Computer Networks and Communications
https://ojs.wiserpub.com/index.php/CNC

Article

Trust and Risk Assessment in IoT Networks

Jeffrey Hemmes[1,*], Steven Fulton[2], Judson Dressler[2], Stephen Kirkman[1]

[1] Department of Computer and Cyber Sciences, Anderson College of Business and Computing, Regis University, Denver, CO, USA
[2] Department of Computer and Cyber Sciences, United States Air Force Academy, Colorado Springs, CO, USA
E-mail: jhemmes@regis.edu

Received: 11 March 2023; **Revised:** 2 May 2023; **Accepted:** 18 May 2023

Abstract: The Internet of Things (IoT) is a large-scale, heterogeneous ecosystem of connected devices encompassing a range of purposes and computing capabilities. As IoT systems grow ubiquitous, new approaches to security are needed. This work proposes a method of risk assessment for devices that combines the use of trust models based on dynamic behaviors with static capability profiles drawn from immutable device characteristics to determine the level of risk each device poses to network security. A risk-based approach allows security mechanisms and monitoring activities to be more efficiently allocated across IoT networks. Simultaneously, devices can be allowed a greater degree of functionality while ensuring system availability and security. This paper presents a methodology and architecture to integrate risk assessment into IoT networks. This allows additional tailoring of security control application and provides higher-level, more human-readable information for security analysts.

Keywords: Internet of Things, IoT security, risk, trust, trust models, device characterization, device fingerprinting, device profiling, access control

1. Introduction

The Internet of Things (IoT) has seen increasingly widespread adoption across a diverse range of applications, from industrial and commercial to military to home systems and beyond. As a consequence, new security challenges specific to the unique characteristics of IoT must be addressed such that future disruptions to critical systems and data are prevented.

The large number of networked devices in IoT networks has the potential to overwhelm the analytical and data presentation capabilities of existing security tools. Traditional security monitoring tools collect and examine individual network packets, which is an effective way of detecting suspicious behavior. However, the large volumes of data and alerts generated by such tools frequently pose challenges for security analysts. Monitoring tools that allow for greater organization and prioritization of data are a necessary part of future IoT security.

Existing security tools rely on sensor information typically collected at a centralized location via mechanisms such as network taps, with data analyzed offline using intrusion detection systems. Security analysts rely on dashboard-type applications such as Sguil or Kibana to manually review and assess the raw data and any corresponding alerts drawn from it. With a significant number of devices each generating substantial amounts of network traffic, the resultant volumes of data are difficult to manually review even with the availability of automated collection and presentation tools. Moreover, current tools are only minimally effective at synthesizing raw sensor or alert data into higher-level information regarding the nature of the data sources. Obtaining a more complete picture of the overall security of a particular network relies heavily on analyst

DOI: https://doi.org/10.37256/cnc.1120232667

knowledge and skill. This limitation introduces a greater possibility of human error that demands an improved set of automated security tools. This work is intended as a step towards addressing that gap.

The objective of this work is to create a balance between flexibility with both functionality and efficiency in security considerations within the design of IoT systems. If administrators can focus security mechanisms and monitoring activity on devices inherently possessing a greater capability of causing harm on the network, then such a balance can be struck while still affording robust protection of the network and network resources. Towards this end, this work examines the use of dynamic trust evaluations and risk assessments in IoT security.

1.1 *Threats to IoT Networks*

One common threat against IoT networks is distributed denial-of-service (DDoS) attacks. A noteworthy example of a successful DDoS attack against IoT networks is the Mirai botnet attack of 2016, which effectively attacked and disrupted more than 1 million devices [1]. This attack demonstrated that IoT systems can have a wide range of exploitable vulnerabilities due to their heterogeneous nature. Furthermore, the more sophisticated the device, the greater the attack surface due to missing patches, and therefore the more software and firmware components which may not be up-to-date. The implication of this observation is that devices with greater sophistication and computing capability may pose a greater risk of successful exploitation. Identifying devices with a greater capability to be used as command-and-control nodes in DDoS attacks could allow additional preventive measures to be taken, as just one example. At a minimum, greater monitoring emphasis, as reflected in security dashboard applications, could be placed on devices which might pose a more significant security threat.

2. Related Work

This work synthesizes a number of techniques for characterizing and assessing devices and their behavior. This section presents and discusses the most significant existing works related to our risk modeling protocol and architecture.

2.1 *Trust Models*

Underlying the assessment of risk posed by network nodes is the notion of trust. The use of security models based on trust in the context of mobile IoT networks has been well established and acknowledged in the literature, e.g., [2]. Consequently, the application of trust models in security protocols and mechanisms is a recurring element in wireless network security [3–13]. Computational trust for IoT appears in a significant number of works, generally taking on one of several distinct forms. Historically, this divergence is attributable to a lack of community consensus of what trust means at a fundamental level and what form it ought to take when integrated into IoT systems [7].

One approach to integrating trust into security architectures involves establishing a time-and-behavior based reputation using demonstrated actions or communication patterns, typically handled in a decentralized fashion. In such an approach, nodes assess and formulate measures of trustworthiness for each directly connected neighbor based on pair-wise interactions. Trustworthiness of nodes that are more distant can be evaluated indirectly via trust evaluations received from other nodes already deemed trustworthy. With general trust models, trust is most commonly the sole decision-making criterion for routing or access control. For instance, in [6] adaptive network protection is based on the computed trust level of devices, which is similar to the underlying use of trust in this work. Trust is also decentralized, with each node deciding locally whether to authenticate messages received from neighboring nodes. In [8], trust is the singular criterion used when authenticating traffic as well. Many other works approach trust similarly [3,14–16]. One distinguishing characteristic of this work is that trust is not an isolated or standalone concept, nor is it the sole basis for any decision making.

2.2 *Trust and Relation to Threats*

An important use of trust models and dynamic behavior analysis in network security is detecting many types of threats, to include loading malware, exfiltrating data, or contributing to denial-of-service attacks. These can be detected when a device that purportedly provides a particular function or service begins generating

network traffic inconsistent with that function. Trust models are an effective way to assessing the extent to which such traffic aligns with a particular device function.

