Programmable Active Scans Controlled by Passive Traffic Inference for IoT Asset Characterization

Hugo Sullivan, Arunan Sivanathan, Ayyoob Hamza, and Hassan Habibi Gharakheili
School of Electrical Engineering and Telecommunications, UNSW Sydney, Australia.
Emails: {*hugo.sullivan@student., a.sivanathan@, m.ahamedhamza@, h.habibi@*}*unsw.edu.au*

***Abstract*—The proliferation of Internet-of-things (IoT) assets has expanded the attack surface of enterprise networks exposed to malicious actors. Therefore, obtaining visibility into connected assets and their behavioral characteristics is increasingly becoming essential to security teams in better managing their network and connected assets. Scheduled vulnerability scans are widely used by enterprises to manage traditional information technology (IT) assets. However, resource-constraint IoT assets may not always withstand disruptive active scans. Passive traffic inference tools have recently emerged for continuous network detection and response capabilities that can be safely applied to IoT and IT networks. Both active and passive approaches come with advantages and limitations in the insights they provide versus measurement and computing costs. This paper attempts to systematically and dynamically leverage the combined capabilities offered by these two approaches. Our contributions are twofold. (1) We highlight capabilities (richness of insights, response time, and temporal utility) and quantify costs (overhead traffic and computing resources) across five active scanning tools (open-source and commercial) and a commercial passive inference tool by applying them to our testbed consisting of 12 commercial IoT devices; and, (2) We develop "*pScan*", a programmable packet emitter with open APIs that is dynamically controlled to perform contextualized scans on target IoT assets via SNMP, mDNS, and SSDP packets, as well as banner grabbing and custom probing via TCP connections. We show on our testbed how pScan integrated with the commercial passive inference tool helps to maximize the insights into the characteristics of IoT assets and their utility at significantly reduced costs. We contribute pScan as open source.**

***Index Terms*—IoT asset characterization, active scans, passive inference**

I. Introduction

Enterprises and businesses are increasingly becoming digitized and hyper-connected by adopting diverse classes of IoT, OT (Operational Technology), or cyber-physical assets (*e.g.,* cameras, programmable logic controllers, building sensors, air quality monitors, printers, or screens). This trend has led to an expanded attack surface and significant growth in the frequency and impact of cyber attacks [1]. Therefore, enterprise security teams find that it has never been a more important time to know what assets are present on the network (asset composition visibility) and what they do (asset behavior monitoring), helping them manage the fast-changing attack surface [2].

Traditionally, enterprise operators relied upon network scanning tools to obtain "point-in-time" (scheduled) visibility into IT assets (*i.e.,* physical or virtual machines). These tools actively and often aggressively generate tailored request packets (*e.g.,* TCP/UDP, ICMP, SNMP) towards the networked assets, interrogating them to produce specific responses revealing installed OS/software, open ports, make/model, security configuration settings, or known malware. However, relatively broad and uncontrolled active scans can risk the operation of sensitive IoT assets by overwhelming their limited compute resources (due to excessive traffic and workload) and hence degrading and/or disrupting their intended functions. As a result, network administrators often isolate the network segments of IoT assets from active scanners, which results in limited visibility. Additionally, if compromised or instrumented by agents like IpMorph [3], target assets can obfuscate their identity against known aggressive scan queries (*e.g.,* TCP SYN packets from tools like Nmap, Xprobe2, and SinFP).

More recently, non-invasive tools [4] have emerged that passively and continuously analyze network traffic to classify connected assets [5], characterize their behavior [6], determine cyber-risks in their profiles [7], detect anomalous patterns [8], and infer their software components [9]. Fortunately, passive techniques are equally applicable to IoT and IT assets, potentially obtaining fresher and richer visibility into connected assets and their attack surface. With pure passive inference, however, it may take a while before certain behavior (possibly infrequent) or explicit knowledge pertinent to networked assets is observed. In this paper, we aim to leverage the combined capabilities of passive and active approaches in a controlled ("selective") and cost-effective manner.

We make two specific contributions. Our **first** contribution (§III) systematically experiments with five (four open-source plus a commercial) active scanning tools and a commercial passive inference tool by applying them to our lab testbed consisting of 12 commercial IoT devices. We highlight the individual capabilities of these tools and quantify their costs in terms of temporal utility, obtained insights, overhead traffic, and computing resources. Our **second** contribution develops a tool called *pScan* to emit packets on the network programmatically and dynamically via API calls—we open-source our tool. We build a system by integrating pScan with the commercial passive inference tool (we used for benchmarking) to perform contextualized scans on target IoT assets. We demonstrate how controlled scans help operators of IoT networks maximize insights into characteristics of certain IoT assets while costs are kept at a minimum.

