusters API Edition: Zero Trust and Its Limitations for API Security*. Salt Security, April 20, 2021. [Online]. Available: https://salt.security/blog/mythbusters-api-edition-zero-trust-and-its-limitations-for-api-security?

[39] Z. Zhou, V. Gligor, J. Newsome, and J. McCune, "Building verifiable trusted path on commodity x86 computers," in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*. IEEE, 2012.

[40] M. Yu, V. Gligor, and L. Jia, "An I/O separation model for formal verification of kernel implementations," in *Proceedings of the 2021 IEEE Symposium on Security and Privacy*. IEEE, 2021.

[41] D. Parnas, "Some hypotheses about the "uses" hierarchy for operating systems," March 1976, research Report BS I 76/1.

[42] Trusted Computing Group, "Hardware Requirements for a Device Identifier Composition Engine," 2018. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-Device-Identifier-Composition-Engine-r78_For-Publication.pdf

[43] ——, "TCG Guidance for Secure Update of Software and Firmware on Embedded Systems," 2020. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TCG-Secure-Update-of-SW-and-FW-on-Devices-v1r72_pub.pdf

[44] ——, "Implicit Identity Based Device Attestation," 2018. [Online]. Available: https://trustedcomputinggroup.org/resource/implicit-identity-based-device-attestation/

[45] ——, "Symmetric Identity Based Device Attestation," 2020. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TCG_DICE_SymIDAttest_v1_r0p95_pub-1.pdf

[46] M. Xu, M. Huber, Z. Sun, P. England, M. Peinado, S. Lee, A. Marochko, D. Mattoon, R. Spiger, and S. Thom, "Dominance as a new trusted computing primitive for the internet of things," in *Proc. of IEEE Symposium on Security and Privacy*, 2019.

[47] P. England, R. Aigner, K. Kane, A. Marochko, D. Mattoon, R. Spiger, S. Thom, and G. Zaverucha, "Device identity with DICE and RIOT: Keys and Certificates," Microsoft, Tech. Rep. MSR-TR-2017-41, September 2017.

[48] B. Parno, "Bootstrapping trust in a trusted platform," in *Proceedings of the third conference on Hot topics in security*. USENIX Association, 2008, pp. 1–6.

[49] E. Palmer, T. Visegrady, and M. Osborne, "Ownership and control of firmware in open compute project devices," IBM 2018. [Online]. Available: https://www.opencompute.org/documents/ibm-white-paper-ownership-and-control-of-firmware-in-open-compute-project-devices

[50] A. Greenberg, "Why the security of usb is fundamentally broken," in *WIRED*, July 2014. [Online]. Available: https://www.wired.com/2014/07/usb-security/

[51] Kaspersky Lab, "CosmicStrand: the discovery of a sophisticated UEFI firmware rootkit," in *Kaspersky Lab*, July 2022. [Online]. Available: https://securelist.com/cosmicstrand-uefi-firmware-rootkit/106973/

[52] L. Mearian, "There's no way of knowing if the NSA's spyware is on your hard drive," *Computerworld*, vol. 2, 2015.

[53] Kaspersky Lab, "Equation: The Death Star of the Malware Galaxy," in *Kaspersky Lab*, Feb 2015. [Online]. Available: https://securelist.com/equation-the-death-star-of-malware-galaxy/68750/

[54] F. Stajano, "Security for Ubiquitous Computing," in *John Willey and Sons*, no. ISBN 0-470-84493-0, Feb 2002.

[55] V. D. Gligor, S. Luan, and J. Pato, "On inter-realm authentication in large distributed systems," in *Proc. of the IEEE Symposium on Security and Privacy, and J. of Computer Security, Vol. 2, no. 2-2,1993*, 1992.

[56] *et al.*. Hecht, Matthew, "Unix without the superuser," in *Proceedings of 1987 Summer USENIX Technical Conference and Exhibition, Phoenix, AZ*. Usenix, June 1987.