It is particularly important to be able to detect any attempt to exploit new vulnerabilities. A behavior-based trust model is very much compatible with this goal. This is because exploiting zero-day vulnerabilities often involves formatting packets in a manner inconsistent with a given device's purpose. Such an attack would manifest as anomalous behavior except in cases in which the attack originates from a more general-purpose computing device with a higher degree of capability. For this purpose, device fingerprinting can be used to identify those devices that may pose such a threat.

2.3 *Device Fingerprinting*

In addition to trust, this work relies on the use of device fingerprinting. We synthesize existing techniques that have been proposed and evaluated in other works into a novel application involving risk assessment rather than proposing new methods for assessing trust or fingerprinting networked devices.

The concepts involved with gathering, describing, and publishing both the immutable capabilities and dynamic state of computing devices have long been understood and employed within distributed computing systems. Distributed batch processing systems, for example, have often relied on profiles of various devices for matching batch assignments to machines available for computation based on whether they are utilized or idle when the jobs run. The ClassAd used in the Condor distributed system [17] is an abstract representation of the hardware and software platform configuration of each compute node that is part of the system. Each machine that is ready to run batch jobs publishes its ClassAd through the Condor framework. Although there are similarities between ClassAds in Condor and the device profiles used for risk assessment here, Condor is intended exclusively for use in computational grid systems. In grid or cluster computing systems, machines are generally similar, if not identical, in their configuration and capabilities. Moreover, grid systems are also organized and deployed within the boundaries of particular organizational domains that are each managed by a separate administrator. Having unknown and untrusted devices join dynamically is not a common occurrence within these types of systems.

In addition to hardware, the profiling of a machine's software configuration has been common practice for many years. The detection of web browser and operating system type and version using information collected from either active or passive techniques has also been well explored. Fingerprinting techniques are categorized as either active or passive approaches based on whether a device is actively queried for information or if normal network packet transmissions are intercepted and evaluated without any device queries. Passive approaches tend to be more resilient to detection techniques and countermeasures that a malicious actor may use.

As an example of a passive fingerprinting approach, TCP/IP fingerprinting can identify different operating systems and operating system versions based on specific implementation details of TCP protocol parameters that differ among operating systems or vendors. However, such implementation differences can be exploited by malicious actors performing surveillance and reconnaissance actions in preparation for platform-specific attacks and other adverse uses [18]. Therefore, network defenses have evolved to protect specifically against these types of surveillance activities. Consequently, many traditional fingerprinting techniques are now ineffective or infeasible even when they are intended to be used defensively. More recently, device fingerprinting has been employed in detection of web browsers [19]. That work demonstrates that hardware characteristics are detectable using JavaScript, even in cases where multiple browsers are running concurrently on the same machine.

Device profiles also have been employed in IoT networks, to include home automation systems [20].

Lastly, device profiles also are commonly used for multimedia applications in which hardware capabilities and software configurations can be used for the purpose of performance tuning, e.g., [21]. In these instances, device profiling enhances functionality and performance, as well as in systems in which behavior is dependent on the specific characteristics of network components and architecture. Such applications create and use profiles consisting of information collected using protocols residing above the data link layer. This requires greater care be taken to assure that the integrity of gathered information is maintained.

Many physical-layer fingerprinting methods rely on variations that result from the hardware manufacturing process. These deviations can enable the unique identification of specific individual devices. This level of detection, in addition to being challenging to implement, is far more than what is required in the security architectures of the IoT deployments we envisage. Features detectable from the physical layer generally involve measuring the RF waveform of transmissions from the connecting device. As such, many hardware-based fingerprinting techniques require specialized hardware or software-defined radios to implement correctly. This

requirement makes such methods impractical or infeasible for systems composed largely of offthe-shelf, commodity or legacy components.

However, some hardware-based approaches can be more practical. There are numerous existing methods used for device fingerprinting that rely on details of the MAC-layer implementation that vary across vendors. Neumann et al. [22] evaluated the suitability and effectiveness of wireless network parameters for 802.11 device identification and concluded that the frame interarrival time metric is an effective parameter for fingerprinting. Moreover, they found that determinations could be made in a "reasonable" amount of time. However, they also noted that different services running on a device can have an effect on the measurements taken. With IoT networks, this limitation may be less of a concern as each device class is typically expected to run similar if not identical applications.

Ensuring the integrity of any profile information is a critical requirement in security mechanisms, but limitations of software profiling in IoT devices can lead to a high susceptibility of spoofing. As a consequence, fingerprinting methods for security purposes must be resistant to tampering and spoofing attacks. This requires collecting data from any of the physical, datalink, network, or transport layers in the OSI model. Techniques dependent on data obtained from layers above the transport layer are more limited in utility because an attacker can manipulate such data with much less difficulty.

There is an important exception to relying on data obtained from OSI layers 1 and 2 that is described in [23]. In that work, detection of operating system type and version can be accomplished securely using HTTP, TCP/IP, and more significantly, TLS. TCP/IP features that may be leveraged for fingerprinting include the TTL and window size values in the IP header, and additional options embedded within the SYN packet. Using TLS, the operating system is detectable using the ClientHello message along with the list of available cypher suite type codes. Fingerprinting using HTTP involves the User-Agent strings, which an attacker can forge with little difficulty. However, the TLS handshake alone can detect at a minimum the operating system major version with a success rate of approximately 80 percent. Significantly more accurate results can be achieved using additional network flows collected over a longer period of time.

An additional method that can be used to fingerprint hardware is proposed in BF-IoT [3], which depends on features unique to the Bluetooth Low Energy (BLE) protocol stack. Machine learning techniques are then used to characterize devices based on detectable differences in vendor-specific implementations. BF-IoT is most similar to this work based on the incorporation of a passive fingerprinting method into IoT networks. The BF-IoT architecture includes a fingerprinting engine. It is deployed in a distributed cloud environment, and it matches device signatures to whitelisted or blacklisted entries contained in a database. The IoT gateway then makes access control decisions based on the verified identity of a device. However, BF-IoT is limited to Bluetooth networks of devices that implement BLE and is not easily generalizable to a more heterogeneous IoT deployment.