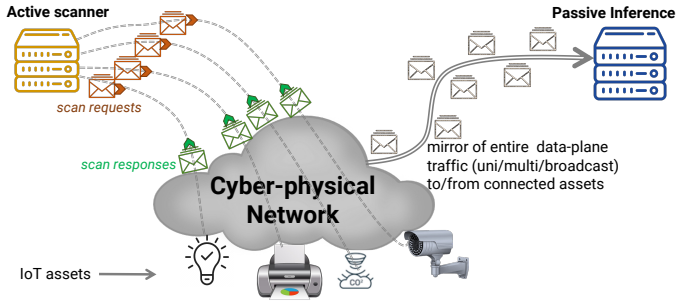

Fig. 1. Active versus passive characterization methods.

II. Related Work

Asset Characterization: Mapping and characterizing network assets have been of interest to academia and industry for more than two decades. Both active [10], [11], [12] and passive [13], [14], [15], [12] monitoring methods have been employed to profile the behavior of various IT (servers and/or end-hosts) and IoT networked assets. In the context of IT asset management, OS fingerprinting [16] is the top use-case whereby active scans are primarily employed to assist network administrators in detecting unauthorized, outdated, or vulnerable versions of operating systems on their network.

Work in [3] highlights the practical challenges of asset fingerprinting and demonstrates how known tools can be actively disturbed (misled) by target assets. Similar to the popular search engine "`shodan.io`", work in [17] builds upon an open-source active scanner (ZMap) to conduct Internet-wide scans, and grabs protocol banners (*e.g.,* HTTP, FTP, SSH) to identify vulnerable assets and networks. In the context of home networks, work in [10] utilizes user-initiated network scans to draw insights into the adoption of consumer IoT devices and their security posture across the globe.

Recent research extensively focused on IoT assets and employed network packets and/or flows to passively infer their identity (clusters [18], make/models [19]) or behavioral characteristics [20], [21]. Authors of [11] attempted to discover known IoT assets on the network by way of custom TCP scans.

IoT Attack Surface Management: Large-scale cybersecurity incidents like Mirai botnet [22] have given rise to research on quantifying [23], [7] and managing [24], [25] cyber risks of IoT assets in residential, enterprise, and critical infrastructure settings. Policymakers, regulatory bodies, and standard agencies have attempted to propose approaches, such as Manufacturer Usage Description (MUD) [26] or Software Bill of Material (SBoM) [9], to enhance the transparency of IoT characteristics, leading to systematically managing their vulnerabilities and attack surfaces at network/device levels. Authors of [23] actively subject IoT devices to reflective attacks quantifying their amplification factors. Work in [7] passively analyze HTTP traffic exchanged by IoT devices to assess their authentication vulnerabilities. Work in [25] proactively enforces intended network flows (supplied by known MUD files) of each connected IoT device to lock their network behaviors and isolate/inspect unintended communications.

Our paper is inspired by prior work and focuses on a new method for characterizing IoT assets, developing a base knowledge from passive inference, and expanding it by contextualized and controlled active scans. This work helps operators of large IoT/OT/cyber-physical environments map connected assets on their network and determine their fine-grained characteristics (make/model, role, firmware, network behaviors). This systematically enables them to perform subsequent management tasks (*e.g.,* appropriate segmentation, firmware upgrade, access rules configuration, assessing security posture).

III. Quantifying Efficacy of Active Scanners versus Passive Inferencing

There exist a number of open-source tools and commercial products for active network scanning and passive traffic inferencing. Fig. 1 illustrates how these two approaches fundamentally differ. Active scanners (shown on the top left) send custom requests (orange packets) to assets on target subnets and analyze scan responses (green packets), if any, from one or more assets connected to the network. On the other hand, the passive inference engine (shown on the top right of Fig. 1) continuously ingests the entire network traffic (unicast/multicast/broadcast packets) sent and received by individual connected assets. In what follows, we will experiment with a collection of these tools/products on our IoT testbed, measure performance metrics, and highlight their capabilities versus costs.

