International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

Threats and Vulnerabilities to IoT End Devices Architecture and suggested remedies

Prateek Mishra, Sanjay Kumar Yadav

Abstract : Due to decentralization of Internet of Things(IoT) applications and anything, anytime, anywhere connectivity has increased burden of data processing and decision making at IoT end devices. This overhead initiated new bugs and vulnerabilities thus security threats are emerging and presenting new challenges on these end devices. IoT End Devices rely on Trusted Execution Environments (TEEs) by implementing Root of trust (RoT) as soon as power is on thus forming Chain of trust (CoT) to ensure authenticity, integrity and confidentiality of every bit and byte of Trusted Computing Base (TCB) but due to un-trusted external world connectivity and security flaws such as Spectre and meltdown vulnerabilities present in the TCB of TEE has made CoT unstable and whole TEE are being misutilized. This paper suggests remedial solutions for the threats arising due to bugs and vulnerabilities present in the different components of TCB so as to ensure the stable CoT resulting into robust TEE.

Keywords: Internet of Things, IoT End Devices, RoT, Bugs, Vulnerabilities, TCB, CoT, TEE

I. INTRODUCTION

An Internet of Things (IoT) interconnects everything e.g. physical devices and their associated software or virtual application layer for information exchange with the existing and upcoming network technologies with anything, anytime, anywhere dynamically. The things that can be connected have actuating and sensing capabilities. Anything, anytime, anywhere dynamic connectivity and actuating and sensing features of the things are a major threat for confidentiality, authenticity and integrity of both data and services[4]. Zhang et al.[14] explains an IoT architecture that comprises of cloud the core, fog the extended cloud and edge layers. The Cloud consists of high-processing power systems and high storage capabilities for decision making (such as data relay, warehousing and big data analytics). Cloud lies at the highest level and is at the remote from edge devices. The Fog layer lies between the cloud and the edge/end devices. The systems at Fog are independent and have less processing and storage abilities than cloud but high processing and storage abilities than edge devices. Fog layer is very near to end devices than the Cloud. Lowest layer is the End/Edge devices layer comprises of interconnected smart devices having sensing , actuating and communication capacity , such as smart phones, smart TV's. End/Edge devices are the resource-constrained physical
devices in terms of very low processing, low storage and low powered battery. Edge devices are equipped with sensors, actuators and homes of local data. The two way communication between The Edge layer and the cloud layer happens through the Fog layer.

With the growth of IoT data processing and decision making has been transferred from cloud to the edge. Due to this increased overhead of data processing and decision making, IoT edge devices has been more vulnerable and visible to the security attacks [7][13]. IoT architecture has been presented below in Fig.1.

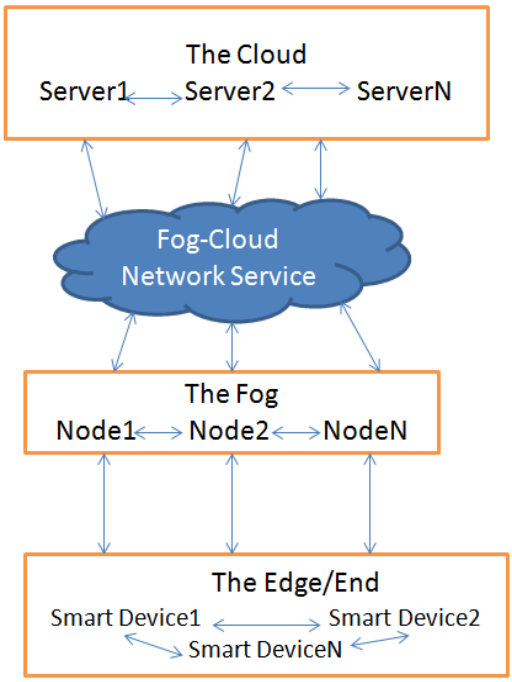

Fig. 1. An IoT architecture

Manuscript received on March 15, 2020.
Revised Manuscript received on March 24, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Prateek Mishra*, Department of Computer Science and Information Technology, Sam Higginbottom University of Agriculture, Technology and Science (SHUATS), Allahabad, UP. India-211007, prateekmishra25@rediffmail.com
Sanjay Kumar Yadav, Department of Computer Science and Information Technology, Sam Higginbottom University of Agriculture, Technology and Science (SHUATS), Allahabad, UP. India--211007. yadav_sk@rediffmail.com

