SERENIoT: Distributed Network Security Policy Management and Enforcement for Smart Homes

Corentin Thomasset
corentin.thomasset@polymtl.ca
Polytechnique Montréal
Montréal, Canada

David Barrera
david.barrera@carleton.ca
Carleton University
Ottawa, Canada

ABSTRACT

Selectively allowing network traffic has emerged as a dominant approach for securing consumer IoT devices. However, determining what the allowed behavior of an IoT device *should be* remains an open challenge. Proposals to date have relied on manufacturers and trusted parties to provide allow lists of network traffic, but these proposals require manufacturer involvement or placing trust in an additional stakeholder. Alternatively, locally monitoring devices can allow building allow lists of observed behavior, but devices may not exhaust their functionality set during the observation period, and the behavior may change following a software update which requires re-training. This paper proposes a blockchain-based system for determining whether an IoT device is behaving like other devices of the same type. Our system, SERENIoT, overcomes the challenge of initially determining the correct behavior for a device. Nodes in the SERENIoT public blockchain submit summaries of the network behavior observed for connected IoT devices and build allow lists of behavior observed by the majority of nodes. Changes in behavior through software updates are automatically added to the allow list once the update is broadly deployed. Through a proof-of-concept implementation of SERENIoT on a small IoT network and a large-scale Amazon EC2 simulation, we evaluate the security, scalability, and performance of our system.

CCS CONCEPTS

• **Security and privacy** → *Intrusion detection systems.*

KEYWORDS

IoT security, traffic filtering, intrusion detection, blockchain

ACM Reference Format:
Corentin Thomasset and David Barrera. 2020. SERENIoT: Distributed Network Security Policy Management and Enforcement for Smart Homes. In *Annual Computer Security Applications Conference (ACSAC 2020), December 7–11, 2020, Austin, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3427228.3427235

ACSAC 2020, December 7–11, 2020, Austin, USA

ACM ISBN 978-1-4503-8858-0/20/12…$15.00
https://doi.org/10.1145/3427228.3427235

1 INTRODUCTION

The rapid adoption of the Internet of Things (IoT) challenges well-established computer security strategies. Due to their deployment scale, IoT devices cannot be secured using traditional techniques such as anti-malware or network intrusion detection systems (NIDS). The diversity in IoT hardware and software combined with the deployment volume makes it difficult to design security systems that are effective yet not overburdened with management complexity. This is of particular importance in smart homes, where users are typically not security experts.

IoT devices are pervasive [21] and always connected. They are manufactured to be low-cost, so security is often not the primary design goal. As expected, numerous papers [4, 9, 29, 30] studying the security of IoT devices have found a steady stream of vulnerabilities (e.g., the Mirai botnet [5]) that pose a threat to users, to their environments, and to the broader global Internet infrastructure.

While IoT devices are diverse, one unifying characteristic is that their feature-set is generally simple; a device may sense its environment and submit readings to a cloud service (e.g., a humidity sensor), listen for inbound requests to perform some action (e.g., a WiFi light switch), or some combination of both. IoT devices by definition are not general purpose computers[1], and as such they do not require the network privileges of a general purpose system to perform their primary task. However, IoT devices are often treated indifferently from mobile phones, laptops, and other general purpose systems on networks, allowing any network communication that originates from the device to reach any host on the Internet. This over privilege allows compromised devices to directly attack remote hosts and services, or to act as steppingstones in more sophisticated attacks.

To prevent these attacks, existing NIDS systems could be used, but these require the user to configure operating parameters and tune the detection logic to avoid being overwhelmed by false positives. A conceptually simpler approach is the idea of allowing only a small set of network traffic to flow to/from an IoT device as needed. By allowing only the types of network activity that a device can generate or accept (which should roughly match the functional requirements of the device's primary task), the device can be constrained without requiring the modification of its onboard software. This is of particular interest in IoT, where devices may have long lifespans sometimes outlasting the manufacturer or software update support period. Moreover, false positives (i.e., blocking legitimate outbound connections) should be few and far between if the device can be accurately profiled.

[1] We note, however, that some IoT devices may be built on top of general purpose operating systems such as Linux.

Manufacturer Usage Descriptions (MUD [22]) standardize the policy language in which IoT network security policies can be written, so that the device manufacturer or a trusted third party can encode device behavior into a machine-readable policy. This policy can be enforced at the network edge, protecting all devices in the local network. The open question that remains is: *what network behavior should be included in the list of allowed traffic?* Requiring manufacturers to provide allow lists may not scale; there are too many unique IoT vendors, some of which simply re-brand devices manufactured by another vendor. A trusted third party could analyze devices and generate allow lists, but the business incentives (including user willingness to pay for such a service) aren't clear. Users themselves could analyze local device behavior and generate profiles, but this approach may not scale to households with large number of IoT devices.

In this paper we propose a blockchain-based network security policy management and enforcement system for home IoT environments. Our system, SERENIoT (pronounced *Serenity*), characterizes IoT device behavior locally and uses a decentralized ledger to determine whether the local behavior matches that observed by other peers in the network. Policies of allowed behavior[2] are the result of a consensus algorithm identifying network behavior observed by the majority of nodes in the network. Network connections that are unique to a device are blocked until they are observed by most nodes, preventing the spread of Mirai-style botnets.

SERENIoT is designed to run on network appliances such as home routers. The system analyses IP traffic between local IoT devices and their cloud companion services, making it compatible with all IP-based IoT devices and hubs. Since these appliances are usually already present in home networks (e.g., ISP-provided home routers), our system does not require any drastic network topology changes. SERENIoT extends the security features of home gateways by adding network policy enforcement for IoT devices.

Our contributions are:

(1) The design and implementation of a novel approach to build behavioral allow lists of IoT device traffic. The approach is based on blockchain and requires no opt in by manufacturers or trust in third parties. SERENIoT's public blockchain provides new data sources to audit IoT device behaviors at scale and assists in the detection of new threats.
(2) The evaluation of our system through large-scale simulations with 53 devices and 1000 nodes and on a small-scale testbed with real world devices.

The remainder of the paper is structured as follows. Section 2 reviews IoT security background and related work and gives a brief overview of relevant blockchain concepts. Section 3 presents the technical details of SERENIoT. Section 4 evaluates scalability, performance and security. Sections 5 and 6 present the limitations of our implementation and discuss related deployment issues. We conclude in Section 7.

[2]Security policies that permit only allowed behavior are more often referred to as *whitelists*. Throughout the paper we deliberately avoid this term in favor of the more descriptive *allow lists*.

2 RELATED WORK AND BLOCKCHAIN REVIEW

2.1 IoT security

One common solution to protect IoT networks is to deploy a signature-based Network Intrusion Detection System (NIDS) [13, 20, 24] on IoT networks. NIDS monitor network traffic and look for known attack signatures. These solutions are therefore only efficient if the attack is already known and require constant updates to have the latest signature base. Although these solutions might be workable for industrial IoT networks with dedicated security teams, complex IDS solutions are not suited for home environment where experts are likely unavailable to monitor, maintain, and configure them. IDSes can be augmented by using machine learning to detect previously unseen attacks. However, this introduces uncertainty as false positives can be exploited by attackers [33]. The accuracy issue is also present when identifying device types [26]. The similarity in behavior of distinct devices makes it difficult to determine which device generated the traffic, or what policy to apply to a particular device.

An alternative approach is to permit traffic based on policies describing the devices' expected behavior. This approach is sometimes referred to as specification-based intrusion detection, where the policy is a narrowly defined list of allowed behavior. The policies can be provided by the manufacturers or trusted parties as proposed by the IETF in RFC8520 [22] or generated by local device observation [7]. Yet another approach is to classify devices into controllers (e.g., smart phones and IoT hubs) and non-controllers (e.g., light bulbs), and prevent non-controllers from connecting to other devices. Non-controllers are given fewer network privileges, and are only allowed to connect to their cloud endpoints [12].

Anomaly detection capabilities can also be embedded into devices themselves [31]. The idea here is that the firmware on the device is updated to include an anomaly detection agent which monitors the system for malicious activity. Since this approach requires changes to software running on every IoT device, it is largely incompatible with devices that are currently deployed and no longer maintained. Moreover, it requires strong cooperation with manufacturers for adoption.

2.2 Blockchain review

We briefly review the key concepts of blockchain technology. A deeper treatment can be found in [32]. Blockchain technology addresses use cases where multiple distrusting parties want to jointly participate in a system. Blockchain provides shared governance where participants collaboratively decide what gets added to the chain and ensures that the protocol is being followed correctly by all the participants. Participation may be open (anyone can join, possibly without registration) or closed (only authorized participants can contribute).

A major aspect of blockchains is their verifiable sate: the data in a blockchain reflects the output of its consensus protocol which has been verified by all the participants. That is, only data that has been agreed upon through consensus can be added to the chain, leading the chain to contain only verifiable data. Once data has been verified by participants in the network, a new block containing this data is added to the chain. This data includes a cryptographic link to the previous block, allowing all parties to verify the continuity of the chain in addition to the validity of each block.

