BYOZ: Protecting BYOD Through Zero Trust Network Security

John Anderson
Clemson University
jda3@clemson.edu

Qiqing Huang
University at Buffalo, SUNY
qiqinghu@buffalo.edu

Long Cheng
Clemson University
lcheng2@clemson.edu

Hongxin Hu
University at Buffalo, SUNY
hongxinh@buffalo.edu

***Abstract*—As the COVID-19 pandemic scattered businesses and their workforces into new scales of remote work, vital security concerns arose surrounding remote access. Bring Your Own Device (BYOD) also plays a growing role in the ability of companies to support remote workforces. As more enterprises embrace concepts of zero trust in their network security posture, access control policy management problems become a more significant concern as it relates to BYOD security enforcement. This BYOD security policy must enable work from home, but enterprises have a vested interest in maintaining the security of their assets. Therefore, the BYOD security policy must strike a balance between access, security, and privacy, given the personal device use. This paper explores the challenges and opportunities of enabling zero trust in BYOD use cases. We present a BYOD policy specification to enable the zero trust access control known as BYOZ. Accompanying this policy specification, we have designed a network architecture to support enterprise zero trust BYOD use cases through the novel incorporation of continuous authentication & authorization enforcement. We evaluate our architecture through a demo implementation of BYOZ and demonstrate how it can meet the needs of existing enterprise networks using BYOD.**

***Index Terms*—zero trust, network security policy, BYOD, continuous authentication**

I. Introduction

With the onset of the COVID-19 pandemic, the world has had to quickly adapt to changing working environments, including working full time and remotely from home. As companies begin to bring staff back to the office, Bring Your Own Device (BYOD) programs have seen an uptick in adoption and expansion as a means to cope with the influx of both on and off-premise compute needs [1]. BYOD is the use of one's personal electronic device for business or other non-personal use cases. This sort of program can be meaningful to a business because of the cost reduction in device management and the flexibility it grants the employees. However, BYOD comes with a myriad of security and technical hurdles that must be carefully managed to maintain the same or better standard of protection for that of company-owned devices [2].

As networks themselves and the applications they support have grown in complexity over the years with the advent of cloud computing, the Internet of Things (IoT), and 5G networking use cases, the need to secure users, companies, and their data is at the forefront of operator's minds [4]. Meanwhile, Zero Trust Architectures (ZTA) for networks have emerged as a fundamentally new way of approaching network security [3]. ZTA offers new paradigms for defining and enforcing policy through various means rooted in modeling trust relationships.

ZTA is based on the premise of explicit trust relationships expressed through administrative policy. This policy is the core of ZTA but is lacking in understanding today. There needs to be the ability to express high-level business needs and derive lower-level policy intent, enforced through various access control mechanisms. In today's enterprise environments, implementing this intent is tedious and error-prone because of the vast contexts a given policy intent may span and the subsequent tools and technologies that must act homogeneously to enable an administrator's policy [20]. This new policy ecosystem needs to understand and adapt to BYOD use cases. Thus, naturally, we feel the marriage of ZTA and BYOD will ultimately offer the right approach to access control policy and management to build a successful enterprise security posture. BYOD administrators need effective ways to establish security policy, and ZTA builds these constructs into its foundation. The COVID-19 pandemic has shown that companies are not willing to bring their business operations to a halt because of employee system access pattern changes. Instead, they have adapted to changing circumstances. They have not, however, always done so in a secure way, especially when enabling employees through the use of BYOD programs [36]. Embracing BYOD as a core component of one's ZTA-based network instead of trying to bolt security onto existing BYOD solutions will ultimately be the path forward. As we begin to see the end of the COVID-19 pandemic, it is clear the catalyst for the remote work transformation will not be slowing down, only strengthening the argument for the need to bring ZTA to BYOD [6].

Policy, of course, is only a part of the solution, as one needs a system to enforce the policy. ZTA, along with the NIST standard, outlines several tactics and methodologies, from which systems can be architectures to meet the new needs of networks [5]. This paper describes the need for ZTA in a BYOD-focused access model and proposes a new ZTA policy language and enforcement architecture. Two central features of our solution revolve around continuous authentication & authorization and dynamic context management, which we see as key to meeting the fundamental needs of Zero Trust. We will

978-1-6654-5408-7/22/$31.00

show what our system architecture can be used to implement ZTA enforcement constructs in an enterprise-friendly manner, not requiring the deployment of entirely new technologies. We offer an instantiation and analysis of our architecture and policy language built on top of iptables [42] through the use of specific attack scenarios.

In summary, our work makes the following novel contributions through the implementation and analysis of our policy language and enforcement architecture known as *Bring Your Own Zero Trust* or *BYOZ*.

1) We develop a high-level policy language that addresses enterprise BYOD's needs through adherence to zero trust network architectures.
2) We design and evaluate a network architecture that enforces continuous authentication & authorization to enable zero trust network security for BYOD use cases.

We prove through our results that BYOZ can successfully address these issues and meet the needs of securing enterprise BYOD use cases through the adoption of Zero Trust network security.