A. Experimental Testbed

Our testbed comprises a TP-Link Archer C7, serving as a local switch and the Internet gateway for 12 unique consumer IoT devices, including cameras ($\times$3), switches and triggers ($\times$3), printer ($\times$1), lightbulb ($\times$1), and entertainment electronics ($\times$4) devices. The testbed also serves two host machines (a Linux and a Windows) running scan tools and feeds a second Linux machine running a passive inference tool. Connected devices and machines share an IPv4 subnet of `192.168.1/24`. The gateway is flashed with OpenWrt firmware (and additional packages) for mirroring the network traffic to the second Linux machine. The first Linux machine hosting open-source active scan tools has an Intel i5 with four cores at 3.20 GHz and 8 GB of memory; the Windows machine for the commercial active scanner has an Intel i5 with four cores at 3.20 GHz and 8GB of memory; and, the second Linux machine running the passive software has an Intel Xeon(R) Gold with eight cores at 2.20GHz, and 16 GB of memory.

B. Tools

For the research of this paper, we experiment with four open-source scanners, a demo version of a commercial scanner ("*Comm. Scanner*"), and a demo version of a commercial passive inference tool ("*Comm. Passive*") with well-documented inbound and outbound APIs.

Nmap is a powerful, well-documented, and popular active OS fingerprinting tool. It is well supported with an extensive operating system database with corresponding signatures, an excess of features, and a graphical interface (Zenmap). Its OS

TABLE I
EVALUATING PERFORMANCE OF ACTIVE SCANNING AND PASSIVE INFERENCE TOOLS.

Tools		Emitted Packet Types	Insights					Traffic Overhead (# packets / asset)				Computing		Response time (s)	Average Utility
			OS	Mnfctr	Services Open/Exposed	Services Accessed	Hostname	Basic Reconnais.	OS	Services Exposed	Host name	CPU (%)	RAM (KB)		
Active	**Nmap**	ARP, HTTP, ICMP, TCP, UDP	✓(5)	✓(9)	✓(6)	✗	✓(8)	378.58	89.92	17.92	0	1.15%	46,220	191.20	79.5%
	XProbe2	ICMP, TCP	✓(1)	✗	✗	✗	✗	61.83	11.58	✗	✗	26.33%	8,040	2791.88	79.5%
	Masscan	TCP	✗	✗	✓(6)	✗	✗	2135.75	✗	6.75	✗	0.92%	44,220	38.08	79.5%
	Angry IP	ARP, HTTP, NBNS, TCP, UDP, MDNS	✗	✓(8)	✓(2)	✗	✓(9)	123.00	✗	6.42	0.50	0%	600	12.42	79.5%
	Comm. Scanner	TCP, SSDP, SNMP, SIP, MDNS, LLMNR, NBNS, ICMP, ARP	✓(1)	✓(12)	✓(2)	✗	✓(9)	422.75				2.10%	166,800	116.81	79.5%
Comm. Passive		–	✓(5)	✓(12)	✓(12)	✓(12)	✓(8)	0				11.96%	7,930,000	0	100%

fingerprinting tool is relatively comprehensive, using thirteen TCP probes[1], one UDP probe, and two ICMP probes. Nmap [27] also offers port scanning and banner grabbing with additional probes to detect the version of services running on open TCP/UDP ports. By default, Nmap begins by banner grabbing, using 3000 "NULL probe signatures" to recognize different types of welcome banners. If services are not fully detected, it will send additional probes progressively until the service and version are determined. These probes are dynamically chosen based on what is inferred from the welcome banner and the rarity value associated with each built-in probe.

Xprobe2 was introduced to overcome practical challenges (*e.g.,* the existence of network filtering devices, load balancers, and scrubbers, or modifications made on target machines) for active OS fingerprinting. This tool primarily relies on the ICMP-based probes, sending four packets, including two echo requests, a Timestamp request, and an Address Mask request. It may also emit a few TCP SYN followed by UDP packets masquerading as DNS queries. In contrast to the TCP-based OS detection tests, Xprobe2 uses fewer scan packets. Still, it applies its sophisticated "fuzzy" algorithm, a matrix-based fingerprinting matching approach based on OCR recognition, inferring the OS. This matching algorithm is more intensive computationally than Nmap's strict matching [28].

Masscan is a tool optimized to perform fast scans (similar to ZMap/ZGrab) at the Internet scale to discover vulnerable connected hosts. Masscan is claimed [29] to be able to scan the entire Internet in under 3 min by sending 25 M packets per second. In addition to standard TCP SYN port scan on the entire subnet, this tool relies on banner grabbing performed on well-known TCP protocols like FTP, HTTP, IMAP4, Memcached, POP3, SMTP, SSH, SSL, SMBv1/v2, Telnet, RDP, and VNC.