II. HYPERVISOR BASED ARCHITECTURE

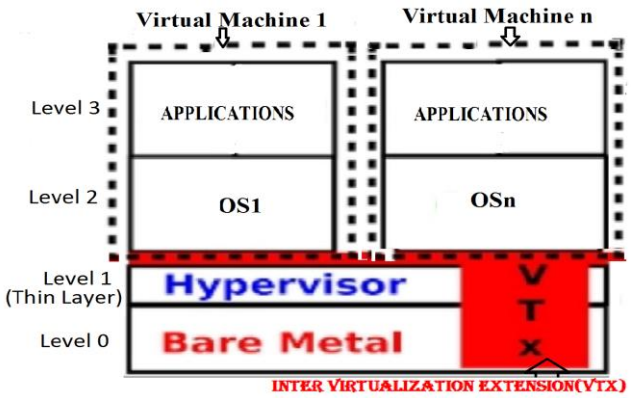

Fig.2 (a). Type-1: Hypervisor based end device

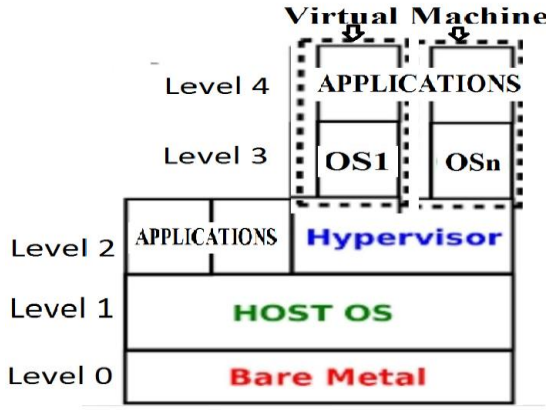

Fig. 2(b). Type-2: Hypervisor based end device

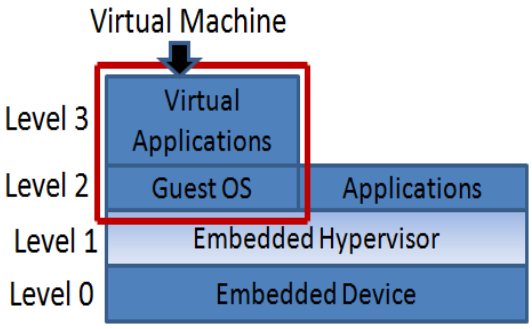

Fig. 2 (c) Embedded Hypervisor based end device[3]

Fig.2 (a) Type-1: Hypervisor based, (b) Type-2: Hypervisor based, (c) Embedded Hypervisor based end device architecture

(c) Embedded Hypervisor based End Device Architecture

Fig. 2 shows the hypervisor based layered architecture of IoT end devices. The complexity level of IoT End Devices allows the application of different components at different levels in the architecture. The compulsory bottom Bare Metal layer in Fig. 2 is the hardware layer at Level 0 that contributes as an I/O, processing and storage. Fig.2 (a) is having thin layer of Hypervisor software at Level 1 hence named as Type-1 Hypervisor, directly connected to the bare metal layer. Type-1 Hypervisor allows the exposition of Inter virtualization extension technology (VTx) to the guest operating systems at Level 2. It means guest operating systems above Type-1 Hypervisor shares the hardware resources CPU and RAM under the rule of virtualization. Type one Hypervisor may in principle allows running N operating systems above itself at Level 2 and in turn every Operating system may allow running N applications. Every operating system and its applications form virtual machine. Thus Type-1 Hypervisor may allow running N- Virtual machines on top of itself at Level 2 & Level 3. Due to minimum performance implications Type-1 Hypervisors are preferred in embedded devices [8].

Unlike Fig 2(a), Fig 2(b) illustrates Type-2 hypervisor based IoT end device architecture having host OS at Level 1 but at Level 2 operating system allows to run N applications along with hypervisor. Hypervisor in turn allows running N operating systems and every operating system in turn allows running various applications thus form N- Virtual Machines at Level 3 and Level 4.

If Host OS here is crashed everything above it will crash therefore Type-2: Hypervisors based architecture has been never a better choice for secured IoT end devices. On the other hand Type 1 hypervisor have better performance and greater flexibility due to the less overhead involved in running the hypervisor itself. In contrast Type 2 hypervisors runs on host operating system thus increases the processing time and thereby leading to a higher overhead. Typically, a Type 1 hypervisor is more efficient than a Type 2 hypervisor [8].

Fig 2(c) represents end device architecture based on Embedded Hypervisor. An Embedded Hypervisor is basically Type 1 hypervisor especially designed to meet the needs of resource-constrained nature of embedded systems environment and real-time operating system environments [3]. The Type 1 & Type 2 hypervisors are lacking support for real-time and memory requirements, making them incompatible or useless for embedded devices[19]. Unlike traditional Type 1 hypervisor, the embedded hypervisors are directly embedded/insulated into the hardware and run without a fully fledged operating system on the device where they are installed so as to allow also to run applications directly on hypervisor itself. Embedded hypervisors are compatibles with processors, system drivers and access system resources without any mediation and strong security against any attack. The basic requirements of embedded hypervisors include i. Small size and sleek implementation. ii. Support for independent execution but interactive coexistence of applications in a secure environment. iii. Low latency communication and switching between system components. iv. Being able to meet real time requirements of embedded applications and minimum affect on the performance. v. Minimum affect on system resources and fast response time[8].