The consensus algorithm is thus a key aspect of every blockchain. It ensures that the chain of blocks containing the data is kept synchronized between participants so that they all have an identical copy of it at any time. It also prevents the blockchain from growing too rapidly by introducing a delay between the creation of new blocks. Multiple consensus algorithms exist [8]. The proof of work (PoW) [6, 17] algorithm is widely used by popular permisionless (open) blockchains such as Bitcoin[3] and Ethereum[4] and requires block hashes to be smaller than a defined target. In PoW blockchains, the weight of each participant' vote in the validation process is thus determined by its capacity to compute hashes and this mechanism ensures that participants are randomly selected to create new blocks. However, this approach is very costly from an energy and computational perspective. Indeed, all the effort has no utility beyond randomly delaying participants' capacity to produce valid blocks. Another approach is the proof of stake [19] which does not rely on computing hashes and thus avoids the massive energy requirements. With this algorithm, the creator of a new block is chosen within a pool of participants who have staked a certain amount of cryptocurrency. The penalty to harm the network is then the cost of losing the staked amount of cryptocurrency. For major blockchains this can amount to tens of thousands of dollars. A participant trying to take over the network would also need to own 51% of the cryptocurrency supply on that blockchain. That amounts to billions of dollars for major cryptocurrencies at the time of writing. It is thus less likely to happen than controlling half of the network hash power for proof of work [23]. However, this consensus mechanism requires a built-in cryptocurrency to work. Both of these consensus algorithms are widely used and provide a probabilistic way to verify the validity of blocks.

Finally, another feature of blockchains is data loss prevention. The decentralized nature of blockchains means that data in the chain is replicated across participants which allows recovery in case of data loss. At any time a participant can ask for a copy of the full chain and verify its contents.

Through these properties, Blockchain provides a tamper-proof decentralized ledger that can be used beyond cryptocurrencies in applications requiring accountability, transparency and trust in data [32].

2.3 IoT security and blockchain

Most closely to our work, Golomb et al. [10] propose CIoTA as a blockchain based anomaly detection system. CIoTA aims to build collaborative models of IoT devices' behavior at the device firmware level. Models are computed locally and validated by the consensus of the blockchain. The blockchain's ledger is then used to inform a client-side intrusion detection system producing alerts when anomalous firmware events are detected. While a preliminary security evaluation of CIoTA appears promising, it requires modification of the devices firmware to embed a software agent. SERENIoT learns device behavior at the network layer (see Section 3.3), and thus does not require any changes to the firmware enabling greater compatibility with existing devices.

Mendez Mena et al. [25] built and evaluated a blockchain based network filtering system for home networks. Their work focuses on the implementation of middleboxes called "gatekeepers" that enforce an allow list of actions on the network level. The allow list is computed based on the information stored in an Ethereum smart contract but their work does not detail how the smart contract is populated. While their study focuses on the enforcement aspects, SERENIoT presents a solution for both allow list enforcement and generation.

3 SERENIOT INTRUSION DETECTION SYSTEM

3.1 Overview

SERENIoT is a distributed specification-based intrusion detection system for home IoT networks. It monitors the network traffic to/from IoT devices to detect and block anomalous packets and connections. It relies on a decentralized ledger that characterizes devices' behavior and hosts a list of allowed packet signatures.

SERENIoT nodes (called *Sentinels*) are designed to be deployed on network appliances or middleboxes such as routers. A typical set-up would see one Sentinel deployed per smart home (see Figure 1), collaborating with other remote Sentinels to determine the correct network behavior of IoT devices. Sentinels advertise a WiFi network to which IoT devices connect, thus allowing mediation and filtering of all network connectivity between the devices and the Internet. The wireless network operates as a network bridge to the home local area network (LAN), so all traffic entering or leaving the Sentinel is monitored. Through its use of a distributed ledger and peer-to-peer communication, SERENIoT can operate with little-to-no user input. Moreover, compared to other network security solutions such as signature-based intrusion detection systems, SERENIoT's Sentinels are implicitly always up to date.

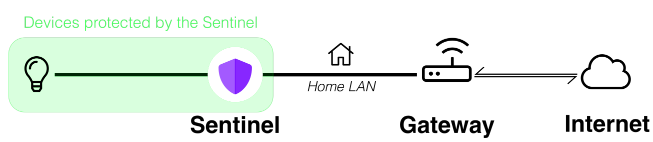

Figure 1: SERENIoT Network topology. Sentinels acts as middleboxes between IoT devices and the network gateway, enabling blocking of connections that are outside the device specification.

Concretely, Sentinels only forward packets that are defined in an allow list. Any network connection that is not specified in the list is discarded by the Sentinel. The allow list specifies network packet signatures characterizing the behavior of a specific IoT device as observed by the majority of Sentinels on the network. Allow lists for all IoT devices are stored in SERENIoT's blockchain. Through the use of blockchain, SERENIoT is fully decentralized and can be bootstrapped with a small number of Sentinels. It allows the system to be fully independent from trusted third parties, device manufacturers, and to support a large set of diverse IoT devices. We discuss additional motivation for building SERENIoT on top of

[3]https://bitcoin.org/
[4]https://ethereum.org/

a blockchain in Section 3.5. SERENIoT is fully backward compatible with many existing IP-based IoT devices, requiring no changes to their hardware, firmware, or apps. The system is also designed to be forward compatible with devices that don't yet exist, as long as they use IP-based communication and can connect to the local Sentinel.

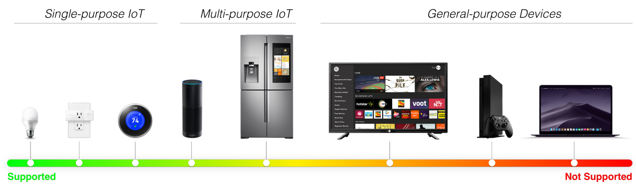

Figure 2: Supported devices for SERENIoT. Devices toward the left have simpler network behavior and tend to have a similar network footprint shared across devices of the same type. Devices toward the right have unique network footprints determined by their users.

We built SERENIoT with a focus on consumer IoT, noting high accuracy and performance when working with feature-limited devices such as smart bulbs, smart switches, smart locks, smart thermostats, etc. (see Figure 2). These devices typically only interact with a small set of cloud services through well-defined APIs, thus their network footprint can be accurately determined (see Section 4.3). According to a 2019 study [21], this targets approximately 41% of devices deployed in North American homes, and 28.4% of devices in Western Europe[5]. SERENIoT cannot support systems with variable (typically human-dependent) network behavior, since each system may create a unique set of network connections.

3.2 Threat model

SERENIoT protects devices against attackers trying to change their behavior, as widely used by botnets [1, 5]. SERENIoT has been designed to defend against the two following attack scenarios:

- An IoT device has been compromised locally by a malware trying to change its behavior to accomplish malicious actions. The attack vector can vary; the IoT device can be infected by another device on the local network (for example by an infected computer or IoT device), or the infection can be the result of a physical action on the device (for example a memory card swap). In this situation SERENIoT would protect the IoT device from attacking targets on the internet by blocking all the outgoing traffic deviating from the specification.
- An IoT device is directly exposed on the internet. SERENIoT would protect the IoT device from incoming attacks by blocking all incoming traffic differing from the specification. Most IoT devices don't normally receive incoming connections from the internet and SERENIoT will then behave as a firewall blocking all incoming connections.

Once an allow list has been populated for a device, an attacker would need to change the behavior of more than 50% of the IoT devices of the same type to change the specification and allow the attack to go through (see Section 4.5).

3.3 Sentinel architecture

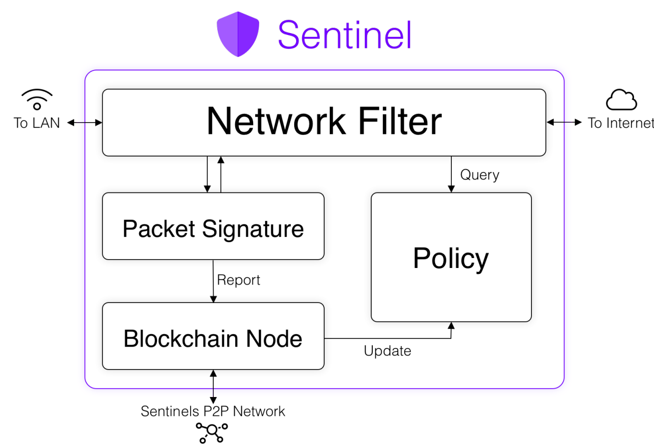

Figure 3: Main components of a Sentinel. Description inline.

Sentinels use a modular architecture with 4 main components (see Figure 3): The network filter component (1) is in charge of enforcing the policy by dropping network packets that are not allowed and forwarding acceptable traffic. The network filter relies on the packet signature module (2) to serialize the raw IP packet into a textual signature and on the policy module (3) that lists all the allowed packet signatures. Finally, the blockchain module (4) keeps the policy updated by synchronizing the ledger with the other Sentinels and by reporting the newly recorded packet signatures.