The literature describes a range of complementary fingerprinting techniques using passive approaches such as features and network parameters of 802.11 implementations [22]. Fields in 802.11 frames used for both data transmission and network management purposes such as rate switching, random back-off timers, and duration field values can provide reliable and spoof-resistant device identification information. For other wireless protocols such as BLE as described earlier, similar passive fingerprinting techniques may be applied [3]. Moreover, BLE information can be spoofed using techniques described in [24].

Individual fingerprinting techniques may be used to classify each device by type based on only a single attribute. Alternatively, a more holistic approach using a combination of techniques, such as described in [16] can achieve greater fidelity and spoof-resistance in profiling devices. In any implementation, a combined fingerprinting approach is preferable due to the requirement for secure device characterization that must ensure device profiles are resistant to spoofing, forgery, or other manipulation by a malicious actor.

In addition to fingerprinting, trust models have had widespread application in wireless network security. The notion of trust in IoT networks appears frequently in the literature but takes on several distinct forms. Most trust models involve establishing and assessing reputation that is established over time based on demonstrated patterns of behavior that conform to expectations and is measured in a decentralized manner [6,8,11,14,16,25]. Such a fully decentralized approach may be incompatible with many IoT deployments, particularly those composed of legacy devices.

2.4 *Fingerprinting and Relation to Threats*

Fingerprinting can be used to determine the types of attacks a particular device may be capable of executing. This could be differentiating between a device capable of service as a command and control node and

a device that can relay packets as part of a botnet. Similarly, fingerprinting a device may determine whether it is capable of exfiltrating significant amounts of data, executing brute-force attacks on a server, and so on.

3. Trust Evaluation

Trust models are a commonly used security mechanism with a significant foundation in the literature, e.g., those surveyed by Yan et al. [2]. Traditionally, the most significant application of trust models has been in wireless ad-hoc sensor networks. However, it is increasingly common for IoT deployments to integrate sensor networks into their architectures [7]. Typically, where trust models fit into IoT security has related to applications with a single purpose, e.g., [5,9]. This work applies trust models more generally into heterogeneous networks, with the goal to quickly identify and prioritize threats, then prioritize the allocation of security controls towards devices posing a more significant threat based on a combination of capabilities and established trustworthiness. This is a departure from existing reputation-based trust systems that do not substantively incorporate the idea of capability [2].

This concept is explored through the development of a more comprehensive risk assessment framework that builds on existing trust modeling by accounting for immutable device capabilities. The trust model that serves as the foundation for risk assessment is proposed by Liu et al. [10]:

$$TL_i(j) = \frac{\sum_{k=0}^{n} TL_k(j)}{n} \tag{1}$$

where $TL_i(j)$ is the computed trust measure for some node j as assessed by node i, and n represents the number of nodes in the system providing trust reports about node j to node i. As an additional safeguard against malicious information, we can estimate the reliability of these reports:

$$TL_i(j) = \frac{\sum_{k=0}^{n} TL_k(j) \div TL_{req}(j)}{n} \tag{2}$$

where $TL_i(k)$ is the measure of trust that node j maintains regarding node k, $TL_k(j)$ represents the measure of trust regarding node j by node k as estimated through observation, and TL_{req} is the minimum acceptable threshold of trust needed for making access control decisions. The trust measure for each device is a scalar value. The system evaluates each device using its behavioral characteristics collected over time and determines a measure of trust.

This trust model was selected for its simplicity, its generality, and its extensibility for future applications. Most trust models are conceptually simple and straightforward to implement. The choice of this trust model is based on both of those characteristics as well as the extent to which it is representative of trust models more generally. Replacing this trust model with a similar behavior-based assessment that produces a scalar value indicative of the level of conformity with expected communication patterns would be feasible, but doing so would constitute an implementation detail specific to a local deployment.

4. Device Characterization

In addition to assessing trust, the system must identify immutable characteristics of each device to construct a device profile. Creating such a device profile requires fingerprinting mechanisms for each connected device. As described in Section 2.3, traditional fingerprinting is accomplished with active participation, in which each device must report information about its own hardware and software configuration and capabilities. Such methods have a critical weakness rendering them unsuitable for use in security mechanisms: their dependence on clients behaving appropriately and providing accurate information. Therefore, only passive fingerprinting techniques, which are resistant to forgery and spoofing, should be used. Passive fingerprinting approaches generally collect and analyze network traffic originating from a device under nominal conditions to detect immutable characteristics based on differences in implementation details unique to each device manufacturer.

The window size field of TCP packets, which is a 16-bit field starting at a 14-byte offset into the TCP header, is a primary source of fingerprinting data. This field is used in conjunction with the 8-bit time to live (TTL) field extracted from the IP header to identify operating system types, giving clues as to the type of device

sending the packets. The TTL field ordinarily is decremented with each hop as the packet is routed. In this systems, the TTL will not decrement since the devices connect directly to the gateway, with no additional hops involved by the time the packets are read by the trust daemon. The combination of TTL and window size fields is used to fingerprint devices using operating system vendor-specific values in their implementations for detection. Table 1 lists some commonly used and legacy operating systems and the window size/TTL pairs used for each [26].

A trust daemon running on the IoT gateway obtains each transmitted packet and extracts the TTL and window size fields from each. From this data, and using the table above, the daemon can fingerprint the devices that are communicating via the gateway. Note that this technique based on TTL/window size values is imperfect since multiple vendors may share the same values, e.g., Mac OS and FreeBSD. However, the purpose is to obtain a rough estimate of the type of device involved. This technique is sufficient for that purpose.

In this system, Bluetooth Low Energy (BLE) detection is not employed. One inherent characteristic of BLE is its intended application in low-powered devices using the 2.4 GHz band, just as WiFi does [27]. Because of this characteristic, devices using BLE to communicate are very likely to be low-powered and pose less risk to a network.

Table 1. TTL and Window Size Values for Common and Legacy Operating Systems.

Operating System	TTL	Window Size (bytes)
Linux 2.4 and 2.6	64	5,840
Android	64	5,720
Linux kernel 2.2	64	32,120
FreeBSD	64	65,535
OpenBSD, AIX	64	16,384
Windows XP	128	65,535
Windows 7, Vista, Server 8	128	8,192
Cisco Router IOS	255	4,128
Solaris 7	64	8,760
Mac OS	64	65,535

The IoT environment is dynamic and quite heterogeneous. Typically, an IoT deployment represents a diverse ecosystem of various protocols and technologies. Ultimately, however, application-specific communication protocols reside higher in the protocol stack than transport and network protocols. It is at the lower layers of the stack where fingerprinting occurs, primarily due to both the ease of forging and spoofing messages at the application layer as well as the relative challenge involved with doing so at lower layers in a manner that maintains interoperability with existing systems.