II. Motivation

Our work in this space is motivated by a set of core ideas and gaps in the literature. First, ZTA, on its merit, is not a tool, solution, or standard that can be implemented and analyzed. Instead, it can be thought of as a guiding framework with a set of foundational concepts that can be applied across a variety of security domains that, when reasoned in aggregate, form the principles of ZTA. This framework primarily seeks to flip the traditional enterprise access paradigm of security posture such that all authorization is explicitly accounted for and expressed. This is achieved through various means such as isolation and segmentation, mutual and continuous authentication, holistic policy expression and administration, and state feedback loops for events and changes in the environment.

Second, at their root, Zero Trust Architectures operate on the premise of a fully explicit policy. That is, there can be no implied trust relationships in any network, application, access control, or other policies an organization implements. A simple analogy is to never trust and always verify. In principle, this idea is sound and ensures that if some flow is to be permitted, there is guaranteed to be a policy that expresses this administrative desire [18]. If enacted within existing enterprise ecosystems, a simple example of a user accessing an internal website would require the policy orchestration of host-level firewall rules, network Access Control Lists (ACL), network firewall policies, and application-level ACLs [20]. In effect, each of the various policy sets is siloed from one another [19] [21]. Different policy languages must be understood, and the lack of feature parity makes simple translation and interoperability difficult.

These challenges motivated us to build a solution that meets three main goals; first, build a system that allows for simplified policy expression by an administrator, regardless of the underlying enforcement technology. Second, provide such an enforcement architecture that fits within existing enterprise network deployment models, which uses concepts of continuous authentication to implement zero trust. Finally, in both cases, focus on the use of BYOD since it is clear this will continue to be a significant access means in the enterprise.

III. Threat Model

It is vital that we clearly define the threat model and environmental assumptions we are targeting in our design and implementation. Since a motivating driver for our work is the usage of BYOD, many of our attack vectors focus on a compromised BYOD device joining the network or the act of compromising a BYOD device while connected to the network. We consider a compromised BYOD device to be under the control of a malicious subject, which could either be the legitimate device owner or a 3rd party acting without the consent of the device owner.

While our system design strives to adhere to zero trust primitives set forth by NIST, we make several assumptions about the administrative and infrastructure components of the environment. For example, we talk about profile extraction and the actions between BYOD devices and the controller in our architecture in Section V. However, we assume the controller and this communication channel to be secure, using existing industry protocols and standards like TLS [44]. We acknowledge, however this is realistically an open area for further research, especially as it relates to zero trust compute resources. There are obviously ways an attacker may attempt to exploit the controller or other components in our design, but this too, we leave to future work and focus in this paper on attack scenarios directly relating to the use of BYOD devices and subjects. These specific scenarios will be discussed in Section VI

Section V discusses the BYOZ architecture in detail, but it should be noted that our design itself is meant to be flexible and adaptable as a blueprint for implementation in several different ways. For this reason, our experimentation utilizes an implementation of our design comprised of certain standardized components rather than reinventing portions of the system. For instance, we use MQTT as the message bus for communication between components and iptables as the low-level policy enforcement mechanism [45] [42].

IV. Related Work

A. Zero Trust

Perhaps the most well-known and directly relevant work related to zero trust is the architectural proposal published by NIST [5]. Here NIST describes the principles of zero trust and provides an architectural framework for implementing zero-trust network security in various scenarios. These deployment scenarios include campus branch, cloud, third-part access, federation, and public-facing services. ZTA is not a new concept. However, as pointed out by Kindervag et al., ZTA comes about from realizing that traditional network security is implemented as an overlay to fundamental network transport architectures [29]. A significant aspect of the proposed zero

trust architecture proposed by Rose et al. is the concept of closed-loop feedback systems [5] [27].

B. BYOD

Enterprises have come to realize that the personal devices that their employees already own are generally capable of the same sort of operations technically necessary to access company assets, opposed to expensive devices owned and issued by the organization. This makes BYOD a financially lucrative proposition for many enterprises, forming the basis of the trade-offs between security and operability that we researchers are drawn towards with BYOD [33]. On the other hand, before the COVID-19 pandemic, in the earlier days of 2020, BYOD was still seen by some organizations as a security conundrum that they may never really have to tackle. However, with the massive shift in workforce needs, BYOD became a saving grace to many organizations [32].

C. Policy Languages

In terms of zero trust and BYOD, there is a need for a comprehensive policy language that can address the specific requirements of the solution but also be able to bridge the gaps in enforcement technologies. Hong, et. al., contribute their PBS language, which focuses on BYOD use cases, but not through the lens of zero trust. The PBS policy specification dawned several attributes for our languages, such as the physical location and the ability to re-check a flow periodically [7]. We also drew ideas from the Poise project by Kang, et al [8]. Specifically, Poise led us down the path of deriving low-level intent from a higher-level language (discussed in Section IV) through the building of a P4 transpiler [8]. It is also important to consider existing industry solutions for security policy, like network ACLs from the Cisco IOS operating system [9], policy statements from the Juniper Junos operating system [10], and firewall policy from the Palo Alto Networks operating system [11]. While each of these languages makes it's own unique contributions, we found none to solve the needs of expressing high-level, dynamic BYOD context for zero trust enforcement and so we created BYOZ.