3.4 Computing packet signatures

Packet signatures allow Sentinels to characterize recorded packets. An effective signature algorithm should be precise enough to differentiate packets from different network connections but flexible enough to produce the same signature across devices of the same model. Unlike general purpose computers whose behavior changes depending on their usage, IoT devices of a same model behave similarly (often identically in terms of traffic transmitted and destination) and produce similar network traces. We have verified this hypothesis during our evaluation in Section 4. Our proof of concept uses packet signatures at the IP level and focuses on the fields that remain constant across devices. While other packet signatures and connection fingerprinting techniques exist, we use a NetFlow[6] like representation to strike a balance between uniqueness and consistency across devices. SERENIoT's packet signature aggregates sequences of packets sharing the following values:

- Protocol of the IP payload
- Endpoint (domain name or IP address if domain is unavailable)
- Service port

[5]We include all IoT devices in the study by Kumar et al. [21] except media boxes, game consoles, and file storage appliances which are functionally as complex as general purpose computers

[6]https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html

The endpoint identifies the remote host with which the IoT device is interacting. To resolve potential domains, we perform reverse DNS lookups. The service port identifies the well-known port number used by the connection[7]. Packets of a same flow will share the same signature that will be used by SERENIoT to identify anomalous flows and packets. Signatures are computed by serializing and hashing the following values:

$$Signature = H(protocol, endpoint, service\ port)$$

Packet signatures don't include device-specific identifiers such as Media Access Control Organizational Unique Identifiers (OUI). Indeed, it is unclear whether all devices of a same model will share a single OUI as many manufacturers are allocated more than one. Two identical devices with different OUIs would be assigned to different chains weakening the security of both chains. Moreover, malicious code running on an IoT device may be capable of manipulating the MAC address. Our choice of packet signatures allows to differentiate packets going to untrusted hosts from those going to the manufacturer's API. It also allows to differentiate packets initiated by the monitored device from those initiated by a remote entity in the case of IPv6 network or networks without NAT where devices are directly exposed on the internet.

Note that SERENIoT does not precisely identify devices. Devices that produce the same set of packet signatures are grouped and the system assumes they are of the same type. Devices are characterized by their packets signatures and device types are fingerprinted by hashing their sorted set of packet signatures.

3.5 SERENIoT's blockchain

The blockchain is the key behind SERENIoT's collaborative policy generation mechanism. It ensures that all the packet signatures written to the policy are agreed upon through a distributed consensus protocol. This provides robustness and trust by making sure malicious signatures are not added to the allow list as long as a majority of Sentinels participating in the network observe legitimate behaviors on their local IoT devices.

Our choice of building SERENIoT on blockchain stems from 3 core design requirements: (1) To allow SERENIoT to operate independently. Blockchain distributes the data hosting across users allowing SERENIoT to be independent from any third party. (2) To deploy a highly available system providing security policy updates at very low cost. (3) To make the system available without restriction of use to certain brands of devices. We evaluate our blockchain-based system in Section 4.

To implement SERENIoT, we have designed a custom public blockchain based on Bitcoin's blockchain principles but with no inherent cryptocurrency. We elected to build a custom blockchain because current blockchain frameworks are either strongly tied to token economics (e.g., Ethereum[8]) or designed to build permissioned blockchains (e.g., Hyperledger fabric[9]). Designing our own chain gives us the flexibility to include only features specific to our requirements while avoiding compatibility challenges arising from trying to retrofit another framework to our use case.

3.5.1 Ledger. SERENIoT's ledger contains packet signatures reported by the Sentinels. It is based on a distributed timestamp server chaining data blocks together. The linked timestamping mechanism ensures that blocks cannot be rearranged or modified without invalidating subsequent blocks in the chain. As the blockchain grows, Sentinels converge on the chain with the most blocks. Figure 4 illustrates a sample chain for a specific device.

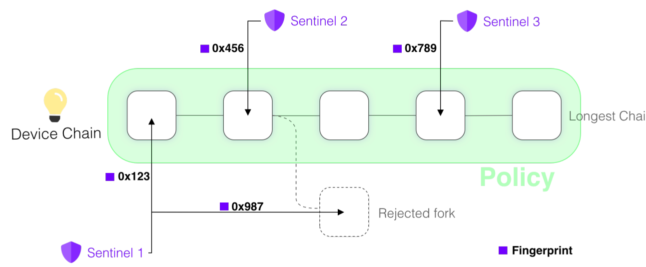

Figure 4: Device chain: The Sentinels add packet signatures into blocks. The chain grows and only signatures listed into the longest chain are trusted.

In our implementation, blocks store a list of packet signatures reported by the Sentinels instead of a Merkle root of transactions as in Bitcoin (see Figure 5). The complete list of reported signatures is indeed necessary to build the policy and there is thus no need for selective reveal.

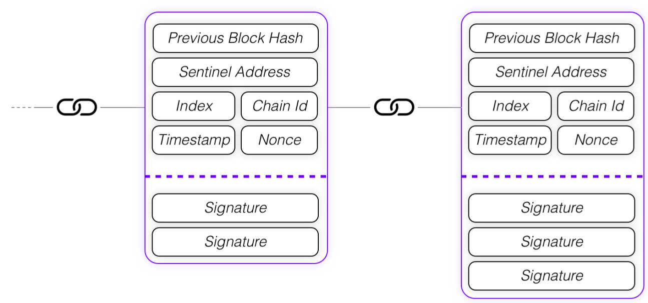

Figure 5: SERENIoT's Block architecture. The Sentinel Address is a unique identifier generated at Sentinels start up using the same algorithm used to generate Bitcoin's addresses.

To extend the chain and report new packet signatures to the system, Sentinels only work on top of blocks that contain signatures that they have previously observed. That is, Sentinels avoid appending to chains that include unknown signatures. These packet signatures may be malicious or reflect previously unseen connections for a device. Thus, the fastest growing chain always contains the most common packet signatures that have been observed by a majority of Sentinels. This mechanism is described in detail in Section 3.5.3.

The policy is a cumulative set of allowed packet signatures that have been included in confirmed blocks since the chain genesis.

[7]The service port generally refers to the remote endpoint's port. However, some IoT devices (e.g., cameras) host certain services locally, in which case the service port refers to the local port hosting the service. To differentiate between local and remote services, we append a direction identifier (L for local or R for remote) to the service port.
[8]https://ethereum.org/
[9]https://www.hyperledger.org/projects/fabric

Note that this policy may allow behavior that is no longer necessary to the device to operate (e.g. a feature that was removed through a software update). Future work will explore removing outdated signatures.

SERENIoT extends this concept and uses a multichain architecture with one device chain / allow list to track the behavior of each protected IoT model (see 3.5.5).

3.5.2 Consensus. SERENIoT's consensus algorithm ensures that the allow list is kept synchronized between Sentinels so that they all converge to an identical copy of the blockchain. It is also responsible for making sure that the fastest growing chain gathers the most Sentinels. To facilitate the development of our proof of concept, we implement a proof of work consensus algorithm [28]. We discuss alternative consensus algorithms in Section 6.1.

With proof of work, blocks are produced by nodes racing to solve computational puzzles. The node that solves the puzzle appends its block to the chain. Each additional block increases the effort required to rewrite the longest chain, since changing a past block would require every subsequent proof of work to be recomputed. As long as the computational power distribution remains balanced across Sentinels, the fastest growing chain will gather the most Sentinels.

3.5.3 Sentinels workflow. Sentinels participate in maintaining and updating the policies by serving as blockchain nodes. Sentinels only subscribe to the policies corresponding to the devices they locally monitor. A Sentinel's blockchain node process can be described as follows.

(1) The Sentinel monitors IoT devices that are connected to it and collects new packet signatures into allow list block candidates. One allow list block candidate is created per subscribed allow list. If a device is inactive or no new packet signatures have been recorded, the Sentinel builds an empty block.
(2) The Sentinel computes the hashes of the allow list block candidates' headers and adds them to a control block candidate.
(3) The Sentinel works on solving the proof of work for the control block candidate.
(4) The first Sentinel to produce a control block is selected to append its allow list block candidates to the corresponding list. To do so, it broadcasts the control block along with all the allow list blocks listed within.
(5) Sentinels always accept broadcasted control blocks. Sentinels only accept a broadcasted allow list block if they are registered to the corresponding allow list and if they recognize all its packet signatures. When accepting a block, Sentinels work on extending the chain on top of that block. An allow list block is only valid if its block header is listed in a control block. Sentinels always converge on the longest chain and forks are resolved when a branch becomes longer than the others.

3.5.4 Adding incentive for open networks. We designed SERENIoT's blockchain to work with no inherent cryptocurrency. Thus, Sentinels that contribute to the network by providing computational power cannot be rewarded with some cryptocurrency. To encourage Sentinels to stay active and contribute to the blockchain, inactive Sentinels are isolated by their neighbors and do not receive the latest allow list updates. To signal their activity and contribution to the network, Sentinels use a mechanism inspired by mining pools and broadcast partial proof of work solutions to the problem they are trying to solve. This proves to their neighbors that they are active and contributing to the system. Isolated Sentinels gradually become less useful since they are no longer able to verify newly recorded packet signatures. This in turn prevents them from differentiating between normal and abnormal behavior as observed by a majority of Sentinels.