5. Risk Assessment

The trust daemon combines measured trust values with device capability information. Once a device is fingerprinted and a static profile established, the trust daemon monitors the device and its behavior. The daemon computes the device's trust value and refines it over time. Trust depends on the extent to which its observed behavior conforms to a statistical behavioral profile created a *priori*, with a value computed using Equations 1 and 2.

These trust values and device characteristic indicators drawn from static device profiles are then combined into a taxonomy of risk values much like a standard risk assessment matrix. Risk level is separated into three discrete categories: high; medium; and low risk. Delineations between different risk levels is a determination made by the system owner as is commonly accomplished as part of any risk management process. Trust levels are discrete scalar values ranging between 0 to 5, with a trust level of 0 assigned to a node known to be malicious and/or compromised. Values between 1 and 5 are assigned to devices that are either unknown (1) or

fully trusted (5). Similarly, device capability values span a range of values from 1, representing a minimally capable device, to 5, which is a device with a very high degree of capability.

Pairing these values in the risk assessment matrix results in a measure of risk, which is shown in Table 2. While the trust evaluation and fingerprinting result in a measure of the risk posed by a particular device, acceptance of risk is always a matter of policy determined by resource owners. Resource owners are responsible for the acceptance, avoidance, or mitigation of any security risks. Therefore, the risk assessment matrices such as that shown in the table may differ across systems based on policies set by individual system owners or administrators. Automated systems or dashboard applications generate alarms in accordance with such policy.

After determining the potential risk that each device poses, access to system services or communication may be restricted or denied accordingly. Similarly, additional, more detailed monitoring can be performed by a security analyst. As trust levels change, the associated risk level fluctuates accordingly.

Table 2. Risk Assessment Matrix.

Trust Level	Device Capability				
	1	2	3	4	5
0	HR	HR	HR	HR	HR
1	M/HR	M/HR	HR	HR	HR
2	L/MR	L/MR	MR	MR	MR
3	L/MR	L/MR	MR	L/MR	L/MR
4	LR	LR	L/MR	L/MR	L/MR
5	LR	LR	LR	LR	L/MR

6. System Design

Figure 1 illustrates the system architecture.

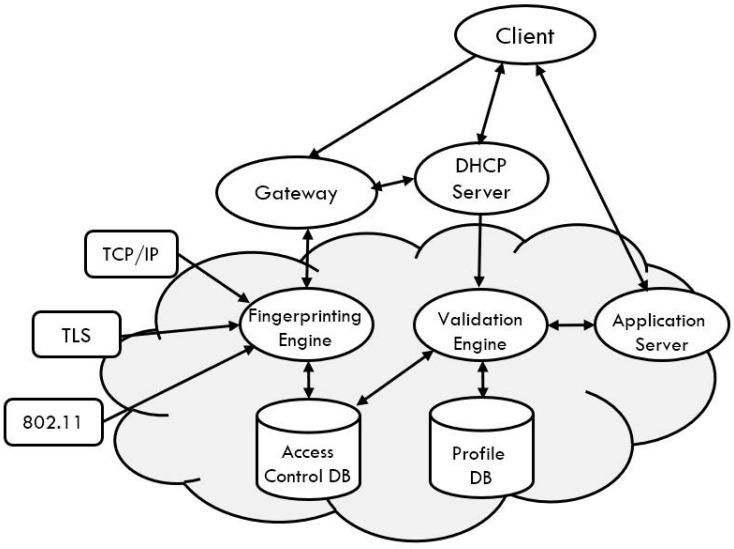

Figure 1. System Architecture.

In the architecture, an IoT gateway provides basic connectivity to the network for a collection of devices each with differing capabilities. When a device joins the network and is fingerprinted, the fingerprinting engine captures its capabilities in a device profile reflecting the immutable characteristics for its corresponding type. Device profiles are used in conjunction with the pre-established behavioral models that form the basis of trust

evaluation. Over time, network behavior is monitored, which leads to the construction of a behavioral profile based on the patterns of traffic the device initiates.

Periodically, the validation engine collects and evaluates packet transmissions based on frequency of occurrence and expected destination, with data compared to the previously established behavioral models for devices in a similar class. For instance, simple devices such as temperature sensors would be expected to all behave in a similar manner since they perform identical functions. Whenever a statistically significant deviation from the behavioral model occurs, a reduction in the device's trust level is effected. However, when behavior is flagged as malicious by an intrusion detection system or other network infrastructure device, the trust level is immediately reduced to zero. This action results in the device being quarantined or even evicted from the network entirely. Nodes which are known or suspected of being malicious or compromised may be readmitted to the network, but not without administrator intervention after appropriate remediation occurs.

For other non-compromised nodes with varying levels of associated assessed risk, network security policy may force the limitation of communications or the restriction of access to particular services. As trust is restored, security policy may permit additional network communications or access to additional services.

6.1 *Gateway*

The IoT gateway platform runs Ubuntu Linux 20.04 and serves as an access point by including a software-based wireless access point with a wireless network adapter in AP mode. Simultaneously, the access point includes a wired Ethernet interface that connects to other available network resources. Trust evaluation on the gateway is accomplished through a trust daemon process that sniffs network packets as they pass through the gateway. Protocols supported in the trust daemon include UDP, ICMP, ARP, with additional expansion possible with future modification. Protocol detection occurs based on the protocol field of the IP header in each packet examined.

No modification to the gateway's operating system is necessary, to include kernel modules. Employing a daemon process for profiling and trust evaluations allows for a standard off-the-shelf firewall to be deployed concurrently with no modification and no need to add, remove, or modify existing rules.

6.2 *Devices*

Devices connect to the gateway access point via wireless Ethernet and forward sensor data contained in TCP packets at periodic intervals to an internal aggregator machine. These packets pass through the gateway from each IoT device. As the packets pass through the access point, the trust daemon process evaluates traffic patterns based on statistical models of device behavior.

