Enabling Malware Remediation in Expanding Home Networks

by

James Howard Loving
B.S., Computer Criminology
B.S., International Affairs
Florida State University, 2013

Submitted to the Institute for Data, Systems, and Society and the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of
Master of Science in Technology and Policy
and
Master of Science in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2017

Author .....
Institute for Data, Systems, and Society and the Department of Electrical Engineering and Computer Science
January 20, 2017

Certified by .....
Dr. David D. Clark
Senior Research Scientist
Thesis Supervisor

Accepted by .....
W. A. Coolidge Professor Munther Dahleh
Director, Institute for Data, Systems, and Society

Accepted by .....
Professor Leslie A. Kolodziejski
Chair, Department of Electrical Engineering and Computer Science
Committee on Graduate Students

Enabling Malware Remediation in Expanding Home Networks

by

James Howard Loving

Submitted to the Institute for Data, Systems, and Society and the Department of Electrical Engineering and Computer Science
on January 20, 2017, in partial fulfillment of the
requirements for the degrees of
Master of Science in Technology and Policy
and
Master of Science in Electrical Engineering and Computer Science

Abstract

As the Internet of Things (IoT) grows, malware will increasingly threaten Internet security and stability. Many actors, from individuals installing antivirus on their personal computers to law enforcement conducting botnet takedowns, have some capability to prevent or remediate malware, but these strategies face technical and economic challenges. These challenges worsen as the IoT expands, due to the high number of IoT devices and other characteristics of the IoT.

Fortunately, Internet Service Providers (ISPs) are positioned to effectively contribute to malware remediation efforts, through the detection and notification of compromise. However, Network Address Translation (NAT) and IPv6 Privacy Extensions prevent ISPs from identifying the specific compromised device. We refer to this last-mile extension of the IP traceback problem as the residential source identification problem. As the IoT grows, the problem worsens: IoT devices are less capable of self-remediation and expected to soon outnumber traditional devices, thus imposing a significant cost on customers to triangulate and remediate an infection.

To address the residential source identification problem, I propose EDICT, an open-source software package for home routers that will enable consumers to identify a specific device, given retrospective notification of the malicious behavior, without compromising the consumer's privacy. EDICT does this by maintaining a mapping of IP flows to devices through a series of scalable Bloom filters, allowing EDICT to operate under the significant memory constraints of home routers. When a customer is informed of compromise, EDICT will query this connection log using a fuzzy check of the timestamp and source port, both provided by the ISP, iterated across a log of identified devices. EDICT will then provide the customer with user-friendly information on the infection's source, enabling remediation.

Thesis Supervisor: Dr. David D. Clark
Title: Senior Research Scientist

Acknowledgments

This leviathan would never have been tamed, but for invaluable help along the way.

To Dave - Thank you for guiding me down this path. When I changed thesis topics with less than a year remaining, you supported my decision unquestioningly. I appreciate your support and advice.

To Steve - Your presentation sparked the question that grew into this thesis, and your patience with my nearly unending late night - and early morning! - questions has been essential. You've made me a better researcher and engineer, and I'm excited to continue working with you.

To Ed, Barb, and Frank - TPP drew me to consider MIT initially, but I never expected it to become the home that it is to me. You three, from my first conversation with Ed to pilfering candy from Barb's office, are much of the reason for that. Thank you.

To Alex, Brandon, Cecilia, Leilani, Mike, and Zane - You've put up with me as a labmate, through countless questions and frustrations. Your advice has been invaluable, even when this work was only a side project I was pitching as a class paper.

To Ainsley, Mark, Parker, Macauley, Nick, Sarah, and Patricia - For your endless support in those moments I managed to be dragged away from research.

And most importantly, to my parents - Mom and dad, you've supported every goal I've ever set. Thank you for your unwavering support, even when I tell you I've applied to one school with no backup plan. I love you both.

Contents

List of Figures

Chapter 1

Introduction

In early 2014, researchers with Proofpoint, a computer security company, discovered a malicious email campaign, with more than 450,000 infected devices sending hundreds of thousands of emails daily. Though unexceptional in scope, this campaign heralded a new phenomenon: household devices from the Internet of Things (IoT) being corrupted for malicious purposes. Set-top boxes, home network-attached storage (NAS), smart appliances, and video game consoles were among the compromised "Things" [103]. This campaign causes concern for two reasons. One, spam email is not the limit of a botnet's malicious effects; other "botnets" have been used to compromise users' privacy and to effect distributed denial of service campaigns. Two, the IoT is expected to eclipse the traditional Internet of personal computers, servers, etc. [67], which increases the hazard. Summed, these threats raise a pressing policy question: How do we handle malware in growing home networks? This thesis analyzes this problem and suggests an answer, coordination between Internet Service Providers (ISPs) and their customers.