Angry IP Scanner is a lightweight [30] network management tool with a graphical user interface that discovers network devices and host-names primarily based on NetBIOS, mDNS, and DNS protocols. Additionally, this tool employs two HTTP probes if TCP/80 is found open, determining whether a web service is exposed by the target asset.

Comm. Scanner focuses more on identifying the make/model of connected devices rather than simply identifying their OS and comes with a comprehensive GUI. By manually analyzing the network traffic generated by this commercial tool while performing a scan, we found that it employs SNMP (with specific OIDs[2]) and more common probes like TCP, UDP, and ICMP to gather information about its target asset.

Comm. Passive aims at automatically generating and profiling the network behavior (in a form compatible with the MUD standard) for individual networked assets by ingesting and processing their network traffic in real time. It also comes with engines to opportunistically infer asset make/model and OS from typical packets (*e.g.,* DHCP, ICMP, SNMP, HTTP, SSDP, mDNS) assets that may exchange on the network for their normal operation. Additionally, this tool highlights cyber risks and detects anomalies (deviations from a baseline) in asset network behaviors on its web-based dashboard.

C. Performance Metrics

We quantify the performance of these tools by five specific metrics: (i) as briefly mentioned above, each tool comes with specific objectives. Therefore, they employ certain techniques/strategies to draw various **insights** into connected devices and their characteristics. We split insights into five pillars: OS, device manufacturer, network services open/exposed, network services accessed, and host names; (ii) for each category of insights (wherever applicable), we measure **traffic overhead** by the average number of packets emitted per asset on the network; (iii) the tools consume **computing** resources to ingest received packets. For passive scanning tools, we measure the average CPU/RAM utilization during the course of one scan round for the entire testbed. For the passive tool, we measure the average computing resources over 30 minutes; (iv) for active tools, expected insights can be obtained only when networked assets respond to probing packets. We measure the total duration of each scan round as a proxy for **response time**. For the passive tool, we consider this measure zero as it continuously analyzes traffic and generates/updates insights; and, (v) the freshness of insights diminishes in time and can be modeled by a **utility** metric. In other words, utility is a non-increasing function of the age of information (AoI) [31] that we estimate by:

$$U(t) = \frac{2}{e^{\alpha t} + e^{-\alpha t}} \tag{1}$$

where the utility $U(t)$ realizes its maximum value (100%) at $t = 0$ when insights are first obtained, and the value tends to

[1]A probe is a packet with custom-crafted headers and/or content designed to elicit specific responses.

[2]An Object Identifier is a string of numbers that identify a device within the Management Information Base (MIB) hierarchy and the status of its specific variables.

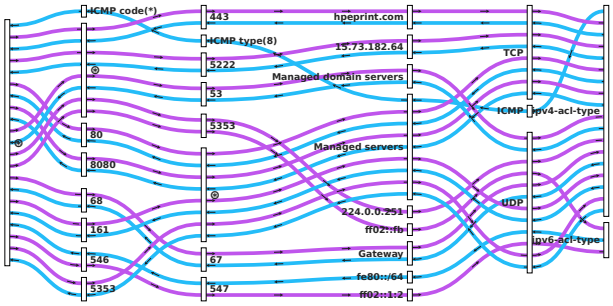

Fig. 2. Full behavioral profile of the HP printer obtained by the passive inference tool.

zero at infinity. Parameter α defines how quickly this function decays. The rate of decay can be perceived differently across IoT environments and/or network operators. For the sake of quantitative measures, let us assume it drops to half after a unit of time (say, a day), meaning $\alpha = \cosh^{-1}(2)$.

D. Evaluation Results

We now apply individual tools to our experimental testbed. Note that Nmap and Masscan require an intended list of scanned ports to be specified as arguments. Nmap allows scanning of the most common ports with two separate built-in lists for TCP and UDP, but by default, it goes with TCP only. Note that these two active tools can also perform full scans, covering more than 65000 possible ports for each TCP and UDP. However, full scans can be significantly expensive (traffic overheads) and long (response times). To balance insights against costs, we choose the top 100 ports from the Nmap built-in TCP list. For consistency, we feed the same list to Masscan. We employ Xprobe2, Angry IP, and Comm Scanner with their default/basic configurations. Table I summarizes the five categories of performance metrics measured for individual tools. It is important to note that the tools we study in this paper differ in their capabilities across performance metrics. Missing or not-applicable metrics are indicated by "✗" marks.