3.5.5 Multichain. The logic described thus far works well for one specific IoT device. Indeed, every IoT device protected by SERENIoT needs its own blockchain as Sentinels cannot adjudicate on blocks containing packet signatures for unknown devices. SERENIoT uses a multichain solution allowing Sentinels to subscribe to the allow lists concerning the devices they protect. Thus, each device type uses a separate blockchain to track its behavior. When a new device is connected to a Sentinel, the Sentinel profiles it and assigns it to a chain aggregating similar devices. To profile a device and subscribe to the right chain, Sentinels observe device behavior during a short profiling phase upon its connection. Once the profiling phase is complete, Sentinels compute the allow list identifier corresponding to the device. This identifier is computed by hashing the sorted list of packet signatures collected during the profiling phase. Note that SERENIoT does not identify devices precisely. Instead, devices that produce the same set of packet signatures are grouped together and the system assumes they are the same type. Thus, devices are characterized by their packets signatures and device models are fingerprinted by hashing their sorted set of packet signatures.

To support multiple IoT devices, SERENIoT's Blockchain is composed of one control chain and multiple device-specific chains (also referred to as allow lists), one for each device model protected by the Sentinels. In our implementation, the control chain stores the block headers of valid allow list blocks and uses the proof of work consensus mechanism. Device-specific chains do not have an independent consensus mechanism, they instead leverage the control chain's proof of work.

The control chain improves robustness by requiring all Sentinels (regardless of their locally monitored IoT devices) to ultimately contribute to a single global chain while building on device-specific blockchains. This increases the effort required for an attacker to target a specific unpopular allow list to rewrite it. Indeed, all blocks in device-specific chains are validated in the control chain that gathers all the Sentinels of the network.

3.6 Detecting behavior changes

SERENIoT is designed to protect IoT devices with limited functionalities such as smart bulbs, smart outlets or smart cameras. These devices typically establish a small set of network connections so we can characterize their expected behavior by observing the network traffic of a large number of devices of a specific model. Sentinels use the most common behavior observed by nodes in the system to create a specification of what the observed device should be allowed to do. Specification-based intrusion detection systems raise alarms when behavior deviates (even slightly) from a narrowly

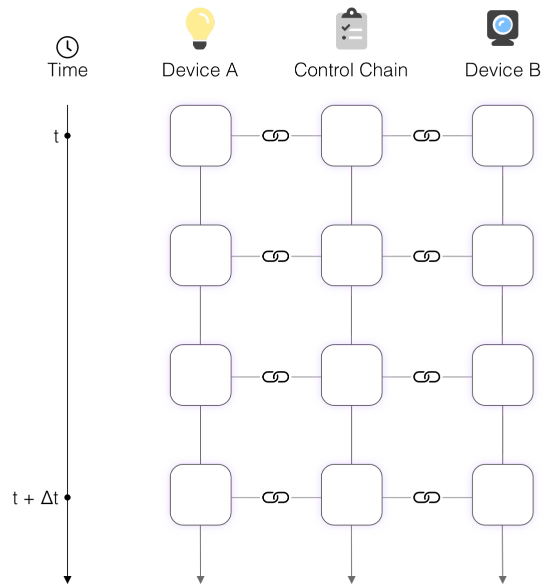

Figure 6: Multichain support in SERENIoT. Block headers of device specific blockchains are incorporated into a single control chain.

defined specification. Network traffic is either permitted or blocked, with no notion of confidence or likelihood of attack, as is the case with anomaly-based IDS. Specification-based IDS is also different from signature-based IDS, where experts define signatures of all known attacks. SERENIoT seeks to define signatures of known good behavior and block all other network traffic.

To do so, Sentinels use allow lists to record the packet signatures characterizing devices' intended behaviors. When a packet from an IoT device is recorded by a Sentinel, the Sentinel computes its signature and verifies whether the signature exists in the allow list for the device. If the signature is trusted, the packet is forwarded to its destination. Otherwise the packet is blocked, and the Sentinel reports the packet signature (i.e., adds it to current block candidate of the allow list). It will eventually be appended to the allow list if the majority of Sentinels also report it. This mechanism allows Sentinels to detect and block anomalous behaviors that are only observed on a small proportion of monitored devices.

3.6.1 Updating the policy. When a device's behavior changes, other Sentinels on the network report whether they also observed the change. The behavioral change can be the result of a firmware update or of an attack. To decide if the new behavior is legitimate, Sentinels rely on the majority's observation: if the change has been observed by the majority of Sentinels, it is considered as legitimate and will be added to the allow list. Otherwise it will be considered as anomalous and will be blocked. This logic is based on the idea that if the majority of devices of the same type share the same behavior, this is their "intended" behavior. Note that intended behavior may itself be anomalous: for example, in January 2020, Google revoked Xiaomi's access to the Google Home Hub ecosystem after users were able to view the video feeds of strangers' security cameras [2]. Theses cases, however, can be better addressed by treating the device itself as untrusted and taking action against the manufacturer.

3.6.2 Transparency & auditing . In addition to Sentinels network filtering capabilities, the open and public nature of SERENIoT's blockchain introduces a new data source for cyber security experts, allowing them to follow and audit in real time the behavioral evolutions of IoT devices. This can be used to monitor emerging threats, update adoption rates, etc. For example, it is possible to measure the spread of a growing botnet by monitoring rejected forks. Transparency and privacy concerns are discussed in Section 6.2.

3.7 Device onboarding

When a new IoT device is added to a local network protected by SERENIoT, the Sentinel first goes through a profiling phase to fingerprint the device before enforcing network filtering. This process is detailed below:

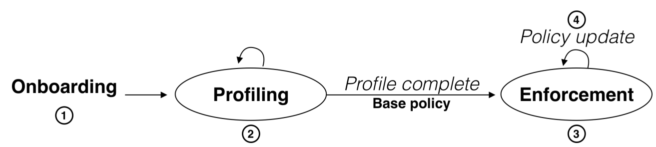

Figure 7: Device onboarding flow-chart.

(1) **Onboarding:** The user buys a new device and connects it to the Sentinel. To connect the device to the Sentinel, the user uses the dedicated WiFi network broadcasted by the Sentinel.
(2) **Profiling:** During the profiling phase, the Sentinel allows all network traffic to and from the device and computes packet signatures for all the network connections to characterize the device. The connection signatures are used to register the new device to the blockchain regrouping all the devices with a similar network footprint. This profiling phase usually requires about 1 minute for most IoT devices tested during our evaluation (see Section 4.3). If the corresponding blockchain does not exist (i.e., it is the first device with this network footprint to be connected to a Sentinel), a new device specific chain is initialized.
(3) **Enforcement:** Once the device has been registered to a blockchain, the Sentinel downloads the blockchain to build the policy and begins blocking all the packets whose signatures don't match the policy. The Sentinel also starts reporting newly recorded packet signatures to the allow list.
(4) **Behavior change / Policy update:** If the behavior of the protected devices changes, the Sentinel will vote to decide whether these changes need to be incorporated into the policy based on the majority's observations. As expected, if the majority of nodes on a chain are malicious and controlled by a single attacker, it will be possible to incorporate anomalous packet signatures into the allow list. We discuss this issue further in the security evaluation Section 4.5

4 EVALUATION

To ensure correct system behavior, we conducted a small scale experiment using real IoT devices. We then tested the compatibility, scalability and robustness of SERENIoT using larger scale simulations on Amazon AWS.

4.1 Implementation

The proof of concept of the SERENIoT Sentinel is developed in node.js[10], a cross platform, open source javascript runtime environment designed to build event-driven and asynchronous web applications. We implemented the blockchain component from scratch, without the use of existing blockchain frameworks. Sentinels communicate with peers using WebRTC[11] and websockets[12]. We use netfilterqueue[13] to intercept, inspect and block network packets forwarded by the Sentinels. In addition, we developed a Web UI using the VueJs framework[14] to monitor Sentinels in real time and to manage experimental instances. Screenshots of the Web UI are presented in Appendix A.1.

4.2 Functional real world experiment

The goal of our experimental set-up (see Figure 8) was to simulate the network topology of a real IoT network. We installed the Sentinel software on 3 Raspberry Pis (model 3B+) configured as WiFi hotspots and connected one LIFX Mini Smart bulb to each Sentinel. The router plays the role of the home gateway.

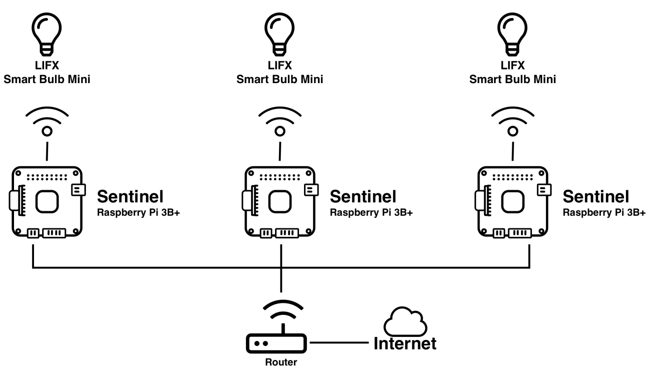

Figure 8: Physical devices used for our real experimental set-up.

The goal of the real world experiment was to validate the operation of SERENIoT by testing it on real IoT devices. Sentinels were initialized with no prior knowledge of the devices' behavior and with empty security policies. During the experiment we interacted with devices through the manufacturers' mobile app.

This first experiment validates the concept behind SERENIoT; The 3 Sentinels were successfully able to record the packets from the bulbs and to converge on the list of the resulting packet signatures shown in Table 1. A security policy was successfully generated. We also noted during this experiment that there was no perceptible delay introduced by the Sentinel and we were able to interact in real time with the bulb through the LIFX mobile app. While we did not conduct a thorough performance overhead analysis in this experiment, SERENIoT posed no noticeable interference with between the smart bulbs and our commands.