6.3 *Device Profiling*

The trust daemon has two primary functions: to use passive fingerprinting to identify devices as they connect to the network while creating static profiles; and to measure packet interarrival times to determine whether patterns conform to a statistical model of network behavior created for that type of device.

This model is designed to be extensible to IoT deployments implemented using a range of hardware and software platforms, application services, protocols, and data formats. Adapting this architecture to a new deployment environment would only require an update to the fingerprinting engine and validation engine based on the anticipated purpose of the deployment. This could be accomplished in an automated manner using machine learning techniques, but a detailed assessment of such an approach is outside the scope of this work.

6.4 *Trust Evaluation*

The system is deployed with pre-constructed statistical models of the network behavior of different types of IoT devices. These models are constructed from data describing packet interarrival times collected over time and represent typical device behavior. For instance, if a sensor device sends a packet with a new sensor reading at 1-minute intervals, then the packet interarrival time should be approximately 60 seconds, with variance expected for normal network conditions. A significant deviation from that average interarrival time could indicate malicious behavior or a device malfunctioning, in either case its trust value should be reduced.

To determine whether a device's behavior conforms to what is expected, a simple X^2 test is used. With a single degree of freedom, we use 3.84 as the critical value when determining whether behavior is anomalous.

The trust daemon reads packets using a raw socket. The ioctl() system call is then used to extract the operating system kernel's time stamp from each packet. At periodic intervals the validation engine computes and records a trust value for each device. If a device's behavior deviates from the model as determined by the

test, then the trust value for that device is decreased. Otherwise, the trust value is increased until it reaches a maximum value, at which point the device is fully trusted. The magnitude of the increase or decrease and the frequency of the trust evaluation is set via system policy.

7. Experimental Setup

System configuration includes a mechanism for detecting statistically anomalous behavior exhibited by connected devices based on expected traffic patterns. The trust daemon is pre-configured with a set of statistical models representing TCP packet interarrival times consistent with common sensor motes that constitute lesser capable IoT devices operating on lower power. For higher capability devices, a set of Raspberry Pi devices connect to the network via the wireless access point. A packet capture recorded the time stamps of incoming UDP packets and describes the construction of the models of packet interarrival times. These are subject to variations due to delays due to variations in incoming UDP packets that each device transmitted over wireless Ethernet through the gateway. Over a 30-minute duration, interarrival times among packets transmitted at 1-second intervals were recorded. Variations in measured packet arrival times is shown in Figure 2, while the CDF of the observed packet arrival times is shown in Figure 3.

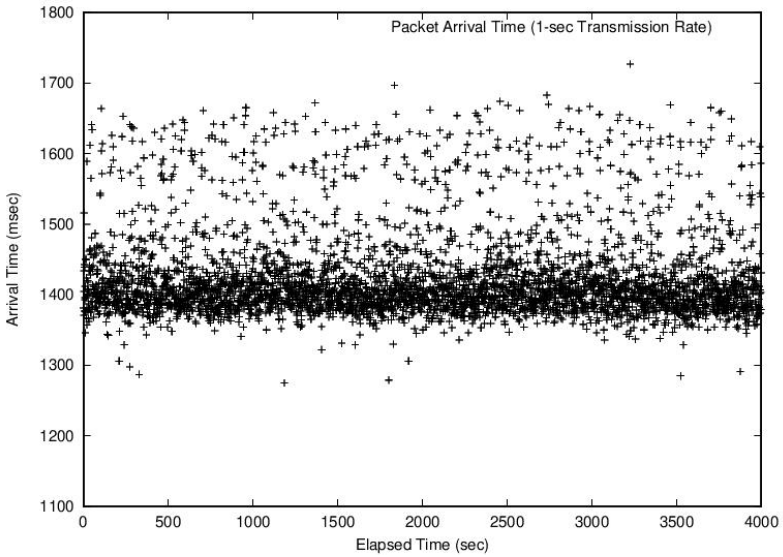

Figure 2. Raw Packet Interarrival Time Data.

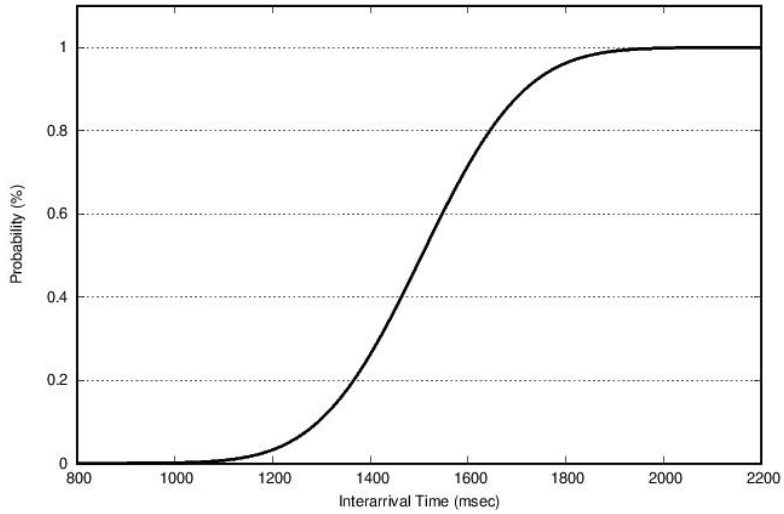

Figure 3. CDF of Packet Interarrival Times.

At 60-second intervals, the trust daemon samples packet data for each device and uses such data as input to trust level assessment. Device network behavior statistically consistent with the model for the device class

increases the device's trust measure. Trust levels are initially set to 0 for all devices and can reach a maximum value of 5.

At designated time intervals, the network traffic pattern is intentionally modified in a statistically significant way through the insertion of delays which introduce substantial deviations in packet interarrival times. Modifications includes additional unanticipated packets transmitted on different ports. This change in behavior, because it no longer conforms to the model and would be considered anomalous, causes the trust measure for the transmitting device to decrease with each additional assessment for the duration of the new behavior. For comparison purposes, devices configured to be less capable and lower powered are included with more capable and higher-powered devices. Both types of devices interact with the trust daemon as they transmit packets through the gateway.