Botnets are networks of "bots," or devices that have been compromised by an adversary [16] - sometimes also referred to as "zombies" [19] or "slaves" [55]. These networks can range in size up to several millions [66]. While their possible uses are nearly limitless, they typically are used to send unwanted, bulk email ("spam"), collect personally identifiable information or sensitive financial information (e.g., credit

card numbers), act as network proxies[1] and mask adversaries' Internet traffic [79], and contribute to distributed denials of service, where a target is overwhelmed with traffic from the entire botnet. As the global economy continues to become Internet-connected, so too does it become vulnerable to these threats. In particular, the spread of broadband Internet access has aggravated the challenge DDoS campaigns pose: More devices and more Internet traffic makes a greater threat.

The Internet of Things exacerbates these threats. As more devices, ranging from home appliances to components of critical infrastructure, join the Internet, malware gains more victims, botnets gain more potential bots, and more of our information and economy is vulnerable. Yet, the IoT is too broad to address with a single policy. While the International Telecommunications Union (ITU) defines the IoT as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies" [61] and the ecosystem as a whole is expected to be predominantly commercial [52], this paper focuses exclusively on the threat posed by devices in the home network. Hosts in commercial or infrastructure settings face fundamentally different challenges, and their surrounding incentive structure differs enough that they should have superior security and thus, despite increased numbers, a lower threat.

Fortunately, Internet Service Providers are positioned to effectively contribute to malware remediation efforts, through the detection and notification of compromise [51]. Due to their position in the network, ISPs see metadata of all of their customers' traffic. Thus, they can readily detect certain behaviors indicative of compromise, such as participating in a DDoS campaign or connecting to a known botnet command and control server. After detecting this activity, the ISP can then notify the customer, allowing the customer to remediate the infection. There is significant precedent for ISPs detecting malicious traffic and notifying their customers, including ISPs supporting more than 40 million US subscribers [51].

[1] A network proxy serves as an intermediary and forwards and receives communications on behalf of a client. This is typically done to protect the client's security and privacy or to increase the network's performance.

However, technologies such as Network Address Translation (NAT) [33] and IPv6 Privacy Extensions [57] prevent ISPs from identifying the specific compromised device for their customers, which frustrates customers' abilities to remediate an infection. I refer to this last-mile extension of the IP traceback problem [70] as the residential source identification problem. Put simply, the challenge is determining the source of malicious Internet traffic, given a low fidelity identification sufficient under existing systems only for determining the home network to which the source device was connected. This problem is complicated by the realities of a modern home network: Devices may be numerous (dozens or hundreds), including IoT devices lacking traditional capabilities. Thus, notified customers will be unable to effectively remediate the infection, as their efforts will be spread across their entire network of potentially-infected devices.

To address the residential source identification problem, this thesis proposes EDICT (Enabling Device-level Identification of Compromised Things), an open-source software package for home routers that will enable consumers to identify a specific device, given retrospective notification of the malicious behavior, without compromising the consumer's privacy. EDICT does this by maintaining a mapping of IP flows to devices. When a device connects to the router, EDICT will identify the device through an open-source WiFi fingerprinting library [30]. EDICT then logs all flows in a series of scalable Bloom filters [8], allowing EDICT to operate under the significant memory constraints of home routers. In order to preserve users' privacy even in the face of router compromise, EDICT does not log information regarding the destination of traffic, ensuring that an adversary cannot gain the full details of users' Internet browsing history. Furthermore, this architecture is extensible to complicated network topologies, such as a home network with a router and multiple Wireless Access Points (WAPs). When a customer is informed of compromise, EDICT will query the Bloom filters using a fuzzy check of the timestamp and source port or source IP address (depending on what version of the Internet Protocol is in use), both provided by the ISP, iterated across all devices previously identified on the network. EDICT will then provide the customer with user-friendly information on the infected device, enabling

remediation.

This research has been motivated, in part, by comments made at the 2015 MIT Communications Futures Project (CFP) Plenary Session [78] and the Measurement and Analysis for Protocols Research Group (MAPRG) at the Internet Engineering Task Force (IETF) meeting 96 [21]. The questions and comments posed there form, when combined, the essence of this problem space: ISPs are capable of detecting malicious traffic and notifying their customers, but individual devices, particularly IoT devices, can be frustratingly difficult to identify.

The remainder of this thesis is organized into five chapters. In chapter 2, I introduce an analytical framework for malware remediation approaches and use it to discuss the malware remediation ecosystem for traditional (non-IoT) devices. Chapter 3 discusses how the Internet of Things and the characteristics of IoT devices stymies traditional remediation approaches. In chapter 4, I detail EDICT's system architecture and analyze several technical and policy considerations for its design. Chapter 5 analyzes EDICT's overall performance and security against particular classes of malware. Finally, in chapter 6, I discuss several conclusions and offer suggestions for future work on EDICT and tangential research questions evoked by this work.