[57] NSA, National Computer Security Center, "A guideline for understanding trusted facility management," NCSC-TG-015, Library No. S-231,439.

[58] V. Gligor, S. Gavrila, and D. Ferraiolo, "On the formal definition of separation of duty policies and their composition," in *Proc. of the IEEE Symp. on Security and Privacy*, 1998, pp. 172–183.

[59] Wheeler, David, "Fully Countering Trusting Trust through Diverse Double-Compiling," in *PhD Thesis, George Mason University, Fairfax, VA*, 2009. [Online]. Available: https://hdl.handle.net/1920/5667

[60] K. Thompson, "Reflections on trusting trust," in *Comm. ACM*, vol. 27, no. 8, Aug 1984.

[61] M. Schroeder, D. D. D. Clark, and J. Saltzer, "The Multics kernel design project," *ACM Operating Systems Review*, vol. 11, Nov 1977.

[62] D. Clark and M. Blumenthal, "The end-to-end argument and application design: the role of trust," in *Federal Communications Law Journal*, vol. 63, no. 2, Feb 2011.

[63] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proceedings of the 30th Annual Cryptology Conference*, vol. LNCS 6223. Springer, 2010.

[64] M. Wallfish and A. Blumberg, "Verifying computations without re-executing them," in *Comm. ACM*, vol. 58, no. 2, Feb 2015.

[65] P. Dasgupta, "A Matter of Trust: Social Capital and Economic Development," in *Annual Bank Conference on Development Economics (ABCDE), Seoul*, June 2009. [Online]. Available: https://www.econ.cam.ac.uk/people-files/emeritus/pd10000/publications/09/abcde09.pdf

[66] G. Akerlof and R. Schiller, in *Phishing for Phools – The Economics of Manipulation and Deception*. Princeton University Press, 2015.

[67] D. Parnas, "On a "Buzzword:" Hierarchical Structure," in *Proceedings of Information Processing 74, IFIP Congress 74*. North Holland, 1974, pp. 336–339.

[68] G. Heiser, in *The seL4 Microkernel – An Introduction*. The seL4 Foundation, June 2020. [Online]. Available: https://sel4.systems/About/seL4-whitepaper.pdf

[69] M. Bishop and M. Dilger, "Checking for Race Conditions in File Accesses," *Computer Systems, vol. 9, no. 2*, 1996.

IX. **Appendix A: Zero Trust Outside of Network Security**

There are at least two ways to prove that zero trust is impossible in security. The first is to find a security property that cannot be verified or monitored unconditionally and with certainty by a provably correct verifier or external monitor. This was illustrated in Section III-A. Specifically, absence of malware that does not have known externally visible behavior pattern (i.e., no property or hyper-property) cannot be verified or monitored unconditionally and with certainty in a "back box" device. The second way is to find a security property that can be verified or monitored unconditionally and with certainty, but the correctness of the verifier or monitor is uncertain. This is illustrated in **Example 1** below.

Example 1: *Almost Zero Trust*

Suppose that a user has the unenviable task of solving a linear Diophantine equation (https://en.wikipedia.org/wiki/Diophantine_equation) in two variables which nevertheless warrants the use of computer-based tools to find *all* roots in a large integer interval. These tools have the ultimate trust liability for their integrity property: they are always believed to produce correct (sets of) roots of the equation despite using a possibly incorrect algorithm, a malware-compromised program, or a corrupt computer hardware. Fortunately, reliance on tool integrity is unnecessary, since

- the user can verify tool integrity with 100% certainty efficiently; i.e., by substituting each set of roots in the equation and checking equality to zero, which requires only two multiplications and one addition; and
- the verification is unconditional: it does not depend on (i) secrets and unproven assumptions or outside trusted help (e.g., a math enthusiast, a calculator); (ii) correct and uncompromised algorithms, software, or hardware, and (iii) limited adversary power in attacking the computer-based tools.