A. Role of Hypervisor:

Every hypervisor have the following mandatory responsibilities

i. To create the virtual machine ii. To allow the execution of virtual machine after authentication iii. To monitor the virtual machine for malicious action and if such activity occurs hypervisor comes into action deny further execution. iv. To provide spatial isolation among VMs so as to avoid interference during virtual machine execution [18].. v.

Correctly distributes processor time among VMs to guarantee temporal isolation. vi. Hypervisor implements secure inter-VM communication channel for sensitive information between VM – VM and VM – Hypervisor communication. Every VM has a functionality called hypercall. Hypercall can call kernel's applications programming interface to implement it. The hypervisor can check the number and size of the message and also can copy message from source application to destination application[18].

III. VULNERABILITIES AND SUGGESTED REMEDIES

A. Bulky TCB: A Threat to Security

Firmware source code, trusted OS and Trusted Applications(TAs) make TCB size bulky and extra TAs un-added in the firmware if necessary can also be dynamically loaded which again makes the size more bulky. This bulky area grows thus increases complexity and difficult to find bugs and more area exposed to the attack therefore unstable chain of trust threat to TEE.

Suggested Remedy : Trusted computing Base(TCB) must be minimized to enhance security.

Hypervisor also known as virtual machine monitor , trusted execution environment Operating System and TEE Applications forms Trusted computing Base(TCB). Hypervisor is a thin software layer mainly responsible for creation of virtual machine and also act as a virtual machine monitor. The complexity of code decides the Hypervisor's efficiency and complexity is measured in terms of minimum space and minimum time of execution hence provides minimal overhead. A bulky hypervisor may have more vulnerability the tougher to find defects and more area also responsible for overhead in virtualization. Therefore code line must be kept minimum possible to make it simpler also to minimize the attack surface area. In future hypervisor can be replaced by containerization since it makes TEE lighter and suitable for low resource IoT end devices [6][20] claims that Intel® SGX have TEE with minimized TCB to makes it optimum secure.

Iqbal, A et al.[2] mentions that Microvisor is another technology solves the problem of Hypervisor's minimal overhead for virtualization and minimal size e.g. The OKL4 microvisor . Add-ons added to the hypervisor for additional services must have quality certifications to avoid any type of overhead such as performance, size or latency. Operating Systems must have light weight to be accommodated in resource constraints IoT end device since low processing power, low memory and low powered battery are big issues in resource constraints devices. TEE applications size should also be standardized to the minimum possible size so that it could be accommodated in resource constraints IoT end device with minimum latency.

B. Security Certification to TEE Components

Virtual Applications must have trust certificate that must be verified by attestation checker for authentication whenever systems boot before loading by the operating system. Its integrity must also be checked during execution by guest OS. Virtualized guest Operating system must also have trust certificate that must be authenticated by hypervisor if successful then only OS loader should be allowed to load OS. Confidentiality and integrity of data must be protected during execution by hypervisor in case of security breach hypervisor must come into action to discard virtual OS. No hypervisor till date have any international security certification. Steps must be taken to make it mandatory for the security certification for the hypervisor. Add-ons added to the hypervisor must also be attested by security certification to ensure authentication.

C. Trusted Applications Mapping to the Unsecure Physical Memory

IoT End devices where TEE and Unsecure environment co-exists . TEE exposes a trusted OS system call that allows any Trusted Applications to map any physical memory belonging to the Unsecure environment and TA can be hacked here by an attacker by scanning the physical address space thus backdoor entry can done[17].

Suggested Remedy: There should be a attack proof kernel isolation mechanism so that two processes could not overlap, the trusted applications could not go out of bound, the two processes could not try to run applications on the same address space and secured process for isolated execution for secure communication channel or security protected I/O communication path to protect the integrity of Trusted Applications. When trusted Applications Map to the unsecure physical memory following care must be taken. 1. TEE must have trusted application checker to check whether any unsecure process is running in unsecure environment, if it is all the unsecure processes must be halted all the vulnerable entry points must be closed. 2. TEE must have to create a secure channel. through which TA should enter into the unsecure world. 3. After the completion of task TEE must have a mechanism to delete all the data computed in the kernel of the secure world. 4. Finally to optimize the security whole environment must be trusted execution environment rather than secure and unsecure part.

D. Vulnerable Resources of IoT End Devices

Vulnerable resources are the known weak hardware and software resources of the devices that can be exploited. If we scan from the top to bottom in the IoT End Devices architecture the Top layer Virtual machines of IoT end devices architecture comprised of applications and Operating systems are the most vulnerable and unsecured components in all the IoT end device architecture since it is directly exposed to the un-trusted external world therefore the vulnerable virtual machine can easily hacked and crashed or data may easily be leaked .