Starting from the insights column, cells with a "✓" highlight the number of IoT assets (out of a total of 12) for which the corresponding characteristic can be obtained. For example, Nmap determines the OS for five IoT devices connected to our testbed, while Masscan cannot provide this specific insight. It can be seen that Masscan only obtains information about the services open/exposed by six networked IoT assets. Standard port scans can determine open services. For the services exposed, active tools may use two methods: banner grabbing, and service and version detection. Banner grabbing is the process of establishing a TCP connection with a target asset and waiting (typically 5 sec) for the asset to respond with a packet called a welcome banner. Service and Version detection is a term used by Nmap and is a process by which the scanner sends a series of probes on a specific port being investigated, attempting to verify a service (*e.g.,* HTTP) and/or determine the service version running on that port. For example, for an HTTP server (*e.g.,* open ports TCP/80, TCP/8080, or TCP/8000), Nmap would start checking `HTTP/1.0` followed by `HTTP/1.1` with corresponding `GET` requests, determining whether a specific version of HTTP is present or not.

A key observation is that the passive tool uniquely differentiates itself from all active scanners in determining services (ports) accessed for all 12 IoT assets. Note that active tools are inherently incapable of obtaining this specific insight. Also, it can be seen that the passive tool maps exposed services for all IoT assets on our testbed at no cost – the next best tools, by this metric, are Nmap and Masscan, which detect services open/exposed by half of the IoT assets. We found that the passive tool detected a total of 166 exposed services across 12 IoT devices. The five active scanners together, on the other hand, detected only 16 services exposed by these IoT devices. The reason for such a significant gap could be twofold: (a) some assets with a certain service exposed, for some reason (*e.g.,* security/privacy), may choose not to respond to scanners' standard port scans and/or synthetic probes of banner grabbing or service&version detection, and (b) scanners often (like in our experiments) look for a subset of popular services, missing less-popular or non-standard ones. Passively observing networked assets, naturally communicating with their intended endpoints, on the other hand, would probably increase the chance of detecting services (ports) they expose on the network. Fig. 2 illustrates the behavioral profile (Sankey diagram obtained by the passive tool) for a representative device, the HP printer, connected to our testbed. Darker lines highlight outgoing communications from the asset (services accessed), and lighter lines highlight incoming communications to the asset (services exposed). In these Sankey diagrams, columns from the left-most to the right-most are the asset, asset port number or ICMP code, the remote port number or ICMP type, remote endpoint (*e.g.,* "`hpeprint.com`" or "`244.0.0.251`"), transport-layer or network-layer protocol (*e.g.,* TCP, UDP, ICMP), and IP protocol (IPv4, IPv6), respectively.

Moving to traffic overhead, we observe that Massscan comes first, with an average of more than 1300 packets emitted per IoT asset, followed by Nmap and Comm Scanner, each with more than 400 packets per asset (on average). It is important to note that such overheads strongly correlate with the number of insights these three tools provide. It can be seen that scanning tools tend to spend most of their emitted packets on basic reconnaissance (presence on the network, open ports, and/or manufacturers) with a combination of ARP and/or ICMP Echo packets, plus (in some tools) simple TCP port scans. The MAC address in ARP responses can help determine asset manufacturers. We found that Massscan (primarily designed for use across the Internet) performs "blind" TCP scans (of 100 popular ports) on the entire /24 subnet in our testbed, though only 14 IPv4 addresses (twelve IoT assets plus two machines) are active on the network, hence resulting in significant overheads. On the other hand, Nmap and Comm Scanner (designed for more controlled/closed networks) launch TCP port scan only on those IPv4 addresses discovered by ARP queries. For Comm Scanner, we could not precisely determine what packets were used to obtain what pieces of information about the target device; hence an aggregate measure of traffic overhead is reported accordingly in Table I. Also, it can be seen that XProbe2 and Angry IP, as

expected, impose comparatively less overhead on the network due to their narrow focus on specific protocols (*e.g.,* ICMP, mDNS) and not using wide-range scans. Finally, Nmap relies on DNS queries to obtain asset host names – probably cached on the Linux machine before our experiments, hence zero packets. However, tools like Comm Scanner and Angry IP use mDNS on top of DNS queries, thus non-zero packets. We note that active tools lack context, resulting in unnecessary costs. For example, they often need to discover open/available ports/services by way of relatively blind and wide scans before they start collecting meaningful insights. We will see in §IV how scans can be contextualized by foundational knowledge obtained from passive inference at no traffic cost.