V. BYOZ Design

We explain the concept of our ideas for BYOZ through a description of the high-level zero trust policy language definition and a system architecture that is capable of accepting such policy input from an administrator and enforcing its intent. We further present a concrete system that acts as a proof of concept to illustrate the effectiveness of our architecture in serving BYOD zero trust use cases. We point out that our architecture may be implemented with existing enterprise technologies via Software Defined Networking (SDN) [41], or as a part of an entirely new system design.

A. Policy Language

Zero trust implies, in many contexts, the ability for higher-level intent to be used to derive lower-level policy statements, which calls for a language capable of this abstract definition.

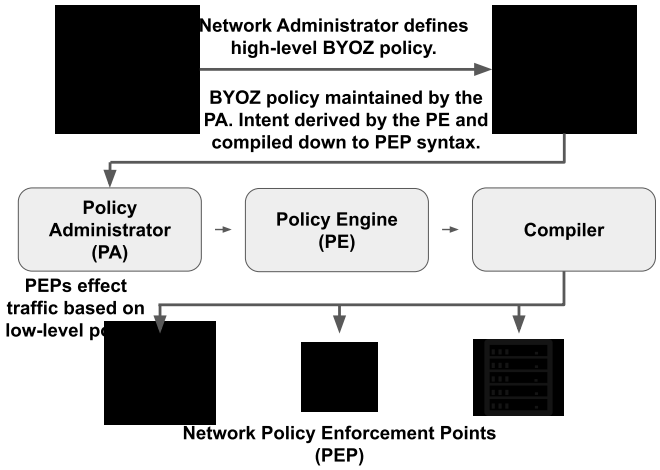

Fig. 1. ZTA infrastructure showing the ability to compile and derive lower-level access control policy intent from higher-level policy definition.

This comes from the NIST standard surrounding zero trust and emphasizes central administration of the policy [5]. While such languages exist today, we argue that none specifically address the unique concerns related to enabling Zero Trust Architectures in BYOD environments. The Zero Trust Architectures themselves propose the infrastructure necessary to support a system of high-level intent without proposing a concrete language. It is our position that such a language needs to be introduced, noting we limit ourselves to the use case of supporting Zero Trust Architecture for BYOD security. Figure 1 shows an overview of derived high-level intent, in which the zero trust business policy is defined using a high-level, user-friendly language and is translated into lower-level intent, which can be acted upon and enforced by various security mechanisms. ZTA also requires a lot of context about the running environment like a central Identity Provider (IDP) [25] or an application registry. We integrate these sources to understand subjects and resources in BYOZ.

Beyond these inventory and state integration, the BYOD use cases call for specific unique characteristics important to our policy definitions. One of these is the concept of logical location. We want to understand and act on metadata about the heuristics of where on the network a device is connecting from. This is about more than simply knowing the subnet a device is attached to and is akin to understanding that the connection is occurring over WiFi, what the SSID is, and whether or not the WiFi connection is encrypted. From a BYOD standpoint, we feel this is an essential attribute because it gives our policy context of what type of environment the source device and user are operating within. By elevating this context to the policy level via our architectural implementation, we allow that context to be exposed throughout the entirety of the end-to-end flow. Ultimately, this logical location context is conjoined with the physical location of the source device obtained via GPS or other means when available.

Figure 2 shows our proposed language specification in terms of all defined attributes and their values. We make use of other

attributes which are more or less standardized for general use cases in terms of network security policy. These include items like host state, defined from Host Information Protocol (HIP) checks [24], a policy action, such as allow and deny, and an event type, which specifies what portion of the flow upon which the rule should act.

Again, our goal is to enable BYOD use cases within zero trust security architectures; thus, each of these attributes plays a role in one or both of those aspects. We attribute these policy attributes to BYOD functionality and use cases: source type, host, host state, location physical, and location logical. As shown in Figure 2, these attributes contribute to zero trust: Source Type, Host State, Location Physical, Location Logical, Destination Type, Event, and Apply Action. The location attributes are of specific interest to our BYOD use cases because of the highly mobile nature of these users. Combined with the logical location, we can tackle remote and on-premises use cases with more than simple network segmentation controls. The destination attributes in Figure!2 show an understanding and reliance on external context information in our system, such as an inventory of potential resources and metadata about those resources. This can include where they reside, potentially what kind of data they hold, or their risk profile to the organization. This metadata can be used to make more informed policy decisions when a subject wishes to access a resource.