8. Discussion

8.1 *System Operation*

With expected behavior, trust values increase over time. Correspondingly, the security risk that a device poses to the network decreases at a rate specified by security policy. Note that a low level of risk does not imply minimal defense or monitoring. A device might behave correctly for some time before executing a malicious payload such as a logic bomb. By assigning a higher risk level to a more capable device, such a device may still be sequestered from critical functions, data, or services or be subjected to a greater degree of monitoring.

8.2 *Fingerprinting*

Although variances in TCP window size are not uncommon, such variances can be substantial at times. During system operation, some rather significant variances in the TCP window size used for fingerprinting were observed depending on the nature of the traffic; this led to occasional mischaracterization of certain devices. Using a device known to have a Linux operating system installed, for instance, window sizes of 512 bytes were observed on occasion in addition to the expected value of 5,840. These specific values were also observed for Ubuntu Linux version 20.04, which is the operating system installed and running on the gateway machine. This type of challenges means only that additional calibration of the system, additional fingerprinting data collection, or an adaptive approach might be necessary to fully deploy it in an operational setting.

Because the time to live (TTL) field is a function of network distance between devices and is actually an initial TTL value, the more hops the packet takes as it is routed across the network the smaller its value becomes, i.e., the TTL value is decremented with each hop during routing. For fingerprinting purposes, it is necessary to estimate what the initial value most likely would have been. For instance, a TTL value greater than 64 would eliminate operating systems such as Linux or Android. For this application, the estimation of initial TTL value is straightforward. Because the host is serving as a gateway and wireless access point, for outbound TCP/UDP packets there is no decrementing of the value, and for incoming packets such decrementing is minimal. However, when designing more complex and larger scale systems, this is an implementation detail that should be considered.

Another behavior with respect to fingerprinting occurs with packet time stamping. Time stamp values do not always correspond exactly with the value expected, even under very controlled conditions. This observation was noted regardless of the mechanism used to obtain the packet time stamp: either via the system clock and the time() system call, or via the kernel's packet time stamp obtained via an ioctl() system call. For example, when testing packet interarrival times using a simple TCP client program that transmits a single-packet TCP message at periodic 1,000-millisecond intervals, the time stamps retrieved do not consistently reflect a packet arriving at that interval when compared to the previous packet time stamp from the same client, and oftentimes result in perceived interarrival times in the low tens of milliseconds. In practice that phenomenon did not appear to impact the function of the trust daemon since it was consistent across both the raw sockets and in the development of the statistical models. Whether that is true in a less controlled setting is undetermined.

8.3 *Trust and Risk Assessments*

The system is able to identify the devices and distinguish normal from anomalous behavior in cases where the devices acted as relatively simple sensor nodes. When operating a client application that forwards packets to an aggregator process at regular time intervals, normal patterns of network behavior were detected and the trust level of each device increased accordingly, thereby replicating the results related to trust levels in low powered

devices presented in [14]. With more complex and sophisticated devices and correspondingly more network traffic emanating from and to them, the trust daemon was able to detect anomalous behavior, but produced somewhat inconsistent results with both fingerprinting the device, which in this instance consists of a laptop machine with an Intel CPU and Windows 11 operating system.

When operating the system, anomaly detection is triggered by deviations in the average packet arrival time, which results in a corresponding increase in risk level associated with the device. Should the anomalous behavior continue, trust eventually decreases to a minimum value indicating a device requiring additional security controls. This is shown in Figure 4. The gateway trust daemon correctly detected such anomalous behavior. Similarly, as packets are transmitted such that the overall traffic pattern is statistically consistent with that captured in the models, the trust level of a device increases.

Trust measures are solely a function of behavior and the extent to which it conforms to a model of expected patterns independent of any other device attribute. The pattern that traditional trust levels exhibit is always the same for all categories of devices. Such a consistent pattern illustrates an inherent weakness of general trust models when used as a security mechanism: they do not provide a means of tailoring security policies to different situations. The only conclusion one can draw from a trust value is whether or not a device operates as expected. They do not necessarily provide readily perceptible indicators for security analysts. More significantly, a trust value alone can possibly conceal potential threats. Still, trust is an effective indicator of malicious or anomalous behavior, especially that exhibited by an unknown entity, and can be used as an input when assessing risk.

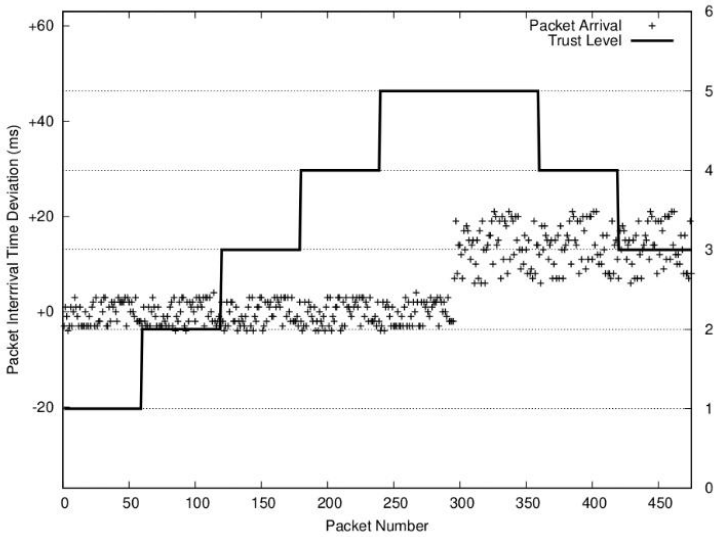

Figure 4. Trust Levels Over Time for Devices Exhibiting Expected, then Anomalous, Behavior.

The difference between trust and risk can be illustrated. To do so, we evaluate the result of the trust assessment on both a high-powered device and a low-powered device operating in a similar manner. Suppose both devices transmit packets in a manner consistent with their associated model over a designated time period, after which behavior becomes anomalous in nature. Such a change can occur abruptly or gradually with no difference in outcome. Once the behavior deviates from the model to a statistically significant extent, the behavior is flagged as anomalous. Figure 5 shows the risk level assessed for a high-powered device. The device begins operating normally. At a designated point, the behavior switches to anomalous with greater deviations in packet interarrival times. Figure 6 shows the corresponding change in risk for a low-powered device. In both cases, the risk levels demonstrate an inverse relationship to the trust levels in Figure 4. However, for a lower-powered, less capable device exhibiting a similar pattern of behavior and assessed with the trust model also used to assess the higher-powered device, the risks posed by each device are demonstrably different. A less-capable, lower-powered device poses a lower security threat than a more-capable, higher-powered device. Even with demonstrated anomalous behavior, it does not constitute the same risk to system security, provided the behavior is not known to be malicious. That difference is reflected in the risk assessments, but not in the trust model.