For computing costs, we employed the `htop` application on Linux machines to measure and record CPU/RAM utilization every three seconds. On the Windows machine, we used the Resource Monitor application to measure these two metrics. For comparison, all measurements of CPU utilization are normalized to 4 cores operating at 3.2 GHz. We see in Table I that the passive tool, unsurprisingly, consumes a substantial amount of resources (≈11% CPU and ≈8GB RAM). Recall this tool continuously analyzes and makes inferences from the entire network traffic, as opposed to certain response packets analyzed by active scanners – hence, relatively high RAM utilization is expected. XProbe2 demands a high amount (26.33%) of CPU among the five active tools due to its intense matching algorithm, while others are relatively light. Comm Scanner costumes the highest amount (>160 MB) of RAM among the five scanners. Thanks to a few scanned packets judiciously and efficiently chosen, Angry IP outperforms the scanner rivals by its tiny usage of both CPU and RAM.

Looking at response time in Table I, Masscan performs impressively fast (≈38 sec), given its highest traffic overhead. This is because Masscan operates asynchronously (as opposed to other scanners) with two independent threads (one for sending queries and one for receiving responses), meaning it does not have to wait for replies before sending more port scan packets. Xprobe2, surprisingly (given its lowest overhead), performs the worst among the five active tools with a response time of ≈2800 sec. This is mainly attributed to its complex matching method coupled with the single threading, severely slowing down its operation. The remaining tools give response times comparable to their way of emitting packets.

Finally, in terms of utility, all active scanners are equally suboptimal. To address this issue, network operators often schedule periodic scans. We model (not measure) this metric with the parameter α in Eq. 1. Let us assume scans are repeated every day to make active scanners, to some extent, comparable with passive inference. Note that scans are typically scheduled quarterly or monthly. Even with an unusual frequency (daily), we see the average utility of active scanners is less than 80% versus 100% of the passive approach. Though the visibility of the passive approach is always fresh, it is incomplete at the beginning of traffic observation. It progressively gets enriched over time as more behavioral patterns and characteristics emerge in the network traffic.

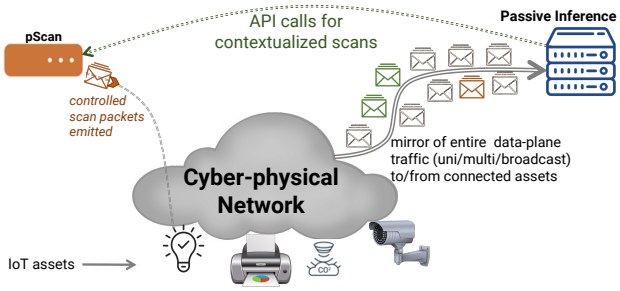

Fig. 3. Our proposed solution: passive inference combined with pScan, a programmable packet emitter.

IV. Our Combined Passive-Active Method

We saw in the previous section that passive inference is inherently more capable than active scanning to obtain rich and real-time visibility into assets and their characteristics at no risk of downtime or disruption (zero overhead traffic) for sensitive IoT assets. Particularly, active scanners are blind to services accessed by networked assets. Also, exposed UDP services in our experiments remained undiscovered by active scanners since only the top 100 TCP ports were targeted (primarily because we aimed to reduce overhead traffic). It is, however, important to note that active scanners can quickly characterize well-known (but rarely-used) services IoT assets expose. In other words, active scans can help accelerate obtaining (close to) complete visibility without waiting for infrequent/rare patterns to emerge on the network. In this section, we aim to leverage the capabilities of active and passive approaches, realizing efficient and comprehensive visibility into IoT asset characteristics. In what follows, we outline the technical details of our solution and evaluate its efficacy.

System Architecture: Fig. 3 illustrates the architecture of our proposed solution. The key component of the solution is a programmable scanner, shown as an orange box on the top left of Fig. 3, that emits specific packets dynamically on demand. We developed and open-sourced a software tool[3] called "*pScan*" for this programmable scanner. It can be seen that API calls (dashed green arrow from right to left in Fig. 3) from other tools drive our packet emitter. The tool unifies existing libraries and exposes a simple API to other tools to utilize. In this paper, the passive inference engine (the blue box on the top right of Fig. 3) is the caller, making dynamic calls whenever additional information is needed for one or more IoT assets. It is important to note that the emitter module solely sends scan requests (to specified assets) and does not process responses to those request packets. Needless to say, the passive inference tool will ingest scan requests and corresponding responses (orange and green packets carried by the mirror link in Fig. 3) in conjunction with other packets to/from all networked assets.