To refine the ability of the ZTA policy to be rendered to enforcement languages, we have built a compiler for our language, which is a part of the enforcement architecture. Figure 3 shows an example policy expressed in our language to service the use case of a managed BYOD device connecting to an internal application from a public WiFi network. This highlights the need for integration with inventory systems, for example, application repositories, to specify the infrastructure on which those applications run. From the policy expression, our compiler can render the policy output for iptables, and the Palo Alto Networks firewall policy syntax [11] as shown in Figure 4. In doing so, we show that our enforcement architecture can support any number of security appliances with their own policy language implementation since we can compile down from our high-level BYOZ definition. Imagine that an administrator uses the policy specification in Figure 2 to define a high-level policy such as that in Figure 3 and the compiler in our system renders that policy down to the enforcement languages in Figure 4.

```
Source Type       := (PRIVATE | PUBLIC
                     | HYBRID)
Source            := {Source Type
                     + Value}
Host              := (MANAGED | UNMANAGED)
Host State        := {HIP Metadata}
Location Physical := {GPS}
Location Logical  := {Source Type
                     Metadata}
Destination Type  := (NETWORK | APP
                     | CLOUD)
Destination       := {Destination Type
                     + Value}
Time              := {Date time range}
Event             := (Session Init
                     | Quarantine Check
                     | Interval)
Action            := ALLOW | DENY
                     | {(REDIRECT
                        | QUARANTINE)
                       + ADDRESS}
Apply             := (IMMEDIATE
                     | {PERIODIC
                       + Value})
```

Fig. 2. BYOZ language specification showing all attributes and possible values.

```
Source Type       := PUBLIC
Source            := 5.5.5.5
Host              := MANAGED
Host State        := HIP SAFE
Location Physical := 41.40338, 2.17403
Location Logical  := WIFI::Starbucks::OPEN
Destination Type  := APP
Destination       := ERP
Event             := SESSION_INIT
Action            := ALLOW
Apply             := IMMEDIATE
```

Fig. 3. Example policy showing a situation of a BYOD managed device connecting to an enterprise app from a public WiFi network.

```
iptables -A INPUT \
  -s 5.5.5.5 \
  -p tcp \
  --dport 443 \
  -d 10.10.10.10,11.11.11.11 \
  -j ALLOW

set security policy
  "5.5.5.5 to 10.10.10.10,11.11.11.11"
  from untrust to trust
  source 5.5.5.5
  destination 10.10.10.10,11.11.11.11
  applciation iROAR
  protocol tcp port 443
  action allow;
```

Fig. 4. Rendered iptables followed by Palo Alto Network firewall rules, compiled from the example policy in Figure 3.

B. Enforcement Architecture

The BYOZ architecture builds upon NIST ZTA concepts and serves to clarify specific roles and how they are to be implemented in enterprise environments for BYOD use cases. To this end, we make use of standard components, including Policy Enforcement Points (PEP), Policy Decision Points, Policy Administrator (PA), Policy Engine (PE), and the constructs of subject and resource. In our architecture, we implement the three central components–Policy Engine, Policy Administrator, Policy Decision Point, and Context Manager–on a single system and collectively refer to this as the Zero Trust Controller or just the Controller. Indeed, this borrows from core concepts of SDN, and while we have not implemented our evaluation system using SDN, it is entirely feasible.

The first significant and novel component of our BYOD-focused architecture is the Context Manager (CM), which performs profile extraction from the BYOD Mobile Device

Management (MDM) agents and aggregates context from external systems. The agent resides on the subject host device and collects metadata about the state of the device and provides it to the context manager. A realistic enterprise scenario would be implemented via the Host Information Protocol (HIP) through a BYOD MDM program.

The CM's role is to process the profile information to maintain a current and accurate state of each subject and their BYOD devices. We discussed previously that continuous authentication & authorization is a key aspect of our solution, and it is the Context Manager which implements this functionality. The Context Manager is responsible for poling subject devices at regular intervals to update the authentication state of open connections. This state is fed into the Policy Engine, which may act upon changes in authorization in real-time. We refer to these regular poling intervals as the continuous context check.

The network administrator uses the BYOZ language in Figure 2 to specify policy intent within the PA which is the interface between the controller and the outside world for defining policy and understanding its meaning.

The PDP will use this intent from the PA in combination with the most up-to-date information in the context manager to make policy decisions that are pushed into the PE. The PDP is responsible for inspecting all profile information included in the device context. In a given flow request, the PE then will generate the enforcement intent for the PEPs, and then these PEPs will act upon that intent. The network forwarding nodes are the Policy Enforcement Points in our architecture and receive dynamic policy updates from the Policy Decision Point through the management network. This enforcement action is for the purpose of establishing or closing a flow between a BYOD subject and the resource that this subject wants to access. Our reference implementation of the architecture uses iptables [42] running on connected nodes to act as the PEPs. Our PE orchestrates the automated deployment and retraction of iptables policy on the PEP nodes.

When a BYOD subject wants access to a resource within the network, it must request permission from the PDP. Therefore, a request will be made from the device agent and sent to the PDP through the CM, depicted by (1) in Figure 5. The PDP will attempt a BYOZ policy match with input from the most recent subject and device context from the context manager. Unlike the routine device continuous context check, this flow initiation request will include the resource the subject wants to access. This is depicted as (2) in Figure 5.