Furthermore, the user's verification trivially satisfies all operational security principles shown on the vertical axis of Figure 1. However, to achieve *zero trust*, the verification algorithm itself, no matter how simple (i.e., two multiplications followed by an addition), must be demonstrably correct using some measure of belief justification. Luckily, the multiplication and addition operations follow the axioms of arithmetic, and hence the only remaining belief that must be measured is that of the user's own ability to perform simple arithmetic operations on *any* integers in *any* large interval. This ability may be perfect for relatively small integers when grade-school algorithms suffice. However, in case of large integers, when recursive divide-and-conquer methods may be needed, a user's arithmetic skills are often prone to error. Hence, a verifier's failure to perform a correct check for large integers is lower bounded by *some* probability $p > 0$. This means zero trust is *almost perfect* when p is small. (Zero trust becomes *conditional* whenever calculating devices or external help are used for verification). This example suggests that the impossibility of unconditional zero trust in enterprise networks is not a mere theoretical fact. As early warnings imply, it is hard to find unconditional zero trust anywhere in security or cryptography. ■

Example 2: *Conjectured Zero Trust*

Early work that introduced the notion of *verifiable computation* [63] showed that – efficiency notwithstanding – a client can always verify the results of computation outsourced to a wholly untrusted server (i.e., not even the hardware was trusted) conditionally. That is, the result verification by clients is predicated on the inability of a polynomially bounded adversary to break the then-required fully homomorphic encryption (FHE) primitive. Unavailability of practical FHE primitives over the following decade motivated intensive research in efficiently verifiable computations in real applications [64], covering integrity of blockchain, non-fungible token, and contract-signing computations. All these applications illustrate useful forms of reduced but definitely *non-zero* trust in practice. Zero trust is conjectured because the cryptographic primitives used assume polynomially bounded adversaries and demonstrably correct trust zones of verifier devices. ■

Example 3: *Impractical Zero Trust*

Removing trust liabilities is not always possible nor practical in computing. For example, verifying the result of a *co-NP complete* problem produced by a computer can be highly inefficient – more so than solving the problem in the first place – and hence often impractical and left undone. There exist numerous examples of patterns produced by an untrusted source whose recognition by a trained machine-learning (ML) algorithm can be successfully attacked; e.g., the ML algorithm recognizes an unintended pattern. An adversary can capture the source of the patterns and produce forgeries. In these cases, verification can be incomplete or incorrect, and hence zero trust cannot be achieved.

Another example of zero-trust impracticality appears in network-mediated asymmetric protocols among human subjects; i.e., protocols in which an initial sender of a message benefits from the execution of the protocol but the recipient of that message has no guarantees of sender's honesty [15]. Can a recipient verify the content of the message received from a sender? Most recipients cannot perform the content verification and must rely on external providers, or external trust zones, that use ML classifiers for rating protocol input message as safe/dangerous. This solution is safe but non-zero trust. When ground truth is not discernible by deep ML classifiers, external implicit trust zone must expand as competition among a marketplace of classifiers becomes necessary to establish which ML classifiers are the *de facto* holders of ground truth in input-message verification. ■

Example 4: *Zero Trust can be Undesirable*

Historically, humans have had implicit trust zones defined by common socio-cultural, political, and business beliefs and preferences. In some communities, individuals trust each other without continuously verifying each other's every decision in matters of common welfare; i.e., trust is definitely non-zero, by any definition. Often community trust is implicitly established and not repeatedly verified. Behavioral economics found an early *correlation* between non-zero trust and wealth, showing that when humans trust each other their communities become wealthy faster than those in which humans do not [14]. In 2009, Dasgupta [65] showed that communities where humans trust each other increase their total factor productivity faster than those in which humans do not, thereby establishing a *causality* between non-zero trust and wealth. Of course, in wealthier communities, trust is established based on beliefs in the trustworthiness of others, decreased risk aversion, and decreased betrayal aversion. Trust establishment occurs at a "phishing equilibrium;" i.e., in the presence of fraudsters who are a staple of economic life [66], just as adversaries are in security.