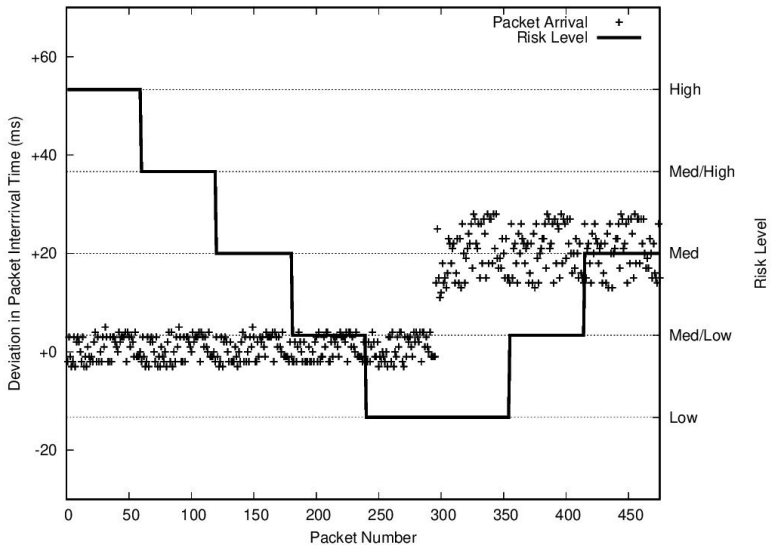

Figure 5. Risk Levels Over Time for a High-Powered Device.

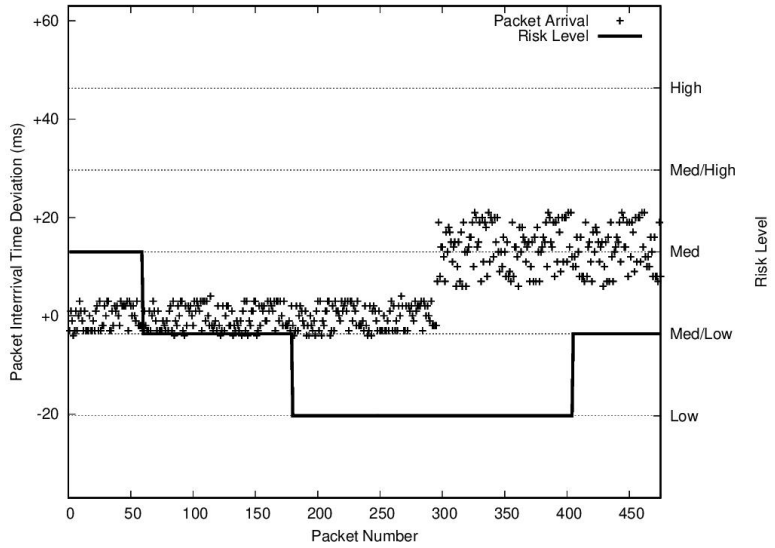

Figure 6. Risk Levels Over Time for a Low-Powered Device.

9. Conclusion

This paper presents an architecture that facilitates the combination of immutable device information and dynamic behaviors for devices of various types into an assessment of risk. The approach and architecture presented constitute a novel synthesis of existing security techniques that allows for either automated tools to focus security measures on devices that may pose a greater threat to system security while still maintaining a baseline security posture for others. Moreover, combining approaches into a cohesive whole reduces the likelihood that attackers can exploit the seams between components of the security architecture. The risk determination is a function of the capabilities of each device along with its conformity to expected behavioral patterns regardless of the specifics of the IoT deployment environment. Such an approach could have broad applicability to the application of security mechanisms and controls and could benefit security analysts by providing higher fidelity and more tailored information about communicating IoT devices.

Acknowledgments

This work was supported by the Air Force Office of Scientific Research (AFOSR) Summer Faculty Fellowship Program (SFFP).

Conflict of Interest

There is no conflict of interest for this study.

References

[1] Jia, Y.; Zhong, F.; Alrawais, A.; Gong, B.; Cheng, X. FlowGuard: An Intelligent Edge Defense Mechanism Against IoT DDoS Attacks. *IEEE Internet Things J.* **2020**, *7*, 9552–9562, https://doi.org/10.1109/jiot.2020.2993782.

[2] Yan, Z.; Zhang, P.; Vasilakos, A.V. A survey on trust management for Internet of Things. *J. Netw. Comput. Appl.* **2014**, *42*, 120–134, https://doi.org/10.1016/j.jnca.2014.01.014.

[3] Gu, T.; Mohapatra, P. BF-IoT: Securing the IoT Networks via Fingerprinting-Based Device Authentication. In Proceedings of 2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Chengdu, China, 9–12 October 2018, https://doi.org/10.1109/MASS.2018.00047.

[4] Gutscher, A.; Heesen, J.; Siemoneit, O. Possibilities and Limitations of Modeling Trust and Reputation. In Proceedings of the Fifth International Workshop on Philosophy and Informatics, Kaiserslautern, Germany, 1–2 April 2008. pp. 50–61.

[5] Han, G.; Jiang, J.; Shu, L.; Guizani, M. An Attack-Resistant Trust Model Based on Multidimensional Trust Metrics in Underwater Acoustic Sensor Network. *IEEE Trans. Mob. Comput.* **2015**, *14*, 2447–2459, https://doi.org/10.1109/tmc.2015.2402120.

[6] Hellaoui, H.; Bouabdallah, A.; Koudil, M. TAS-IoT: Trust-Based Adaptive Security in the IoT. In Proceedings of 2016 IEEE 41st Conference on Local Computer Networks (LCN), Dubai, United Arab Emirates, 7–10 November 2016, https://doi.org/10.1109/LCN.2016.101.