APIs: Our packet emitter exposes APIs, allowing for dynamic (on-demand) and contextualized scans. Each `POST` API call would specify target devices (*e.g.,* a device unit, multiple units, or a subnet), an intended type of scan packet

[3]https://github.com/hugo-sullivan/pScan

(*e.g.,* SNMP, mDNS), the number and timing of packets, and parameters specific to each packet type (*e.g.,* OID for SNMP).

Packet Types: pScan is extensible and currently supports four packet types, namely "*TCP banner grabbing*", "*SNMP*", "*mDNS*", and "*SSDP*". Banner grabbing is an efficient way of detecting services running on the target device and actively verifying whether there is any change from the last measurement (some devices may not display event-based behaviors for days/weeks). The parameters of this call include intended TCP ports to be checked and a boolean variable if additional probes are required. For this specific functionality, pScan uses Nmap under the hood, supporting 650 protocols. pScan calls for SNMP packets require a boolean indicating `get` or `get-bulk`, a string for the OID (a tree structure), and two integers (start and iteration times) specifying a region of the OID tree for a get-bulk request, as parameters. The required parameters of mDNS calls is a query name. Subsequently, pScan emits a PTR query (aiming at discovering services and hostnames) to the default multicast address `224.0.0.251`. For SSDP an integer between 1 and 5 for the `MX` (max waiting time in seconds). pScan emits an M-Search query to the default multicast address `239.255.255.250`. The tool supports additional packet types being added in the future – pScan's public repository provides instructions on how it can be extended.

Benefits of Contextualized Scans: Based on the results from §III-C, we start using the passive tool as a base to characterize IoT assets of our testbed, assisted by the pScan contextualized scans. The passive tool discovered that SNMP is only supported by one of the connected devices, but no informative OID is set in those generic SNMP communications by that device. The passive tool thereby calls pScan to emit an SNMP packet to that device with OID set to `1.3.6.1.2.1.1.5.0` (SNMP MIB-2), leading the device to reveal its model "SNH-6410". It next requested a second SNMP packet with the OID equal to `1.3.6.1.2.1.1.1.0` obtaining a description "Samsung iPolis H.264 Premium Level IP Camera" from the device. The passive tool found a device port TCP/80 long inactive following initial discovery. Therefore, it makes a call to pScan to emit an `HTTP 1.1` probe to that specific device which responds with its model "HP ENVY 5540 series - G0V47A" and serial number. Finally, the passive tool discovered two devices manufactured by Google. A request for emission of a multicast mDNS packet querying for a Google Cast service was sent to pScan, resulting in a response from one of those devices revealing its model "Google Chromecast Ultra".

V. Conclusion

Characterizing connected assets is paramount to manufacturing and industrial operators to manage the attack surface of their IoT networks. This paper started by systematically assessing the efficacy of two approaches, active scanning versus passive inference, by their insights and the costs/risks they impose. We next developed pScan, a programmable packet emitter (open-sourced), integrated it with the passive tool, and demonstrated how contextualized scans enrich insights into IoT assets of our testbed in a controlled and cost-effective manner.

References

[1] S. Chhillar and D. Geach, "Digital Transformation and Emerging ICS/OT Cyber Attacks," in *Proc. ACM CPSIoTSec*, Nov 2021.

[2] H. Habibi Gharakheili *et al.*, "Cyber-Securing IoT Infrastructure by Modeling Network Traffic," in *Security and Privacy in the Internet of Things: Architectures, Techniques, and Applications*, A. Ismail Awad and J. Abawajy, Eds. John Wiley & Sons, 2021, ch. 6, pp. 151–176.

[3] G. Prigent *et al.*, "IpMorph: Fingerprinting Spoofing Unification," *Journal in Computer Virology*, vol. 6, no. 4, pp. 329–342, Nov 2010.

[4] Gartner, "Market Guide for Network Detection and Response," 2020. [Online]. Available: https://bit.ly/3BXL9Pl

[5] A. Sivanathan *et al.*, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, Aug 2019.

[6] A. Hamza *et al.*, "Combining Device Behavioral Models and Building Schema for Cyber-Security of Large-Scale IoT Infrastructure," *IEEE Internet of Things Journal*, pp. 1–1, Jul 2022.