X. **Appendix B: Zero trust is a "buzzword"**

Nearly five decades ago, David Parnas examined the use of the hierarchical structure "buzzword" in operating systems [67]. He has pointed out that buzzwords often lack clear definition, as their proponents:

A. assign different meanings to them in different systems;

B. do not explain them to others;

C. are unable to rule out inadequate alternatives; and

D. adopt imprecise terminology.

We use Parnas' intuitive yet precise characterization of "buzzwords" to argue that "zero trust" has earned the status of a buzzword in network security.

The *zero-trust* buzzword can have egregious implications in security. For example, when used beyond its original intent of removing exclusive reliance on location-based perimeter protection, it can lead to demonstrably inadequate countermeasures to common security attacks, let alone advanced and yet-unknown exploits. The buzzword label itself implies the impossible in network security and excludes *trust establishment* as a sound alternative.

A. Different meanings

Zero-trust architectures can have different meanings in different network architectures.

1. In the simplest case, an enterprise network has a fixed set of known endpoint devices and servers, and a single administrative domain; i.e., a potentially large implicit trust zone. As shown in *Example 4* of Section V-B above, this architecture (e.g., illustrated by NIST [1]) fails to minimize this large trust zone as it lacks basic minimization criteria.

2. Enterprise-network architectures can be extended to allow users to "bring your own devices" (BYODs) rather than limit configurations to fixed-endpoints networks. *Example 2* of Section V-B illustrates the expected insecurity consequences.

3. Multi-cloud services and patch servers under the control of externally administered suppliers are also encouraged, and so are multi-enterprise (e.g., *peer*) networks. This introduces a new form of trust – inter-enterprise authentication trust [55]. *Example 3* of Section V-B illustrates how this makes enterprise networks dependent on externally administered trust zones without meaningful recourse.

4. While not recommended by NIST but encouraged by industry [6], enterprise-network architectures allow IoT and edge devices that can operate in an *unattended* manner; e.g., devices that could be captured and manipulated by an adversary. Even if such devices are assigned immutable unique identities [42], [47], an enterprise network cannot often detect an adversary modification of the firmware in a physically compromised device; see *Example 3* of Section V-B.

B. Unexplained and unanticipated consequences

The key characteristics of zero trust have four unexplained and unanticipated consequences. The examples below illustrate four areas. Others exist.

1. Section III shows that *zero trust* is unachievable in any enterprise network. This means that it is both theoretically impossible as well as fundamentally impractical for network application. Hence, it is senseless to ask [2], [3], [4], [5] for any enterprise network to achieve it.

2. *Trust-zone* minimization is often conflated with *trust* minimization. For example, NIST's zero-trust architecture [1] incorrectly suggests that the so called "zero-trust principles" can provide *trust minimization*. Even if minimization shrinks an implicit trust zone to a single device, it *cannot* minimize trust because it admittedly does *not* provide any correctness assurance for services within the trust zone. No matter how useful, continuous verification, monitoring, and "zero-trust principles" can never overcome the lack of security-property assurance for services and devices inside a trust zone; see Figure 1.

3. Implicit trust zone minimization can never be achieved by an enterprise network for all its *externally administered* trust zones, as illustrated in Figure 7 and explained in *Example 3* of Section V-B. Reliance on external servers expands an enterprise's implicit trust zones and prevents trust minimization since external-server secure operation may be assured, if at all, outside the jurisdiction of the enterprise. Unaccountable external patch servers must often have unobstructed access to enterprise device software and firmware, and lowering privileges of these devices cannot prevent cross-zone attacks; see *Example 3* in Section V-B. An adversary's locus of attack is thus expanded without recourse.