[7] Hudson, F.D.; Laplante, P.A.; Amaba, B. Enabling Trust and Security: TIPPSS for IoT. *IT Prof.* **2018**, *20*, 15–18, https://doi.org/10.1109/mitp.2018.021921646.

[8] Kamvar, S.; Schlosser, M.; Garcia-Molina, H. The Eigen-Trust Algorithm for Reputation Management in p2p Networks. In Proceedings of the 12th International Conference on the World Wide Web (WWW'03), New York, NY, USA, 20–24 May 2003, https://doi.org/10.1145/775152.775242.

[9] Kerrache, C.A.; Calafate, C.T.; Cano, J.-C.; Lagraa, N.; Manzoni, P. Trust Management for Vehicular Networks: An Adversary-Oriented Overview. *IEEE Access* **2016**, *4*, 9293–9307, https://doi.org/10.1109/access.2016.2645452.

[10] Liu, Z.; Joy, A.; Thompson, R. A Dynamic Trust Model for Mobile Ad Hoc Networks. Proceedings of 2004 10th IEEE Workshop on Future Trends of Distributed Computer Systems, Suzhou, China, 28 May 2004, https://doi.org/10.1109/FTDCS.2004.1316597.

[11] Nitti, M.; Girau, R.; Atzori, L.; Iera, A.; Morabito, G. A Subjective Model for Trustworthiness Evaluation in the Social Internet of Things. In Proceedings of 2012 23rd IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), Sydney, NSW, Australia, 9–12 September 2012, https://doi.org/10.1109/PIMRC.2012.6362662.

[12] Sicari, S.; Rizzardi, A.; Grieco, L.A.; Coen-Porisini, A. Security, privacy and trust in Internet of Things: The road ahead. *Comput. Netw.* **2015**, *76*, 146–164, https://doi.org/10.1016/j.comnet.2014.11.008.

[13] Trivedi, A.; Kapoor, R.; Arora, R.; Sanyal, S.; Sanyal, S. RISM - Reputation Based Intrusion Detection System for Mobile Ad-Hoc Network. *arXiv*, arXiv:1307.7833, https://doi.org/10.48550/arXiv.1307.7833.

[14] Abu Bakar, K.A.; Daud, N.I.; Hasan, M.S. Adaptive Authentication: A Case Study for Unified Authentication Platform. *Comput. Sci. Inf. Syst.* **2015**, 61–72, https://doi.org/10.5121/csit.2015.51006.

[15] Mahmoud, R.; Yousuf, T.; Aloul, F.; Zualkernan, I. Internet of things (IoT) Security: Current Status, Challenges and Prospective Measures. In Proceedings of 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 14–16 December 2015, https://doi.org/10.1109/ICITST.2015.7412116.

[16] Xiong, L.; Liu, L. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 843–857, https://doi.org/10.1109/tkde.2004.1318566.

[17] Litzkow, M.; Livny, M.; Mutka, M. Condor - A Hunter of Idle Workstations. In Proceedings of the 8th International Conference on Distributed Computing Systems, San Jose, CA, USA, 13–17 June 1988, https://doi.org/10.1109/DCS.1988.12507.

[18] Bratus, S.; Cornelius, C.; Kotz, D.; Peebles, D. Active Behavioral Fingerprinting of Wireless Devices. In Proceedings of the 1st ACM Conference on Wireless Network Security, Alexandria, VA, USA, 31 March–2 April 2008, https://doi.org/10.1145/1352533.1352543.
[19] Cao, Y.; Li, S.; Wijmans, E. (Cross-)Browser Fingerprinting via OS and Hardware Level Features. In Proceedings of the 2017 Network and Distributed Systems Security Symposium, San Diego, CA, USA, 26 February–1 March 2017.
[20] Lee, Y-K.; Lee, D.; Han, J-W.; Kim, T-H. Home Network Device Authentication: Device Authentication Fr amework and Device Certificate Profile. *Comput. J.* **2009**, *52*, 871–877, https://doi.org/10.1093/comjnl/bxn 038.
[21] Ibrahim, M.F.; Yahya, S.; Taib, M.N. Device Characteristics and Capabilities Discovery for Multimedia Content. *J. Comput. Networks Commun.* **2012**, *2012*, 1–15, https://doi.org/10.1155/2012/935653.
[22] Neumann, C.; Heen, O.; Onno, S. An Empirical Study of Passive 802.11 Device Fingerprinting. In Proceedings of 2012 32nd International Conference on Distributed Computer Systems Workshops, Macau, China, 18–21 June 2012, https://doi.org/10.1109/ICDCSW.2012.8.
[23] Anderson, B.; McGrew, D. OS fingerprinting: New techniques and a study of information gain and obfuscation. In proceedings of 2017 IEEE Conference on Communications and Network Security (CNS), Las Vegas, NV, USA, 9–11 October 2017, https://doi.org/10.1109/cns.2017.8228647.
[24] Zhang, Y.; Lin, Z. When Good Becomes Evil: Tracking Bluetooth Low Energy Devices via Allowlist-Base d Side Channel and its Countermeasure. In Proceedings of 2022 ACM SIGSAC Conference on Computer a nd Communication Security, Los Angeles, CA, USA, 7–11 November 2022, https://doi.org/10.1145/35486 06.3559372.
[25] Bao, F.; Chen, I. Dynamic Trust Management for Internet of Things Applications. In Proceedings of the 2012 International Workshop on Self-Aware Internet of Things (Self-IoT ’12), San Jose, CA, USA, 17 September 2012, https://doi.org/10.1145/2378023.2378025.
[26] Tyagi, R.; Paul, T.; Manoj, B.; Thanudas, B. Packet Inspection for Unauthorized OS Detection in Enterprises. *IEEE Secur. Priv.* **2015**, *13*, 60–65, https://doi.org/10.1109/msp.2015.86.
[27] Stute, M.; Heinrich, A.; Lorenz, J.; Hollick, M. Disrupting Continuity of Apple’s Wireless Ecosystem Security: New Tracking, DoS, and MitM Attacks on iOS and MacOS Through Bluetooth Low Energy, AWDL, and Wi-Fi. In Proceedings of the 30th USENIX Security Symposium, virtual event, 11–13 August 2021.