Recorded packet	Pkt. Signature	Desc.
UDP time1.google.com R123	0cca40...aed4d4	NTP
TCP 104.198.46.246 R56700	4e2b3d...2a4474	LIFX API

Table 1: Packet signatures recorded for the LIFX Smart Bulb during our experiment. Signatures have been truncated.

4.3 Compatibility and scalability simulation

For testing compatibility and scalability, we set-up a virtualized testbed with 1000 Sentinels on Amazon AWS. Each Sentinel was run in a separated Docker container and we used Docker Swarm to orchestrate our cluster and deploy the Sentinels' containers on AWS instances. We used 10 Amazon EC2 c5.2.xlarge instances hosting each 100 Sentinels and connected through a Docker virtual network.

Each Sentinel was simulating a set of devices from the dataset of Alrawi et al. [4]. This dataset provides packet captures of an IoT network with 53 different devices for 9 continuous days. Devices were simulated by replaying these packets in random order and at random intervals to imitate the unpredictable aspect of user interactions (for a example a smart light bulb might be powered off during a period of time and the user can interact with it at any time).

4.3.1 Dataset analysis. Based on these captures, we extracted the devices' behavior by isolating packets generating unique signatures. Figure 9 shows the evolution of the number of unique packet signatures for the devices in the dataset. We clearly denote two classes of IoT devices: devices with a simple behavior characterized by a small number of packet signatures as LIFX Smart Bulb, TPLink WiFi plug or Nest Guard and other more complex multipurpose devices such as iPads, smart TVs, etc. We also observe on Figure 9 that devices with a simple functionality are characterized by a stable behavior over time that does not change often. This validates our initial hypothesis that IoT devices' typical behavior only contains a small set of actions which remain constant over time.

4.3.2 Simulations. The goal of the simulations was to validate the compatibility of SERENIoT with multiple simulated devices and to validate the scalability of the system on a larger scale experiment. To do so, we ran multiple simulations for different time periods (from 1 hour to multiple days) and with different number of Sentinels (from 20 to 1000). The Sentinels were initialized with no prior knowledge of the devices behavior and with empty security policies before each simulation.

Sentinels were able to converge and produced lists of trusted packet signatures for simple devices. Sentinels were also able to identify and block anomalous packets injected in a small number of Sentinels' devices' behavior while keeping the device functional.

[10]https://nodejs.org/
[11]https://webrtc.org/
[12]https://developer.mozilla.org/docs/Web/API/WebSockets_API
[13]netfilterqueue is a wrapper around libnetfilter_queue that gives access to the packets matched by specific iptables rules. More information: https://netfilter.org/projects/libnetfilter_queue/
[14]https://vuejs.org/

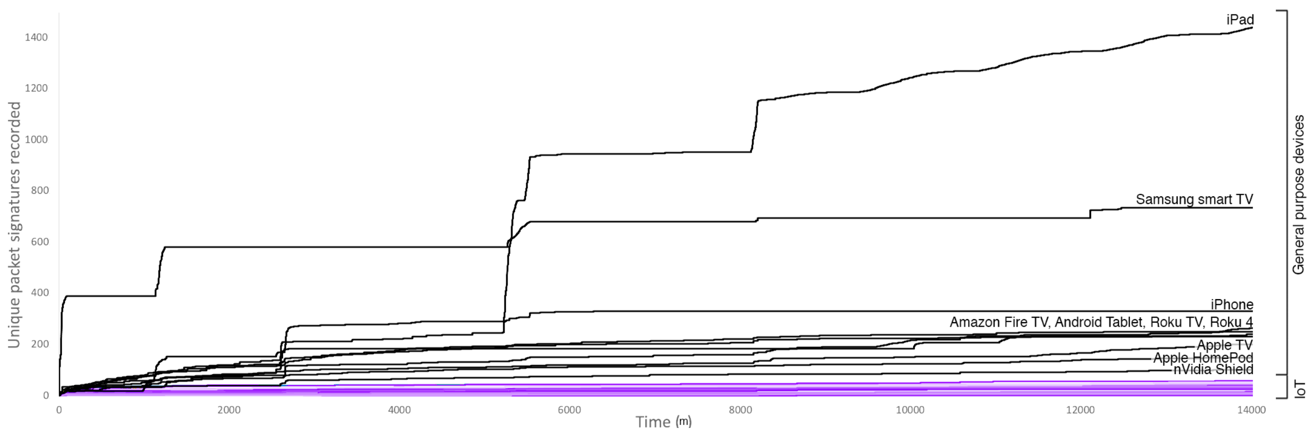

Figure 9: CDF of distinct packet signatures per device recorded over a 9-day period. Labeled lines identify general purpose devices.

A screenshot showing the identification and rejection of a fork with anomalous packet signatures can be found in Appendix A.1. However, they were unable to converge on a security policy for general purpose devices as iPads, iPhones and Android tablets.

This experiment show that SERENIoT also behave as expected with a larger set of simulated devices. Sentinels are able to generate security policies for IoT devices with a simple network footprint and to identify and block anomalous packets. We have observed during the simulations that the breaking point where behavior changes are incorporated in the trusted list of packet signatures usually happens when 51% of the Sentinels record a same packet signature. This means that 51% of the Sentinels need to record a same packet signature to be authoritative on the longest chain and include the packet signature in the allow list for a given device. Thus, popular devices are less likely to be attacked as more Sentinels need to be infected to incorporate malicious packet signatures in the allow list. However, they also require more time for updates to be deployed as updates need to reach a greater number of devices before being trusted.

4.4 Blockchain performance evaluation

This section evaluates the capacity of our system to run over long periods of time. During our experiments we measured the growth of the blockchains and monitored the runtime metrics of the Docker containers running the Sentinels.

Blockchains size. Sentinels store blocks as JSON files. To measure the blockchain growth we connected to different Sentinels during a 24-hour experiment and recorded the number of stored blocks as well as the size of the blocks directory after 1 hour, 5 hours and 24 hours. For this experiment we used 20 simulated Sentinels with one Belkin Netcam connected. The number of Sentinels in the network does not influence the blockchain size as the block production rate is fixed and determined by the consensus algorithm. Sentinels also delete rejected fork blocks as soon as they converge on a longest chain.

Table 2 shows that the control chain block size tends to be constant over time. The block size for the control chain is determined by the number of different IoT devices types protected by the Sentinels. Indeed, each device type has is own device chain and each device chain is indexed into the control chain. Control chain blocks list the block headers of the latest produced blocks from the device chains, containing at maximum the number of device chains, block headers. Block headers are SHA256 hashes and have a fixed size of 32 bytes. It is thus straightforward to compute the control chain block size for a given number of IoT devices. Based on our measurements, the control chain with 1 IoT device should be around 511MB after one year running.

$$1.4MB * 365days = 511MB/year$$

If we consider 10K different IoT device types protected by our Sentinels, the control chain should be around 511GB after one year running.

$$4384blocks * 10000 * 32B + 1.4MB = 1.4GB/day$$

$$1.4GB * 365days = 511GB/year$$

Table 3 shows that the device chain block size also tends to be constant. Indeed, most of the blocks in device chains are empty as blocks list packet signatures of newly observed behaviors. In the long run, the majority of blocks will thus be empty as new behaviors are rarely recorded. Based on our measurements, device chains should be around 474MB after one year running.

$$1.3MB * 365days = 474MB/year$$

Sentinels are required to maintain a copy of the control chain. However, they only need to download and maintain the device chains corresponding to the device types they protect. Thus, a Sentinel with 10 different IoT devices would need to maintain a copy of the control chain and 10 device chains.

Future work will explore a block expiration feature where outdated blocks will be deleted. This feature would prevent the blockchains to grow infinitely while allowing old policies to be updated by deleting outdated behaviors no longer in use by the majority of the devices.

Elaps. Time	No. of blocks	Size	Avg. block size
1 hour	205	64KB	312B
5 hours	918	291KB	316B
24 hours	4384	1.4MB	316B

Table 2: Block size measurements for the control chain with 1 IoT device.

Elaps. Time	No. of blocks	Size	Avg. block size
1 hour	193	59KB	304B
5 hours	906	268KB	296B
24 hours	4367	1.3MB	295B

Table 3: Block size measurements for a device chain.

Sentinels runtime metrics. To record Sentinels' metrics, we used the *docker stats* command[15]. We recorded the metrics for 20 simulated Sentinels during a 24 hours experiment after 1 hour, 5 hours and 24 hours. These metrics show the CPU, memory and network usage. We observed that Sentinels use approximately 140MiB of RAM after running for 24 hours (with an initial usage of 120MiB of RAM) and that their network usage is correlated with the blockchains growth. Sentinels only download blocks for the control chain and for the device chains they are registered to. The network usage varies between two Sentinels based on the number of different IoT devices they are protecting. Finally, Sentinels were using a simulated proof of work consensus algorithm to run the experiment. Their CPU usage is thus not representative of the usage one would observe if the Sentinels were using real proof of work instead. In the case of real proof of work we expect CPU usage to be maxed out at 100% for all Sentinels.