Cheruvu et al.[20] mentioned that VM can be halted or aborted by Denial of Service(DoS) attack therefore may be rejected by hypervisor for further operation.

DoS deny the access of VMs to all the resources of the edge devices if system does not boot again, violating device mediation. DoS attack can also consume resources, like network socket handles, resulting in other VMs not being able to acquire the resource necessary to execute a function. Virtualized drivers and VM tools have usual vulnerabilities may try to violate memory separation and get access memory outside of its logical memory space .

Suggested Remedy : One should avoid Direct Execution of Commands from Guest VMs to prevent malicious software otherwise it may cause Stack smashing, heap smashing, and use-after-free vulnerabilities, that allows an attacker to execute their own code on the platform and allows the attacker to become a privileged user. Thus VM can cheat the \faith of hypervisor resulting into an execution separation violation and memory separation violation. The more the number of VMs the more will be attack surface area of VMs by untrusted world hence number of VMs must be suitably decided and cryptographically authenticated.

The larger the number of un-trusted IoT devices linked virtualized system, the higher the risk of vulnerabilities presents endangered threats to the hypervisor.

E. Discarding Un-trusted IoT End Devices

Attestation and root-of-trust are used as fundamental tools to discard risk before connecting with another IoT end devices. The attestation verification process must be implemented at the boundary line of trusted and untrusted IoT end devices to differentiate between trusted and untrusted. All the unknown , unverified and unattested devices are “untrusted,” while the known verified and third party attested devices are “trusted.”

The new unknown device must have robust “root-of-trust” environment containing a manufacturer embedded attestation key to attest trustworthiness of the device so that the device might provide proof of its trustworthiness. A Robust trust gotiation mechanism must be implemented using an attestation protocols that allow the root-of-trust to prove to a verifier that it is capable of protecting secrets, identities, and data.

Hence an untrusted IoT end device trying to connect first must attests to its level of trustworthiness and must have certificate of approval from quality control agency and receive a statement of approval that could be included with the attestation exchange at on boarding. Otherwise, untrusted device must be discarded . Thus attestation process must be able discard risk by discarding untrusted nodes. Attestation must have database updated risk profile.

Therefore, a proposed algorithm has been given below is must to discard an unknown and untrusted device without trust certificate so as to guarantee the confidentiality, integrity , authenticity of a IoT end device as a first step before connection. If any attack succeeds the attestation check must be able to detect it and must provide real time response to nullify.

Algorithm : Discarding untrusted device and attack
If(Untrusted device)
Either device identity known or not known
If the device has a trust certificate then
Verify certificate
If Verification succeed
Device identity verified then allows to connect
Update risk profile
Else
Discard the Untrusted device
Update risk profile
If(Untrusted device)
Either device identity known or not know
If attack succeeds then attestation check
Attack detected then
Attack nullified

F. Hypervisor’s Vulnerabilities

Fig. 2. illustrates that Hypervisor is the core to virtualization in every end device architecture. No Hypervisors have any security certifications till date. The more the code line of a hypervisor the more complex it will be and the more vulnerabilities it will have the more tougher to find defects. Hypervisor also allow add-ons for additional services, like management and configuration. These add-ons can have additional vulnerabilities. Cheating a hypervisor is possible by Spectre and meltdown vulnerabilities resulting into leakage of confidential data to guest applications

. The device drivers may also be corrupted by hackers if unauthenticated or device drivers can be scanned that leads towards security breach and can be threat to the confidentiality and integrity [20] .USB device drivers provide large area for attack.

Suggested Remedy: The lesser the number of codes in hypervisor the simpler it will be and easier to find defects and lesser will be the attack surface area. The followings are the responsibilities of the hypervisor if i. A VM tries to access memory outside the address space and violate spatial isolation. A hypervisor must take action to implement spatial isolation mechanism ii. The hash signature of virtual machine does not match with stored public key supplied by the 3rd party during the chain of trust phase then VM must be discarded ..iii. A VM spoils the TEE then hypervisor must discard that VM also must send warning message to make all the components aware and immediate action.

The device drivers must be authenticated for trusted source before installation .must be made attack proof using cryptographic algorithm so as avoid scanning the device drivers mechanism. Authentication and trust must be verified at the port level before entering into the system. USB device drivers must be cryptographically protected to ensure protection and avoid security breach. The host OS of Type-2 hypervisor based architecture presented in Fig 2(b) has a burden of hypervisors and guest operating systems and other virtual machine applications. This burden may leads towards annihilation of all the layers including host OS and other upward layers above it and therefore is considered as highly unsecured architecture. Applications depicted on Layer 2 of Type-2: Hypervisors in Fig 2(b) are in direct insecure communication with host OS of Level 1 and makes the environment unsecure. `The presence of insecure communications among virtual machines also presents threats for data leak, integrity and authentication, confidentiality. Type-2 hypervisor based architecture being highly unsecure is highly desirable and preference should be given to Type – 1 or embedded hypervisor.