When a policy match occurs that results in an allowed flow, the PE will compile the BYOZ policy specification down to PEP-specific syntax, which is pushed to the PEP, as shown by (3) in Figure 5. Again, our implementation renders iptables policy, however, our policy compiler also supports Palo Alto Network firewall syntax [11] as an example of the extensibility of the architecture and specific plug-ability of the compiler. Figure 6 shows this overall workflow. A valid flow remains active in the network for as long as it is needed or dictated by policy. The PDP does, however retain the critical right to revoke a flow at any point in time if the dynamic context of the

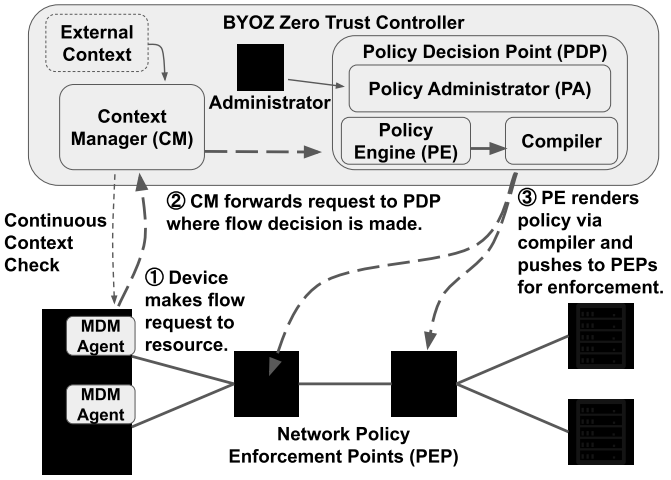

Fig. 5. High level design showing major architectural components. Here we depict the basic steps in BYOZ for a device to request a flow to a network resource.

subject or device changes in a negative way. The highlighted portion of Figure 6 shows how this action is taken by the PDP upon a continuous context check.

All the communications within the network among the BYOD devices, CM, PDP, PE, and PEP, are encrypted to ensure the integrity of the message. The encryption and decryption keys are assigned and stored before the BYOD device connects to the network. The keys are generated and configured when registering the BYOD device via MDM using standard TLS.

C. Continuous Authentication & Authorization

At the core of our design are the concepts of continuous authentication & authorization, plus context management. Traditional network access paradigms are predicated on an early and often single authentication check and this fundamental trait has been exploited numerous times through means such as session hijack attacks. Our system employs the idea of continuous authentication which fits well within the scheme of ZTA because as the name implies, there is no implicit trust of an active session simply because it exists on the network. Instead, the authentication and authorization of the session's subject credential is interrogated throughout of the lifetime of the flow by the network Policy Enforcement Points, Policy Engine, and Context Manager.

Our design conforms to the abstract architecture specified in the [5]. Each time a BYOD device wants to access a resource within the network, it must communicate with the controller that implements the Policy Decision Point to verify its identity and authorize its permission to acquire the access.

The access control workflow in our demo system is shown in Figure 6. After the BYOD device is registered and connected to the network, the MDM agent will continuously report profile metadata about the device and its subject to the CM. As stated previously, the device agent is also responsible for sending a flow request to the CM for evaluation when the device wishes to access a resource on the network. This PDP will

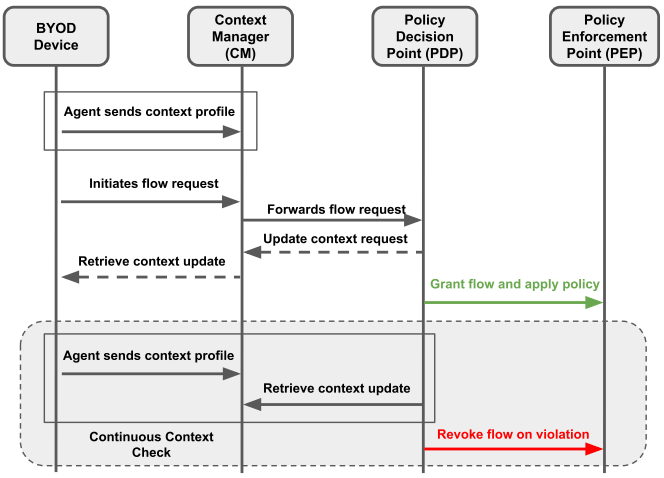

Fig. 6. Sequence diagram depicting the BYOZ flow request and context check protocols.

evaluate this request and either allow or deny the flow. In the case of an allowed flow, beyond the PEP involvement already covered, our system implements continuous authentication and authorization for the duration of the flow. Because the device agent will periodically report state, the CM and PDP have the opportunity to reevaluate the eligibility of the device and/or subject to access its requested resource. A subject change on the device (i.e., switching users on a shared device) would register in the CM as a change of authentication. Indeed, the CM may also take an external context, for example, from a Security Information Event Manager (SIEM), which may inform the CM that a device or subject is compromised in some meaningful way. The CM aggregates all this information and makes it available to the PDP for evaluation. At each interval, the PDP may decide that the BYOD device or subject now violates the previously allowed policy. In such an event, the PDP would instruct the PE to revoke the flow policy from the PEPs, effectively disconnecting the device from the resource. In another case, an administrative decision may be made elsewhere in the organization that the employee no longer has access to some resource or data. By understanding this external context, our architecture would be able to react to this dynamic authorization change and act accordingly, revoking any active flows.