4.5 Security evaluation

When a new IoT device is connected to a Sentinel, the Sentinel determines the corresponding allow list based on the device behavior. Thus, devices behaving similarly will be grouped on the same allow list and already compromised devices behaving differently will be assigned to a separate list.

Uncompromised devices are thereby grouped, and the corresponding allow list will only contain packet signatures reflecting the behavior of these devices. In this section, we consider the different attack vectors that may lead to successful attacks incorporating malicious packet signatures into an allow list or exploiting devices to change their behavior.

Attacks against IoT devices during profiling. During the profiling phase (see Section 3.7), Sentinels allow all the traffic and don't enforce any network filtering for the newly connected IoT device. Even if this phase only lasts a few minutes, an attacker could use this window to perform an attack. In this case, the attack will modify the network footprint of the device which will likely cause it to be registered to a different chain than other benign devices of the same type. This chain will regroup all the devices of this type that have produced the same network footprint during the profiling phase (i.e., all the devices of the same type that have been targeted by the same attack during the profiling phase) and the Sentinels will not filter the resulting malicious network connections.

Devices may already be infected when initially connecting to Sentinels. In this case, if the infected devices' behavior is similar to benign devices of the same type, they will be registered to the same device chain. However, if the infected devices' behavior is different, they will be registered to a chain with similarly-infected devices.

Attacks against IoT devices during enforcement. Once the Sentinel is in the enforcement phase (see Section 3.7), we differentiate two types of attacks against the IoT devices. Attacks from the local network and attacks from the internet.

Attacks from the local network will eventually succeed and compromise devices as Sentinels do not enforce any network filtering on the local network. However, Sentinels will block any behavior deviating from the specification trying to reach the internet. This protects against compromised local IoT devices trying to attack targets on the internet and restricts them to the strict behavior listed in the specifications. Due to the open nature of its blockchain, SERENIoT also provides new metrics allowing experts to monitor the behavior of a large number of IoT devices in real time.

Attacks originating from a remote attacker targeting a specific IoT model will be blocked by Sentinels as long as only a minority of Sentinels observe the same attack pattern. Thus, these attacks grow in difficulty with more Sentinels protecting more devices of a specific model. For an attack to succeed, its network footprint has to be similar on a majority of Sentinels (meaning that Sentinels should record packets signatures with the same IP address). The attacker also needs to target 51% of all devices in less than one block-interval (the time interval between blocks). For popular IoT devices, attacker unlikely have the resources to initiate an attack targeting simultaneously a large number of devices from a single host. Commonly, attackers rely botnets to carry massive attacks against IoT devices. IoT botnets such as Mirai [5], Brickerbot [18] and Hajime [14] share similar network footprints during the infection phase: they scan for listening telnet, ssh and http services and try to bruteforce the credentials. However, the source IP addresses of the attacker vary as botnets use infected devices to spread and infect new hosts. The packet signatures of each botnet attack will thus likely vary from one target to another as the IP and port used by the attacking device will vary. By design, SERENIoT should effectively block P2P botnets, particularly those having multiple attackers. Because these signatures are not consistent across all Sentinels, they are unlikely to be added to the allow list and the attack will be blocked.

In both cases, unpopular devices are more prone to attacks because packets are filtered based on policies built from observations from a smaller number of Sentinels. However using SERENIoT to protect unpopular devices still provides a better protection than using no protection as long as an attacker does not control more than half of the Sentinels for a particular device.

Attacks on Sentinels. Sentinels act as both blockchain nodes and network traffic enforcement points. Thus, if a Sentinel is compromised, the attacker may insert or remove traffic rules arbitrarily[16].

[15]More information on the Docker runtime metrics can be found here: https://docs.docker.com/config/containers/runmetrics/

[16]This is not unlike the security of a firewall or router, which should typically be better protected than internal hosts on the network.

However, attacking a specific node does not allow the inclusion of malicious packet signatures into the blockchain, as these must still be validated and confirmed by the majority of the nodes. SERENIoT's blockchains are vulnerable to two types of majority attacks.

- Majority attacks against the control chain can succeed if an attacker has the computational capacity of more than half of all Sentinels. This adversary can win every proof-of-work round, allowing the inclusion of arbitrary packet signatures into any device blockchain/allow list. This type of attack is devastating to the network since allow lists are shared amongst all Sentinels. We discuss alternative consensus protocols to lower the probability of such an attack in Section 6.1.
- Majority attacks against device chains can succeed if an attacker controls more Sentinels than half of all Sentinels registered on a specific device chain. This adversary will be able to append blocks to these chains more often than legitimate Sentinels which can result in incorporating malicious blocks into the longest device chain. This attack grows in difficulty with more Sentinels protecting the same device of a specific model and contributing to its chain. For popular IoT devices, with a large number of Sentinels registered on their chain, this attack's difficulty is similar to a majority attack on the control chain.

5 LIMITATIONS

Our system currently only monitors LAN-to-WAN connections so it does not protect IoT devices from other infected devices on the local network. While this limitation allows P2P infection methods to succeed on the local network, infected IoT devices will be unable to attack remote hosts; Sentinels will block the outgoing traffic that does not comply with the policy.

Our system is tailored to support IoT devices with a small network footprint. It is unclear, however, how effective the system can be in protecting IoT devices with more diverse network behavior. An open question is whether any collaborative intrusion detection system (ours included) can converge on a set of connections that should be permitted. One strategy for complex devices is to make the packet signature algorithm less specific, but this may have the disadvantage of missing certain attacks.

In its current design, SERENIoT will block any user-defined connection to remote servers (e.g., a cloud-based FTP server for video stream backups). Connecting to user-defined endpoints will typically be blocked because connections to these arbitrary servers will not be observed amongst the broader population of Sentinels. One option to permit user-defined servers to be allowed is to enable a manual override in the user interface, allowing advanced users to allow specific connections without impacting the distributed allow list. We will explore adding this feature in future work.

Finally, like any collaborative system, performance (both accuracy and resilience to attack) improves with the deployment of each additional node. By deploying more Sentinels, manipulating the blockchain (and therefore the allow lists) requires more effort from an attacker. Similarly, in the case of a small number of Sentinels, the likelihood of an attack influencing the packet signatures that get added to the allow list is higher.

6 DISCUSSION

6.1 Consensus

In our proof of concept, we used Proof of Work for its implementation simplicity and wide availability. However, we expect that most Sentinels will be deployed on devices with limited computational power such as small office and home routers. Since proof of work relies on computationally intensive problems to secure the chain, an attacker with large computing resources can easily overpower even a large number of routers. To mitigate this, an alternative consensus algorithm could be used. In addition to the consensus algorithms mentioned in Section 2.2, two promising alternatives are *proof of elapsed time* (PoET) [15] and *robust round robin* (RRR) [3]. These algorithms both leverage Intel's Safe Guard Extension (SGX) [16]. PoET has the same objective as proof of work: to randomly delay block production so that it is evenly spread across the network over time. To do so, nodes are required to run the code generating the delay inside Intel's SGX to certify they effectively wait a random time at each block production round. On the other hand, RRR selects nodes alternately to distribute block production across all participants. In RRR, node selection is based on elapsed time since last block mined; the node that has not mined a block for the longest time is selected. To join the network, nodes require a unique identity provided by SGX. This ensures that each participant is unique, as it is not possible to generate multiple identities with a single SGX chip. Unlike PoW, PoET and RRR do not require heavy computational effort and thus are less energy consumptive. However, these methods also have limitations: they are tied to a given chip vendor and rely entirely on a third party platform to work exposing them to vulnerabilities [11, 27, 34, 35].

6.2 Transparency and privacy

Blockchain systems, in particular those which are public tend to elicit privacy concerns since the ledger is replicated across all participants. By using SERENIoT, IoT device behaviour is published into an immutable public data structure that can help users understand the expected functionality of a device before it is purchased. Auditors and regulators can use this information as well to inform regulation of future devices. The disadvantage of this transparency is that directly connected nodes can query each other for the availability of blocks corresponding to particular chain. Because Sentinels only keep blocks for devices they are protecting, this could allow an attacker who knows the mapping between a device and a chain to learn the presence of specific devices on specific Sentinels. We see two possible mitigations: (1) Sentinels could store a copy of device-specific chains for devices they don't protect, allowing them to respond to block requests even if they do not protect the corresponding device. This privacy improvement comes at the expense of block storage. (2) Sentinels could throttle requests per source IP address and exponentially increase the response delay for every subsequent request from the same source.

6.3 Usability

Despite its complexity, SERENIoT can fully operate without any user interaction to protect the vast majority of simple IoT devices. We expect that this zero-configuration will encourage adoption even by non-expert users. However, in the event of software updates that change the behaviour of an IoT device, the system will prevent new functionality from working until the majority of Sentinels observe the same behaviour on their devices. The research community has not yet measured the speed of deployment of software updates on IoT. Lack of updating may leave early adopters without the ability to use the new features. One might argue that for security reasons, waiting for the majority of devices to upgrade is safer, but some users may want the latest features as soon as possible.

In future, we may add a manual override to allow expert users to clear the currently learned behaviour of their device and treat it as new after the update has been applied which will likely force the device onto a different chain.