G. Processor’s Vulnerabilities

At the start of 2018 Spectre and melt down(S&M) Vulnerabilities were found in Intel and AMD processors. According to M.S. Kerner[15] these vulnerabilities were mis-utilized by attackers to get access of speculative execution contained in the processors and read the data contained therein. A DoS attack on a virtualized hardware device represents a violation of execution separation.

Suggested Remedies: According to M.S. Kerner[15] followings steps must be taken to prevent confidentiality of data

1. Patch management i.e. a combination of micro-code patches from CPU vendors and OS patches to prevent confidentiality.
2. Active monitoring of logs in security information and event management system,
3. Encryption of data will make the data difficult of read after stealing.
4. Endpoint detection, response , user and entity behaviour analytics technologies can recognize unusual activities that could be indicative of cyber attacks.

IV. BUGS ISSUES

A. Validation Bugs[15] [16]

These are software overflow bugs in TEE due to configuration problem, programming errors, mishandling of input and/or output values known as validation bugs. e.g. Buffer overflows, incorrect parameter validation, mishandled integer overflows, etc. These Bugs are well known and can easily be exploited by a hacker for unauthorized access of secured TCB components. These types of bugs are mostly present in almost all TCB components. Suggested Remedies: These are software bugs must be taken care during program development e.g., buffer overflows can be handled using security measures in code or languages offering built-in protections or using run time protection mechanism of OS such as structured exception handler overwrite protection, data execution protections. etc. All such overflows can be handled just by allowing legitimate traffic and preventing illegitimate traffic.

B. Functional Bugs

These are the errors in codes due erroneous coding if implementation is not exactly as per algorithm. (e.g. incorrect coding of a cryptographic algorithm). Functional Bugs may introduce security vulnerabilities in the memory protection mechanisms of a TEE system [17]. Suggested Remedies: If inconsistencies are between implementation and the program specification it means it is the fault of tester since the unawareness or bad intention of programmer but be identified during function testing. This carelessness in testing leads towards negative consequences of incorrectly programming of a cryptographic algorithm in security breach since the created code could not perform the desired task of security.

C. External Bugs

Bugs that takes place due to mis-handling of external factors that might introduce security breach in TEE such as bugs in concurrent executions of multiple threads caused by the interference of multiple concurrent programs e.g. In concurrent access of a file system a directory created by previously executed TAs on trusted storage can be deleted by TA executed just next. [17]. Suggested Remedy: Since concurrent access to the file system by different TAs is a critical region and might be solved by the application of concurrent programming control since at any moment one and only one of them is engaged in the "critical section"[1] to avoid security concern such as deletion of a directory on trusted storage by another TA. There must be another program that must keep the integrity value of the authenticated directory so that current TA must authenticate before deletion .

D. Information Leakage from Caches

On TrustZone supported processors due to equal permission of accessing cache lines cache contention begins Zhang et al.[9]. Due to this contention information can be leaked from the SW by monitoring caches from the Unsecure world. Guanciale et al.[10] implemented a low-noise cache storage channel which can successfully extract a 128-bit key from an AES encryption service.

Suggested Remedy: Cryptographic algorithms can be applied in software[10]to prevent the leakage of confidentiality. Cache maintenance techniques also protects from information leakage via caches. Cache partitioning deprives a hacker to take the advantage of contention with victim [11][9].

V. UNSTABLE CHAIN OF TRUST:

From the above observations it has been found that Vulnerabilities exists in all the components of IoT end device architecture. Starting from the bottom of the architecture i.e. from Level0 to the top at Level3 vulnerabilities makes TCB vulnerable and un-trusted and the chain of trust becomes weaker hence trusted execution environment becomes vulnerable and un-trusted. These vulnerable hardware and software resources can be exploited by the attacker resulting into data leak. In the beginning of 2018 Spectre and meltdown Vulnerabilities allows the attacker to steal the data from the memory currently processed in the modern processors such as Intel , ARM processors[12].

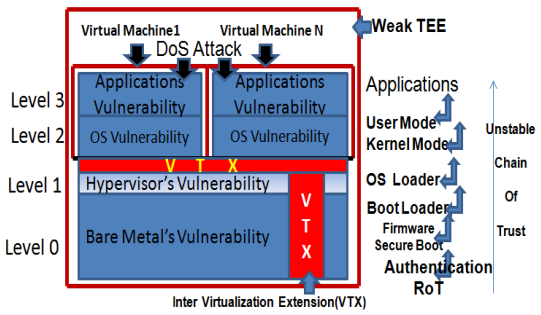

Fig. 3. IoT end devices with unstable Chain of Trust & weak TEE

VI. RESULTING PROPOSED ARCHITECTURE WITH ENHANCED SECURITY

According to Cheruvu et. al.[20]: Security can be categorized i. Security mechanism ii. Security assurance . Security mechanism deals uninterrupted temper resistance secure boot process, secure storage of symmetric and asymmetric keys and cryptographic algorithm execution where as security assurance ensures the protection of integrity , authenticity , confidentiality of data and keys . Trusted Execution Environment (TEE) is an isolated secured execution environment.