VI. Evaluation

A. System & Experiment Setup

For our evaluation, we implemented a prototype of the BYOZ system described in Section V. Later parts of this section will describe the BYOD device specifications used in the various test scenarios. In each case, though, we used the same controller and network PEP environment, also with a static set of resources. The controller was implemented primarily in Python and utilized the MQTT messaging protocol for communication between the components [45]. The PDP is a single process that comprises the PA and PE. The PA function takes as input a JSON file that contains BYOZ policy objects created by the administrator. For each experiment discussed later, a specific policy set was crafted. The PE implements the BYOZ compiler and, in this case, is configured to render iptables PEP policy and apply that policy to the PEPs. The rest of the PDP implementation deals with the logic of receiving a flow request from the CM, performing a policy lookup, and acting accordingly via the PE. The CM is a separate process that deals with communication from the BYOD agents. It maintains the context database in memory and also takes as input another JSON file which contains a listing of resource objects and metadata about those resources like IP addresses and application location. This mechanism simulates an external connection between the CM and a system or set of systems, which would provide that level of dynamic resource context in a real-world situation.

Using the prototype instance of our BYOZ architecture, our goals are to show that we can enable zero trust security policy enforcement in dynamic BYOD environments. Our prototype system can realize dynamic policy enforcement and continuously authorize to meet these needs while following the NIST standards for ZTA. To demonstrate these achievements, we measure the performance overhead and impact on both the network and system.

B. Performance Analysis

1) Network Performance Overhead: We performed an analysis of the overhead incurred in the network for the operation of the BYOZ access flow. The first evaluation is measuring the delay caused by the continuous context check when the number of BYOD devices increases. As shown in Figure 6, the continuous context check persists for a BYOD device so long as it maintains an active flow to a resource. To ease this management traffic pressure caused by the continuous context check sessions, we designed a dynamic scaling function in the controller based on our reliance on the MQTT protocol and, precisely, the publish-subscribe (pub-sub) nature of communication. Each BYOD device connecting the network is issued its own MQTT topic channel for communication. This not only adds a layer of protection in the form of isolation, but we can also scale the number of PDP processes necessary to subscribe to all channels. In this way, a new PDP process will be running for the policy match on a newly generated continuous context check when a batch of new access requests arrives at the zero trust controller. Therefore, if there is an influx in connections and thus continuous context checks, this load can be dynamically scaled as needed by the controller. It is important to note, however, that our MQTT implementation was specific only to our prototype, and overall BYOZ architecture would support other means of communication.

Several reasons can cause a surge of continuous context checks. One reason can be that a malicious user is trying to generate many access requests to the controller. The PDP, in concert with the CM, should be able to discern this as abnormal behavior, given an understanding of normal conditions in the network. The zero-trust controller will block the traffic from this single device by detecting the abnormal profile in the context. Another reason can be that there is a

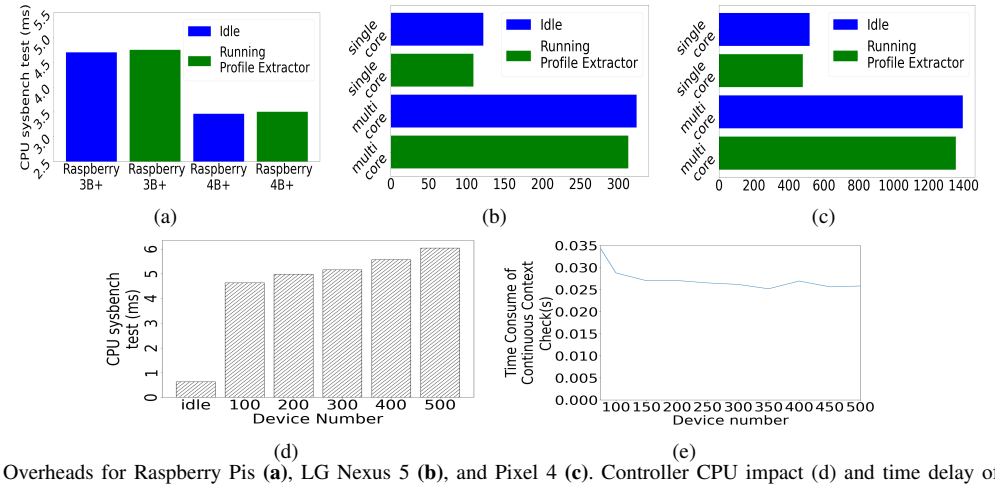

Fig. 7. Performance Overheads for Raspberry Pis **(a)**, LG Nexus 5 **(b)**, and Pixel 4 **(c)**. Controller CPU impact (d) and time delay of context check (e) as the number of BYOD devices increases.

surge devices that want to request access to a resource at the same time due to some specific event within the network. We have tested the second scenario with our demo system and measured the time taken for the continuous context check. We set up the time interval of the continuous context check at 10 seconds. After the PDP grants access from one BYOD device, the zero-trust controller will keep checking the context of this approved BYOD device every 10 seconds. We also tested our demo system with a different number of associated devices and let them run for a long period of time. As shown in Figure 7(e), when there comes up a big number of devices, the process scaling mechanism will be triggered and it will make sure the time delay is at a stable value and free of the worry about a large quantities of devices influx into the system. The continuous context check itself is a relatively lightweight operation both in terms of resources and time.

2) System Performance Overhead: To measure the system performance overhead imposed by the continuous context check, we also tested the system overhead on both the BYOD device and controller sides to measure the overhead of the MDM agent sending the profile context to the CM and the CM/PDP processing that context check.