The system provides usability for regular users. It does not require any user interaction to work and will protect devices once they are onboarded. For real world deployment, SERENIoT could be deployed by ISPs on consumer gateways.

7 CONCLUSION

IoT devices in smart homes are often unnecessarily overprivileged, increasing the risk of compromise and impact of attacks. This paper explored leveraging blockchain technology to assist in determining a strict specification of essential network behavior of IoT devices. We presented SERENIoT as a proof of concept network policy management and enforcement system that can operate with little to no user input. Our evaluation shows that the system is able to converge on small network security policies for many simple consumer IoT devices without requiring changes to firmware, software, or apps, and without requiring vendor buy-in. The consensus algorithm forces attackers to execute majority attacks to make changes to those policies. While implementation and deployment challenges remain, we hope that SERENIoT can be viewed as a first step toward blockchain-based network security policy enforcement systems.

ACKNOWLEDGMENTS

We thank Jeremy Clark and the members of Catallaxy for insightful discussions that helped shape the ideas in this paper. We also thank José Fernandez for his support and guidance. We'd like to thank François Labrèche, Chris Bellman and members of the security groups at Polytechnique Montréal and Carleton University for their comments and suggestions on drafts of this paper. The second author acknowledges support from the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Discovery Grant.

REFERENCES

[1] 2018. Your smart fridge could be mining Bitcoins for criminals. CBC News. https://www.cbc.ca/news/technology/bitcoin-hacking-smart-devices-1.4728222
[2] 2020. Google denies Xiaomi access over security bug. BBC. https://www.bbc.com/news/technology-50981993
[3] Mansoor Ahmed-Rengers and Kari Kostiainen. 2020. Don't Mine, Wait in Line: Fair and Efficient Blockchain Consensus with Robust Round Robin. *arXiv preprint arXiv:1804.07391v3* (2020).
[4] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security evaluation of home-based IoT deployments. In *IEEE 40th Symposium on Security and Privacy (S&P)*.
[5] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the Mirai Botnet. In *26th USENIX Security*.
[6] Adam Back. 1997. A partial hash collision based postage scheme. http://www.hashcash.org/papers/announce.txt
[7] David Barrera, Ian Molloy, and Heqing Huang. 2017. IDIoT: Securing the Internet of Things like it's 1994. *arXiv preprint arXiv:1712.03623* (2017).
[8] Christian Cachin and Marko Vukolić. 2017. Blockchain Consensus Protocols in the Wild. In *31st International Symposium on Distributed Computing*.
[9] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *IEEE 37th Symposium on Security and Privacy (S&P)*.
[10] Tomer Golomb, Yisroel Mirsky, and Yuval Elovici. 2018. CIoTA: Collaborative IoT anomaly detection via blockchain. *arXiv preprint arXiv:1803.03807* (2018).
[11] Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. 2017. Cache attacks on Intel SGX. In *10th European Workshop on Systems Security*.
[12] Sanket Goutam, William Enck, and Bradley Reaves. 2019. Hestia: Simple Least Privilege Network Policies for Smart Homes. In *12th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*.
[13] Javid Habibi, Daniele Midi, Anand Mudgerikar, and Elisa Bertino. 2017. Heimdall: Mitigating the internet of insecure things. *IEEE Internet of Things Journal* 4, 4 (2017), 968–978.
[14] Stephen Herwig, Katura Harvey, George Hughey, Richard Roberts, and Dave Levin. 2019. Measurement and Analysis of Hajime, a Peer-to-peer IoT Botnet.. In *NDSS*.
[15] Hyperledger. [n.d.]. PoET 1.0 Specification. https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/poet.html
[16] Intel. [n.d.]. Software Guard Extensions. https://software.intel.com/en-us/sgx
[17] Ari Juels and John Brainard. 1999. Client Puzzles: A Cryptographic Defense Against Connection Depletion Attacks. In *NDSS*.
[18] Simon Kenin. 2017. BrickerBot mod_plaintext Analysis. https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/brickerbot-mod_plaintext-analysis/
[19] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *Springer Advances in Cryptology (CRYPTO)*.
[20] Ayush Kumar and Teng Joon Lim. 2019. Early Detection of Mirai-Like IoT Bots in Large-Scale Networks through Sub-sampled Packet Traffic Analysis. In *Future of Information and Communication Conference*.
[21] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. 2019. All things considered: An analysis of IoT devices on home networks. In *28th USENIX Security*.
[22] Eliot Lear, Dan Romascanu, and Ralph Droms. 2019. *Manufacturer Usage Description Specification*. Technical Report 8520. Internet Engineering Task Force. https://tools.ietf.org/html/rfc8520
[23] Jon Matonis. 2014. The Bitcoin Mining Arms Race: GHash.io and the 51% Issue. https://www.coindesk.com/bitcoin-mining-detente-ghash-io-51-issue
[24] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. 2018. N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Computing* 17, 3 (2018), 12–22.
[25] Diego M Mendez Mena and Baijian Yang. 2018. Blockchain-Based Whitelisting for Consumer IoT Devices and Home Networks. In *19th Annual SIG Conference on Information Technology Education*.
[26] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. IoT Sentinel: Automated device-type identification for security enforcement IoT. In *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*.
[27] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. 2020. Plundervolt: Software-based Fault Injection Attacks against Intel SGX. In *IEEE 41st Symposium on Security and Privacy (S&P)*.
[28] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).
[29] Lily Hay Newman. 2018. An Elaborate Hack Shows How Much Damage IoT Bugs Can Do. wired.com/story/elaborate-hack-shows-damage-iot-bugs-can-do/
[30] Ray and Michael Huebler. 2019. Moving from Hacking IoT Gadgets to Breaking into One of Europe's Highest Hotel Suites. Black Hat USA.
[31] Shahid Raza, Linus Wallgren, and Thiemo Voigt. 2013. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad hoc networks* 11, 8 (2013), 2661–2674.
[32] Scott Ruoti, Ben Kaiser, Arkady Yerukhimovich, Jeremy Clark, and Robert Cunningham. 2019. SoK: Blockchain Technology and Its Potential Use Cases. *arXiv preprint arXiv:1909.12454* (2019).
[33] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE 31st Symposium on Security and Privacy (S&P)*.
[34] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *27th USENIX Security*.
[35] Nico Weichbrodt, Anil Kurmus, Peter Pietzuch, and Rüdiger Kapitza. 2016. AsyncShock: Exploiting synchronisation bugs in Intel SGX enclaves. In *21st European Symposium on Research in Computer Security*.

A APPENDICES

A.1 SERENIoT screenshots

Figure 10 shows the identification and rejection of a fork with anomalous packet signatures.

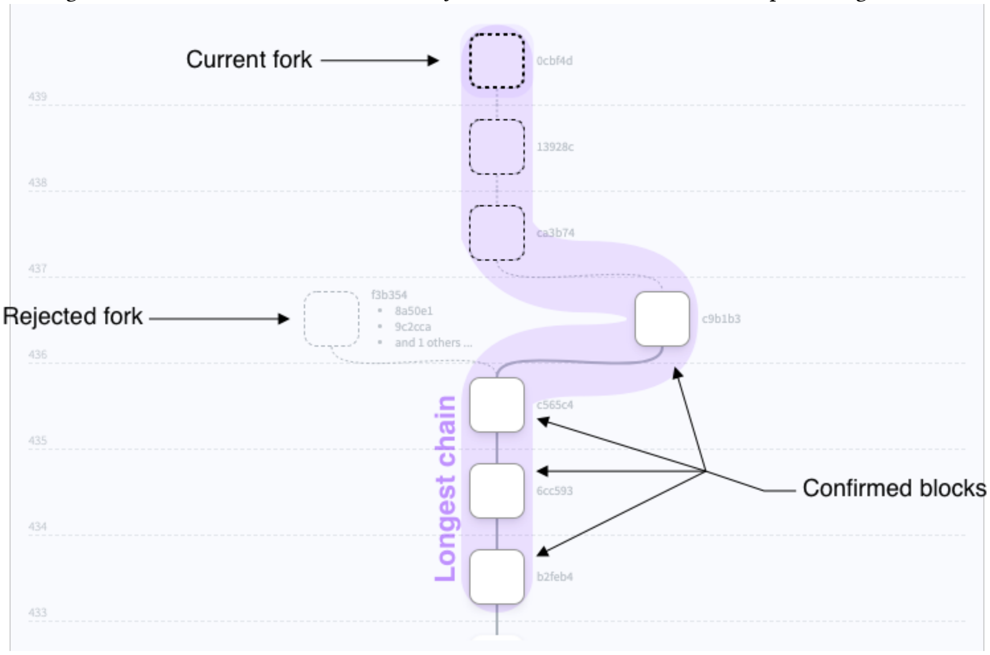

Figure 10: Rejected fork with anomalous packet signatures. The whitelist contains only packet signatures listed in the longest chain.

Figures 11, 12 and 13 show screenshots of the Web UI during a simulation with 100 Sentinels.