TEE is separated from the normal unsecured execution environment e.g. a security coprocessor is an isolated environment and the core CPU is a general-purpose execution environment. TEE implements cryptographic keys to ensure confidentiality and integrity protection of data in TCB as it is transformed to and from cipher text are performed e.g. ARM TrustZone isolated execution environment within the ARM core. [5] defines TEE as a tamper resistant processing environment that runs on a separation kernel. that comprises of Secure Boot , Secure Scheduling ,Secure Inter-Environment Communication , Secure Storage components with the objective of the authenticity of the running code, the integrity of the runtime states (e.g. CPU registers, memory and sensitive I/O), and the confidentiality of its code, data and runtime states stored on a persistent memory and Remote attestation for the trustworthiness for third-parties to protects against all software attacks and physical attacks performed on the main memory of the system.

Trusted Execution Environment(TEE) combines security functionality with security assurance mechanisms so that security risks could be discarded. One such security functionality is the *Root of Trust(rot) that must be built as soon as the power is on*. Low resource devices must have single unclonable roots-of-trust that must come into action as soon as the power is on. All hardwares, firmwares, softwares that form trusted execution environment are combinely called Trusted computing base (TCB). Bugs or vulnerabilities in TCB may collapse the system in one go. Therefore it is mandatory to authenticate every bit and byte of TCB to build stable chain of trust from the lowest hardware layer to the highest virtualized applications layer , before admitting into the system .Therefore for being security proof the TCB component must be equipped with the following security components and is summarized in the following Fig 4.

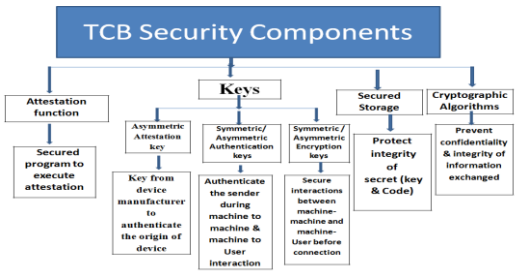

Fig. 4. Mandatory security components in TCB for Attestation of device origin , Integrity of keys & code ,

Authentication of the sender , Confidentiality of data and interaction.

R.T. Tiburski et. al.[18] mentioned that at the IoT end device a root public key is stored in the onetime read only memory. The first software after power on is primary boot loader in tamper resistant read only memory must have cryptographically signed manufacturer signature. This signature is the certificate from manufacturer . This signature must match with the root public key(i.e. authentication) stored in a write once memory to avoid key substitution hence to protect the integrity of keys. If match is found the booting process starts else system fails to boot. The same process will be repeated with the hypervisor . If successful, then the same process will take place in the next boot stage until top software. layer above the lower layers is authenticated thus a stable of the Chain of Trust (CoT) is built.

Thus authentication(i.e. signature matching with keys) must be done and integrity of code and keys must be preserved before every step of next boot process till the booting process complete thus leads towards a stable Chain of Trust(CoT) . Further every virtual application must have a signature(certificate) that must match with the root public key stored in the hypervisor . Hence authentication, integrity and confidentiality at every TCB ensure stable CoT and robust TEE.

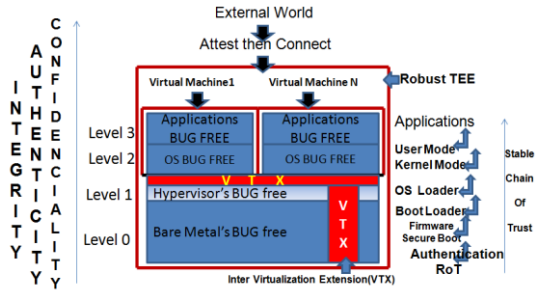

Fig. 5. IoT End device with stable CoT and Robust TEE

VII. COMPARISON WITH EXISTING APPROACHES

Earlier it was the perception that attacks volume is high to those components of the IoT end devices highly exposed to the external un-trusted world[20].Therefore there is a need to allow to connect only to the authenticated device . In the recent past the attacks on the IoT end devices internals have increased by exploiting the vulnerabilities present in the TCBs of the devices. R. T. Tiburski et al. [18] implemented the architecture for resource-constrained IoT end devices having MIPS32 processor core running at 200 MHz, with 2 MB of flash memory and a 512 KB SRAM to ensure integrity , authenticity, non-repudiation using TEE and embedded virtualization with total memory footprint of 319 KB, communication latency 99 micro second for message of size 256 B , and cryptographic performance but their architecture have no support for remote updates and secured terminal access and security threat due to internal bugs, hypervisor's vulnerability.