[7] J. Anand *et al.*, "PARVP: Passively Assessing Risk of Vulnerable Passwords for HTTP Authentication in Networked Cameras," in *Proc. ACM Workshop on DAI-SNAC*, Dec 2021.

[8] A. Sivanathan *et al.*, "Detecting Behavioral Change of IoT Devices Using Clustering-Based Network Traffic Modeling," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7295–7309, Mar 2020.

[9] E. Lear and S. Rose, "Discovering and Retrieving Software Transparency and Vulnerability Information," IETF, Internet-Draft, Oct 2022.

[10] D. Kumar *et al.*, "All Things Considered: An Analysis of IoT Devices on Home Networks," in *Proc. USENIX Security*, Aug 2019.

[11] A. Sivanathan *et al.*, "Can We Classify an IoT Device using TCP Port Scan?" in *Proc. IEEE ICIAfS*, Colombo, Sri Lanka, Dec 2018.

[12] J. Bauer, "An Algorithm for IoT Device Identification," in *Proc. ICOIN*, Barcelona, Spain, Jan 2020, pp. 255–261.

[13] T. Karagiannis *et al.*, "BLINC: Multilevel Traffic Classification in the Dark," *ACM CCR*, vol. 35, no. 4, p. 229–240, Aug 2005.

[14] ——, "Profiling the End Host," in *Proc. PAM*, Louvain-la-neuve, Belgium, Apr 2007.

[15] M. Lyu *et al.*, "Classifying and Tracking Enterprise Assets via Dual-Grained Network Behavioral Analysis," *Computer Networks*, vol. 218, p. 109387, Dec 2022.

[16] B. Anderson and D. McGrew, "OS fingerprinting: New Techniques and a Study of Information Gain and Obfuscation," in *Proc IEEE CNS*, Las Vegas, NV, USA, Oct 2017.

[17] Z. Durumeric *et al.*, "A Search Engine Backed by Internet-Wide Scanning," in *Proc. ACM CCS*, Denver, Colorado, USA, Oct 2015.

[18] S. Marchal *et al.*, "AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication," *IEEE JSAC*, vol. 37, no. 6, pp. 1402–1412, Mar 2019.

[19] A. Pashamokhtari *et al.*, "Progressive Monitoring of IoT Networks Using SDN and Cost-Effective Traffic Signatures," in *Proc. IEEE ETSecIoT*, Sydney, Australia, Apr 2020.

[20] A. Hamza *et al.*, "Verifying and Monitoring IoTs Network Behavior Using MUD Profiles," *IEEE TDSC*, vol. 19, no. 1, pp. 1–18, May 2020.

[21] A. Pashamokhtari *et al.*, "PicP-MUD: Profiling Information Content of Payloads in MUD Flows for IoT Devices," in *Proc IEEE WoWMoM*, Belfast, United Kingdom, Jun 2022, pp. 521–526.

[22] M. Antonakakis *et al.*, "Understanding the Mirai Botnet," in *Proc. USENIX Security*, Vancouver, BC, Canada, Aug 2017.

[23] M. Lyu *et al.*, "Quantifying the Reflective DDoS Attack Capability of Household IoT Devices," in *Proc. ACM WiSec*, Jul 2017.

[24] A. Hamza *et al.*, "Detecting Volumetric Attacks on LoT Devices via SDN-Based Monitoring of MUD Activity," in *Proc. ACM SOSR*, San Jose, CA, USA, Apr 2019.

[25] ——, "Combining MUD Policies with SDN for IoT Intrusion Detection," in *Proc. ACM IoT S&P*, Budapest, Hungary, Aug 2018.

[26] E. Lear, R. Droms, and D. Romascanu, "Manufacturer Usage Description Specification," RFC 8520, Mar 2019.

[27] F. Gordon Lyon, *Nmap Network Scanning*, 12th ed. Insecure, 2009.

[28] F. Y. Ofir Arkin, "Xprobe v2.0:a "fuzzy" approach to remote active operating system fingerprinting," Aug 2002.

[29] R. Graham, "Masscan: the entire internet in 3 minutes," Sep 2013.

[30] "Angry IP Scanner Documentation," last accessed Oct 2022. [Online]. Available: https://angryip.org/documentation/

[31] R. D. Yates *et al.*, "Age of Information: An Introduction and Survey," *IEEE JSAC*, vol. 39, no. 5, pp. 1183–1210, 2021.