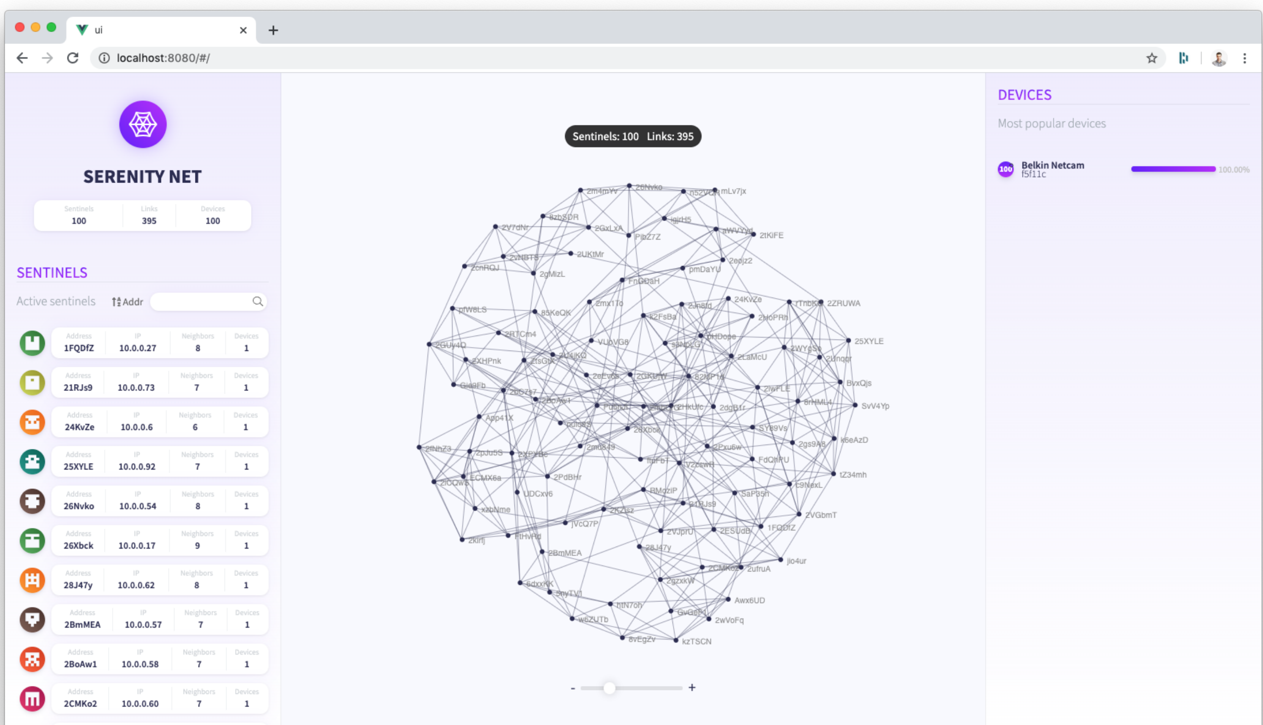

Figure 11: Screenshot of the Network view of the Web UI.

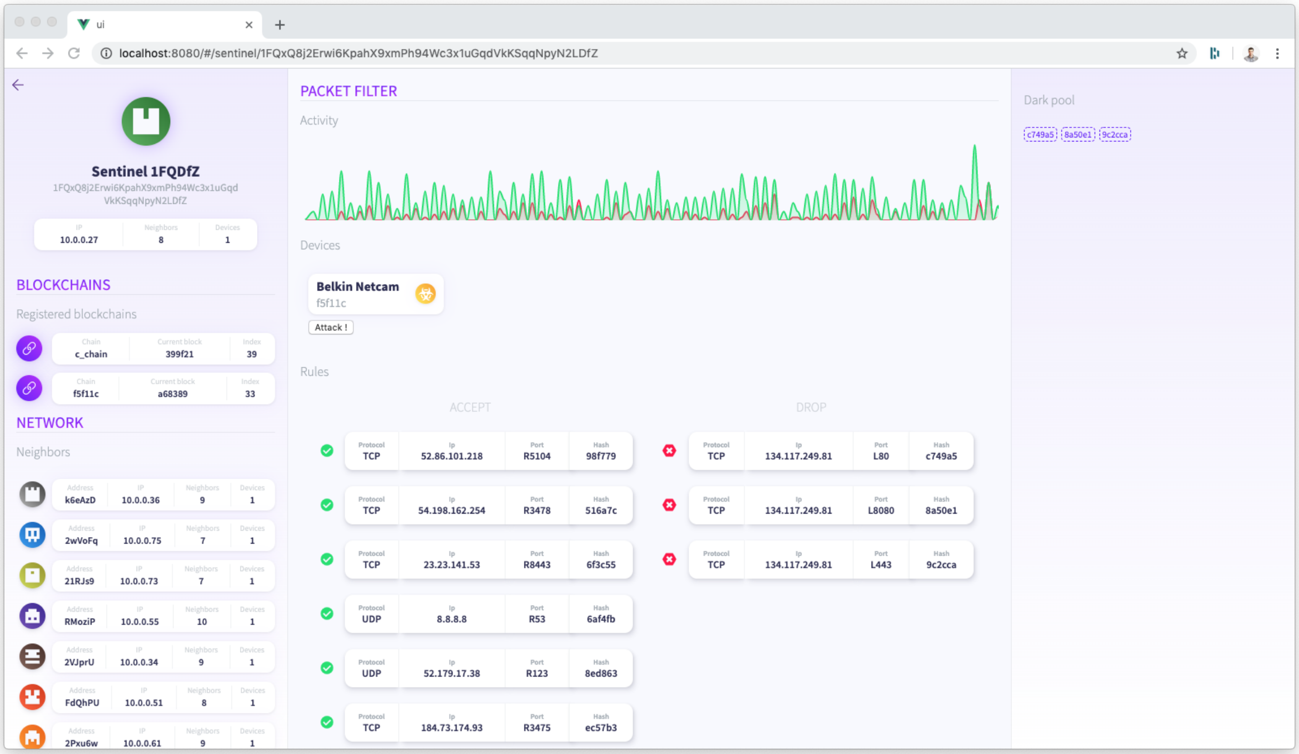

Figure 12: Screenshot of the Sentinel view of the Web UI.

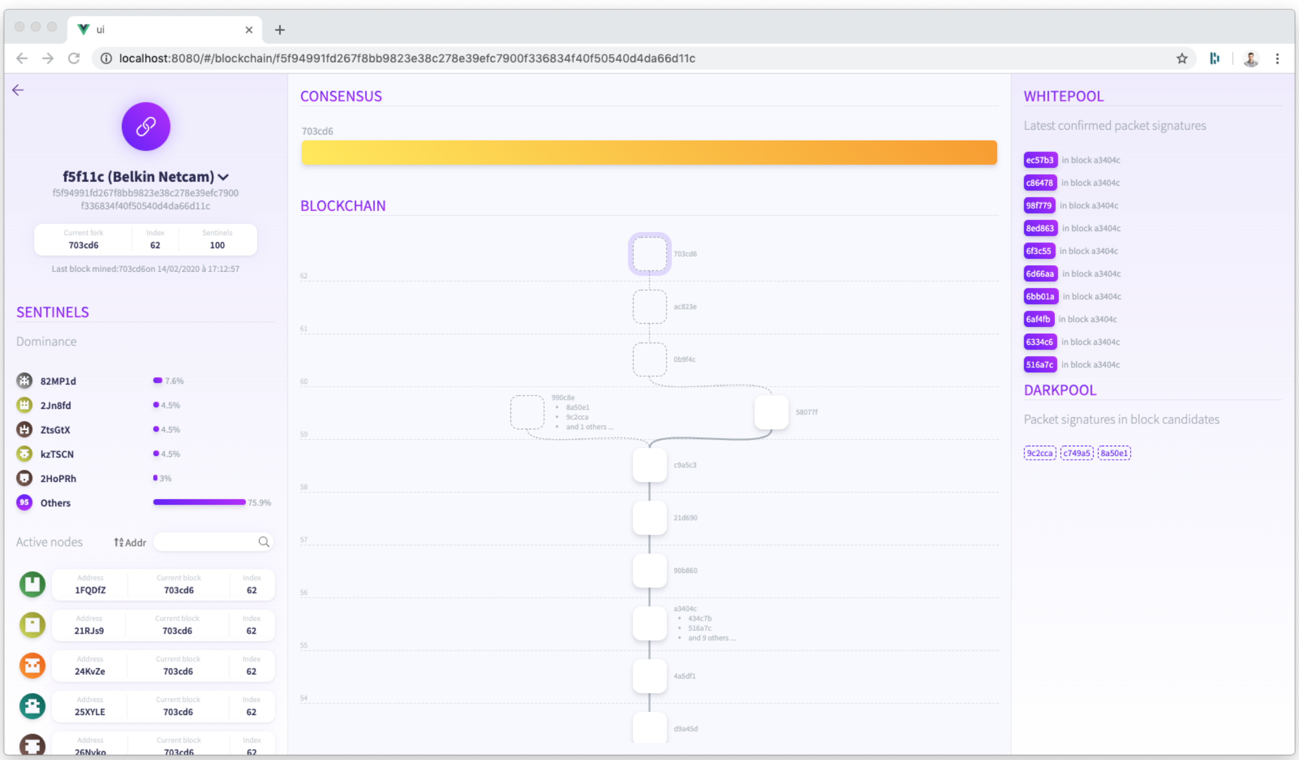

Figure 13: Screenshot of the Blockchain view of the Web UI.