S. Pinto et al. [21] proposed an Enhanced TrustZone-based architecture named Industrial IoT Trusted Execution Environment for Edge Devices (IIoTEED) .They evaluated the architecture using a dual ARM Cortex-A9 running at 600 MHz and implemented i. secure boot process to confirm integrity at boot time. ii. Spatial and temporal isolation ensures confidentiality and availability. IIoTEED had some loop holes such as Confidentiality is partial since Trustzone architecture doesn't authenticate access to the resources and enable man in the middle attack and so manipulation of the messages transferred through the channel .

Side channel attacks are also out of scope of the this TrustZone specification. Integrity is also partial since Trustzone doesn't provide any hardware mechanism to ensure integrity of data over time. To guarantee tight industrial security IIoTEED must be complemented with other critical securities for edge devices(e.g.

hardware assisted device identity) to better leverage the three fundamental elements CIA. Dai et al.[22] proposed a TEE architecture that uses the Xen hypervisor to allow multiple VMs to enjoy Dynamic Root of Trust for Measurement(DRTM)-like secure execution environments. The tools used to evaluate TEE was Intel Core Duo processor running at 1.8 GHz and 2 GB RAM. They evaluated TEE's performance with an Intel Core Duo processor running at 1.8 GHz and 2 GB RAM. Times consumed to establish TEE Domain with one vCPU and 64 MB memory is 173 ms, the TEE kernel is of 1.30 MB, and the time consumed for encryption is 436.9 ms. DRTM's primary goal was to attest the authenticity of mother board platform and an O.S. and assures that an authentic O.S. starts in a trusted environment. It was tried in all these above proposed architectures to cover some aspects of security but no architecture till date could provide tight security, therefore lots of efforts needed to done to ensure authenticity , confidentially , integrity ***of every bit and byte of Trusted Computing Base(TCB) especially in resource constrained IoT end devices.*** Therefore more efforts are needed to ensure tight security against untrusted external world as well as against intentional or unintentional vulnerabilities with minimum possible memory footprint , communication latency.

The remedy suggested to make the TCB components free from bugs and vulnerabilities as well as *no connection without attestation with the external would make the security tight and enhance* authenticity , confidentially and integrity *of* every bit and byte of TCB and ensures stable chain of trust resulting into secured IoT end devices.

VIII. CONCLUSION

This paper focuses on threats due to vulnerabilities present in the components found in the TCB of TEE of IoT end devices. Earlier emphasis was given by researchers on i. the attack surface area of TCB 2. Building Chain of trust (CoT) to ensure integrity i.e. the code will remain un modified authenticity i.e. virtual software must have certificate 3. After authentication if VM into spoiling the environment, the responsible party cannot deny it (non-repudiation) 4. Spatial and temporal isolation security mechanism incurred by hypervisor. But with the advent of technology these security features were not sufficient such as Spectre and meltdown vulnerabilities involves data leak at run time. More new vulnerabilities were found in hypervisors at the code level, in caches , processors and all other TCB component from bottom to top made the chain of trust weak and presented new threat . Therefore some remedies(such as no connection before authentication with the external un-trusted world.) have been suggested to strengthen the root of trust followed by trusted computing base resulting into strong chain of trust consequently robust TEE.