We first performed this evaluation in the demo system using Raspberry Pis because they are readily available to us and are easy to configure for this type of testing and on-box analysis. We simulated BYOD devices using the Raspberry Pi 3 Model B+ equipped with 1.4GHz 64-bit quad-core processor, 1GB of SRAM, and the Raspberry Pi 4 Model B+ equipped with a quad-core 64-bit processor, 4GB of RAM. Both ran the Raspbian operating system. We implemented our controller on an Ubuntu Linux x64 VM with 2 processor cores and 4 GB of memory. We ran the sysbench benchmark tool [43] before and during the time when the BYOD devices send context profiles to the CM. As shown in Figure 7(a), for the Raspberry 3 Model B+, the CPU sysbench test only exhibited an increase of 0.05 ms after it runs the MDM agent and sends the context profile to the CM on a continuous basis. This delay increases 0.04 ms for the Raspberry 4 Model B+.

We also used an LG Nexus 5 and a Google Pixel 4 to measure the performance overhead imposed on the BYOD device. LG Nexus 5 ran on Android 9 OS, and the Google Pixel 4 ran on Android 12 OS. We ran the MDM agent process on these two mobile phones and let them send the context profile to the CM at an interval of 1 second. We used the same controller from the previous experiment and ran the Geekbenchmark5 [46] before and during the time when the MDM agent sent context profiles to the CM. As shown in Figures 7(b) and 7(c), the overheads for these two devices are within a reasonable range, and the frequency of sending the profile can be adjusted based on an administrator's risk profile. We found in both cases that there is negligible performance impact caused by the MDM agent and the context communication with the CM.

VII. Conclusions and Future Work

Through a review of the literature related to ZTA and BYOD, we have identified several exciting research challenges and opportunities related to enabling ZTA-based BYOD use cases. The unique requirements derived from our use cases show a clear need for a novel set of policy languages that allow high-level business intent to be rendered down to low-level enforcement policy. The surrounding infrastructure that must be built out can still be composed of specific technologies for the various portions of an access control flow. This is, of course, until solutions exist that can bridge the gap in standardized communication and control interfaces within this broader ZTA infrastructure.

To meet these needs, we introduced the BYOZ language specification. We also designed a zero trust network architecture and system to meet the exacting needs of enterprise BYOD use cases. Our architecture builds upon the NIST ZTA standard and incorporates novel ideas from continuous authentication & authorization to protect against specific enterprise BYOD threats. We showed in our evaluation that while our architecture on actual BYOD devices is MDM agent-based, the performance overhead is negligible, making our system feasible in existing enterprise environments.

There are many open areas to explore within ZTA, including applications for IoT, cloud computing, and 5G/Next-G cellular networks. Plus, we previously pointed out possibilities for moving our implementation into the realm of SDN. We are interested in the interactions of the CM and external systems to provide additional context into the zero trust environment. We also want to call out future work in the space of privacy concerns for ZTA due to the nature of metadata that is collected to process policy evaluations. Of keen interest to our research are the privacy concerns related to zero trust, since it is clear that a great deal of user metadata is required to make effective policy decisions.

ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation (NSF) under Grant No. 2114920, 2031002, 2129164, 2114982, 2226339, 2120369, 2128107.