4. *Trust establishment*, which fundamentally excludes *zero trust*, is necessary in practice. Since *zero trust* is unachievable, trust establishment becomes necessary; for example, in enterprises engaging in electronic commerce or in supporting social interactions [15]. When assurance of secure operation in an implicit trust zone is low, few beliefs of trustworthiness can be justified. Then both attack-risk mitigation and attack

deterrence must be employed to compensate for the extra security liability. (Addressing any subset of them is insufficient, as already noted in Section II-B and illustrated in Figure 1). Consequently, analysis of trade-offs among justified beliefs of trustworthiness, risk mitigation, and attack deterrence is also missing from zero-trust architectures.

C. Demonstrated inability to rule out inadequate alternatives

Zero-trust architectures admittedly avoid correctness assurance and fail to employ some key operational assurances. While this lowers development cost significantly, it also implies that these architectures will be forever be unable to rule out insecure network and applications designs. This means that all "zero-trust principles," verification checks, and monitoring employed by these architectures are insufficient to protect critical infrastructures. Here we illustrate two classes of inadequacies that are not ruled out. Others exist.

1. *Zero-trust architectures* are vulnerable to incorrect enforcement of verification checks. Examples include:

- inability to guarantee that verification-check isolation cannot be subverted; e.g., the signature verification check guaranteeing that a firmware upgrade[22] by an external supplier is authentic must be isolated inside the device's micro-controller by a verified firmware operating system (a la *seL4* [68]);

- inability to eliminate time-of-check-to-time-of-use (TOCTTOU) gaps in applications, hosts, and network protocols, and rule out well-known vulnerabilities [69];

- inability to eliminate inadequate composition of verification checks that violate a desired policy; e.g., the order of mandatory (MAC) and discretionary access-control (DAC) checks matters, since an incorrect order (i.e., DAC before MAC) violates information flow. Otherwise, new covert channels are introduced in violation of an information-flow policy.

A more general problem is that *zero-trust architectures* fail to distinguish security properties that are verifiable/testable from those which must be continuously monitored externally. For example, they fail to separate which security properties can be black-box tested and which require external monitoring [17]. This is particularly hard to reconcile since one of the key requirements of *zero-trust architecture* is to use black-box methods and tools (e.g., remote cloud-based tools) for verification/testing and monitoring of network devices and services. Inexpensive, scalable use of cloud services to monitor remote endpoint systems and privileged system components, such as OS kernels, is assumed by these architectures.

2. Cloud-based implementations of *zero-trust architectures* are vulnerable to endpoint device attacks.

Exclusive reliance on cloud-based, client-less monitoring of endpoint devices can be inexpensive but is inadequate since end-to-end security can *never* be established by a single (cloud) end. Deep endpoint-device scanning both by remote cloud tools (e.g., intended to detect misconfiguration) and anti-malware tools cannot possibly work with significant assurance because their remote device-firmware scanning requires *direct access* to firmware content, which is impossible for black-box devices; see Section III-A.

D. Inability to adopt precise terminology

The "zero trust" label itself fails to inspire confidence, as it assigns a sharp all-or-nothing attribute (zero/non-zero) to a noun (trust) that has always signified a range of beliefs and preferences. For example, less verification leaves a higher trust liability and more verification leaves less.

More precise terminology would distinguish the weaker notion of implicit-trust-zone minimization from the strictly stronger notion of trust minimization (which requires correctness assurance), and both would be distinguished from trust establishment – a stronger notion than both. These distinctions are essential for understanding the virtues and limitation of the zero-trust architectures.

[22]"By 2022, 70% of organizations that do not have a firmware upgrade plan in place will be breached due to a firmware vulnerability." Quoted by *Enterprise Best Practices for Firmware Updates*, *Eclypsium*, April 2, 2020. https://eclypsium.com/2020/04/02/enterprise-best-practices-for-firmware-updates.