REFERENCES

1. L. Lamport, "*A new solution of Dijkstra's concurrent programming problem ," Communications of the ACM,vol. 17, no. 8 , pp. 453–455.* 1974, doi:10.1145/361082.361093
2. Iqbal, A. N. Sadeque, and R. I. Mutia, "An Overview of Microkernel, Hypervisor and Microvisor Virtualization Approaches for Embedded Systems, " 2010.
3. M. Jones, "Virtualization for embedded systems The how and why of small-device hypervisors." developer.ibm.com. https://developer.ibm.com/technologies/linux/tutorials/l-embedded-virtualization/ (accessed Feb. 14, 2020).
4. **ITU-T , "** Overview of the Internet of Things ," International Telecommunication Union, June , 2012.[online]. Available: http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=11559 2012
5. M. Sabt, M. Achemlal, and A. Bouabdallah , "Trusted execution environment: what it is, and what it is not," IEEE Trustcom, vol. 1, pp. 57–64. , 2015 , DOI 10.1109/Trustcom-BigDataS
6. R. Morabito, J. Kjällman and M. Komu, "Hypervisors vs. Lightweight Virtualization: A Performance Comparison," 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, 2015, pp. 386-393.
7. R. Roman, J. Lopez and M. Mambo, "*Mobile Edge Computing A Survey and Analysis of Security Threats and Challenges," Future Generation Comp. Sys.*, vol. 78 , Part 2 pp. 680–698 , January 2016, doi: https://doi.org/10.1016/j.future.2016.11.009.
8. M. Mounika and C. N. Chinnaswamy, " A Comprehensive Review on Embedded Hypervisors ," *IJARCET, vol. 5,no. 5,pp. 1546 – 1550,May* 2016.
9. N. Zhang, H. Sun, K. Sun, W. Lou and Y. T. Hou, "CacheKit: Evading Memory Introspection Using Cache Incoherence," 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrucken, 2016, pp. 337-352.
10. R. Guanciale, H. Nemati, C. Baumann and M. Dam, "Cache Storage Channels: Alias-Driven Attacks and Verified Countermeasures," 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, 2016, pp. 38-55.
11. M. Lipp., D. Gruss, R. Spreitzer, C. Maurice and S. Mangard, "ARMageddon: Cache Attacks on Mobile Devices," August 2016 USENIX Conference on Security Symposium. USENIX Association, pp. 549–564.
12. N. Zhang, K. Sun, D. Shands, and Y. T. Hou,"TruSpy: Cache Side-Channel Information Leakage from the Secure World on ARM Devices," *IACR Cryptology ePrint Archive,vol 2016 ,pp.* 980- 995 .
13. A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng., "Fog Computing for the Internetof Things: Security and Privacy Issues," IEEE Internet.Comput., vol. 21, no. 2, pp. 34–42, Mar. 2017.
14. P. Zhang, M. Zhou and G. Fortino, " *Security and Trust Issues in Fog Computing: A Survey," Future Generation Comp.* Sys.,vol. 88, pp. 16–27, Nov 2018 , doi: https://doi.org/10.1016/j.future.2018.05.008
15. M.S. Kerner. "Protecting Against the 7 Vulnerabilities of Meltdown and Spectre," eSecurityplanet.com. https://www.esecurityplanet.com/applications/meltdown-and-spectre-vulnerabilities.html (accessed Feb. 28, 2020).
16. D. Berard. "Kinibi TEE: Trusted Application Exploitation," SYNACKTIV.comhttps://www.synacktiv.com/posts/exploit/kinibip-tee-trusted-application-exploitation.html (accessed Feb. 12, 2020).
17. D. Cerdeira, N. Santos, P. Fonseca and S. Pinto,"SoK: Understanding the Prevailing "Security Vulnerabilities in TrustZone-assisted TEE Systems," 2020 IEEE Symposium on Security and Privacy.
18. R. T. Tiburski, C. R. Moratelli, S. F. Johann, M. V. Neves, E. d. Matos, L. A. Amaral and F. Hessel., "Lightweight Security Architecture Based on Embedded Virtualization and Trust Mechanisms for IoT Edge Devices," in IEEE Communications Magazine, vol. 57, no. 2, pp. 67-73, February 2019.
19. C. R. Moratelli, R. T. Tiburski , E. d. Matos, G. Portal, S. F. Johann and F. Hessel, "Privacy and security of Internet of Things devices , " *Real-Time Data Analytics for Large Scale Sensor Data*, pp. 183-214 , 2020 .
20. S. Cheruvu, A. Kumar, N. .Smith, and D. Wheeler, "*Demystifying Internet of Things security*," Springer. 2020, doi :https://doi.org/10.1007/978-1-4842-2896-8
21. S. Pinto, T. Gomes, J. Pereira, J. Cabral, and A. Tavares,"IIoTEED: An Enhanced, Trusted Execution Environment for Industrial IoT Edge Devices," IEEE Internet. Comput., vol. 21, no. 1, pp. 40–47, Jan 2017.
22. W. Dai, H. Jin, D. Zou, S. Xu, W. Zheng, L. Shi and L. T. Yang, "TEE: A virtual DRTM based execution environment for secure cloud-end computing," Future Gener. Comput. Syst., vol. 49, pp. 47–57, 2015.

AUTHORS PROFILE

Prateek Mishra has received Master's degree in Computer Application from Mahatma Gandhi Kashi Vidyapith University, Varanasi and Master of Technology in Computer Science and Engineering from Shobhit University Meerut.He works as a Research Scholar in the Department of Computer Science & Information Technology(DCS&IT) at Sam Higginbottom University of Agriculture, Technology and Sciences(SHUATS), Allahabad, Uttar Pradesh, India. His areas of interest are Design & Analysis of Algorithm, Internet of Things, Network Security, Mobile Computing, Modeling and Simulation and Software Fault tolerance.

Dr. Sanjay Kumar Yadav has received Ph.D. degree in CSE, M.Tech in CSE. He works as an Associate .Professor in the Department of Computer Science and IT, Sam Higginbottom University of Agriculture, Technology and Sciences Allahabad, India. His areas of interest are IoT, Distributed Systems, Mobile Ad-hoc Networks, Data Mining.