REFERENCES

[1] Waldman, Arielle, "Remote work increases demand for zero-trust security", https://searchsecurity.techtarget.com/news/252498838/Remote-work-increases-demand-for-zero-trust-security, 2021
[2] Ogden, Jacqueline von, "The 7 Scariest BYOD Security Risks (and How to Mitigate Them!)", https://www.cimcor.com/blog/7-scariest-byod-security-risks-how-to-mitigate, 2017
[3] Warner, Jeannie, "What is Zero Trust Security?", https://www.crowdstrike.com/cybersecurity-101/zero-trust-security/, 2021
[4] Palo Alto Networks, "What is a Zero Trust Architecture", https://www.paloaltonetworks.com/cyberpedia/what-is-a-zero-trust-architecture, 2021
[5] Rose, Scott, et al. Zero trust architecture. No. NIST Special Publication (SP) 800-207 (Draft). National Institute of Standards and Technology, 2019.
[6] Oehler, Andy, "Zero Trust Network Access offers benefits that will make it the future of secure remote work", https://www.scmagazine.com/perspectives/zero-trust-network-access-offers-benefits-that-will-make-it-the-future-of-secure-remote-working/, 2021
[7] Hong, Sungmin, et al. "Towards SDN-Defined Programmable BYOD (Bring Your Own Device) Security." NDSS. 2016.
[8] Kang, Qiao, et al. "Programmable In-Network Security for Context-aware BYOD Policies." 29th USENIX Security Symposium (USENIX Security 20). 2020.
[9] https://www.cisco.com/c/en/us/support/docs/security/ios-firewall/23602-confaccesslists.html
[10] https://www.juniper.net/documentation/us/en/software/junos/routing-policy/topics/ref/statement/policy-statement-edit-policy-options.html
[11] https://docs.paloaltonetworks.com/pan-os/9-0/pan-os-admin/policy/security-policy
[12] https://www.netfilter.org/projects/iptables/index.html
[13] Pillay, Ashwin, et al. "Does BYOD increase risks or drive benefits?." (2013).
[14] Olalere, Morufu, et al. "A review of bring your own device on security issues." Sage Open 5.2 (2015): 2158244015580372.
[15] Marjanovic, Zoran. "Effectiveness of security controls in BYOD environments." (2013).
[16] Miller, Courtney, et al. "" How Was Your Weekend?" Software Development Teams Working From Home During COVID-19." arXiv preprint arXiv:2101.05877 (2021).
[17] Barlette, Yves, Annabelle Jaouen, and Paméla Baillette. "Bring Your Own Device (BYOD) as reversed IT adoption: Insights into managers' coping strategies." International journal of information management 56 (2021): 102212.
[18] Mehraj, Saima, and M. Tariq Banday. "Establishing a Zero Trust Strategy in Cloud Computing Environment." 2020 International Conference on Computer Communication and Informatics (ICCCI). IEEE, 2020.
[19] Bobbert, Yuri, and Jeroen Scheerder. "Zero Trust Validation: From Practical Approaches to Theory."
[20] Kerman, Alper, et al. "Implementing A Zero Trust Architecture." The MITRE Corporation, Tech. Rep (2020).
[21] Li, Xun, et al. "Sapper: A language for hardware-level security policy enforcement." Proceedings of the 19th international conference on Architectural support for programming languages and operating systems. 2014.
[22] https://www.businessinsider.com/solarwinds-hack-explained-government-agencies-cyber-security-2020-12
[23] Vashisth, Akanksha, and Avinash Kumar. "Corporate espionage: The insider threat." Business information review 30.2 (2013): 83-90.
[24] Droms, Ralph. "RFC2131: Dynamic Host Configuration Protocol." (1997).
[25] El Maliki, Tewfiq, and Jean-Marc Seigneur. "A survey of user-centric identity management technologies." The International Conference on Emerging Security Information, Systems, and Technologies (SECURE-WARE 2007). IEEE, 2007.
[26] DeCusatis, Casimer, et al. "Implementing zero trust cloud networks with transport access control and first packet authentication." 2016 IEEE International Conference on Smart Cloud (SmartCloud). IEEE, 2016.
[27] Eidle, Dayna, et al. "Autonomic security for zero trust networks." 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON). IEEE, 2017.
[28] "Embracing a Zero Trust Security Model" https://media.defense.gov/2021/Feb/25/2002588479/-1/-1/0/CSI_EMBRACING_ZT_SECURITY_MODEL_UOO115131-21.PDF
[29] Kindervag, John. "Build security into your network's dna: The zero trust network architecture." Forrester Research Inc (2010): 1-26.
[30] Vanickis, Romans, et al. "Access control policy enforcement for zero-trust-networking." 2018 29th Irish Signals and Systems Conference (ISSC). IEEE, 2018.
[31] Chen, H., Li, Hoang, T., Lou, X. (2013) Security challenges of BYOD: a security education, training and awareness perspective. The University of Melbourne, Australia. Pp. 1-8.
[32] McGillicuddy, Shamus, "Survey: Zero Trust benefits remote work during pandemic", https://www.networkworld.com/article/3587598/survey-zero-trust-benefits-remote-work-during-pandemic.html, 2020
[33] Downer, Kathleen, and Maumita Bhattacharya. "BYOD security: A new business challenge." 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity). IEEE, 2015.
[34] Miller, Keith W., Jeffrey Voas, and George F. Hurlburt. "BYOD: Security and privacy considerations." It Professional 14.5 (2012): 53-55.
[35] Thomson, Gordon. "BYOD: enabling the chaos." Network security 2012.2 (2012): 5-8.
[36] Scott, Ben, Raina Mason, and Patryk Szewczyk. "A snapshot analysis of publicly available BYOD policies." 2021 Australasian Computer Science Week Multiconference. 2021.
[37] Baillette, Paméla, and Yves Barlette. "Coping strategies and paradoxes related to BYOD information security threats in France." Research Anthology on Securing Mobile Technologies and Applications. IGI Global, 2021. 527-558.
[38] Barlette, Yves, Annabelle Jaouen, and Paméla Baillette. "Bring Your Own Device (BYOD) as reversed IT adoption: Insights into managers' coping strategies." International journal of information management 56 (2021): 102212.
[39] Boucher, Magdalena, et al. "BYOD-Bringing Your Own Device into Single-Surface Interaction Models." Proceedings of the Fifteenth International Conference on Tangible, Embedded, and Embodied Interaction. 2021.
[40] Ofusori, Lizzy Oluwatoyin, and Prabhakar Rontala Subramaniam. "The Influence of Technical and Social Factors in Mitigating Threats in a BYOD-Enabled Environment." International Journal of Information Systems in the Service Sector (IJISSS) 13.1 (2021): 1-30.
[41] McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM computer communication review 38.2 (2008): 69-74.
[42] https://linux.die.net/man/8/iptables
[43] https://github.com/akopytov/sysbench
[44] https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/
[45] https://mqtt.org/
[46] https://www.geekbench.com/