The primary contributions of this thesis are the following:

- Development of EDICT, an open source software package that will enable home Internet users to identify compromised devices given outside notice of compromise, while preserving their privacy. EDICT has only minimal memory requirements (<20MB) and has no significant impact on CPU utilization or network throughput.

- Holistic analysis of malware response options for home Internet of Things devices.

- Identification of the residential source identification problem as a unique subset of the challenges in remediating home malware infections. In contrast, the literature has previously either conflated the challenges of home malware remediation or focused exclusively on the problems of detection and remediation.

Chapter 2

Traditional Malware Remediation

Malware is a multi-faceted problem and cannot be fully addressed by any single actor. In this chapter, I outline the several layers at which malware might be addressed and analyze the strengths and weaknesses of each layer, acting independently. These layers are shown in figure 2-1. First, there is the source host (A): This is the device that has been infected and is the source of the malicious traffic (spam, DDoS traffic, etc). This malicious traffic is then routed through the source network (B), which for the purposes of this discussion is a consumer, non-commercial private network (as opposed to an educational, corporate, or government network). Next, the traffic travels to the source network's ISP and, through the interconnection of ISPs, through the Internet (C). The "cloud" image is, thus, a gross simplification of the Internet, but as subsection C discusses, many possible solutions in this layer require significant coordination. Depending on the type of traffic, it can then reach the botnet's command and control (C2) server (E via the C2 network, D) or the destination host (G via F). Typically, malware will first communicate with the C2 server and receive a "hitlist," which indicates which destination hosts to victimize [55].

2.1 Source Host Solutions

Preventing and remediating malware at the infected host faces significant technical and economic challenges. This is unfortunate: If malware is successfully prevented or

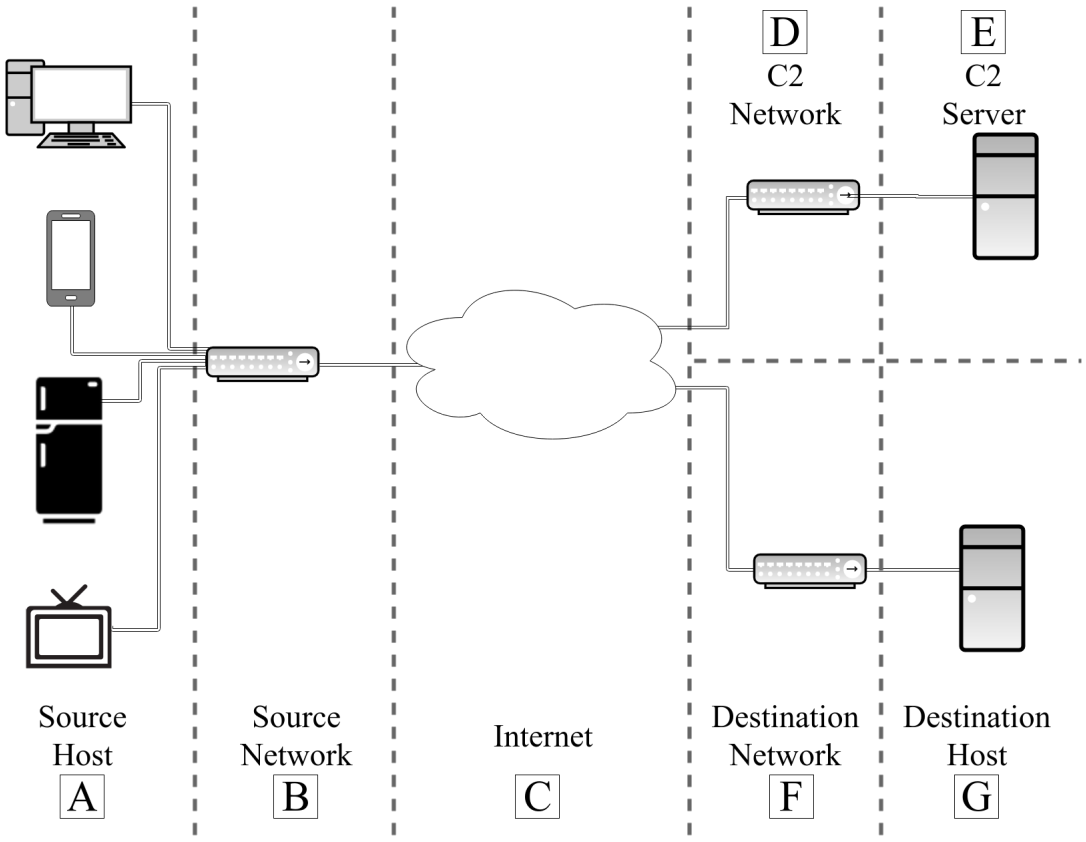


Figure 2-1: Layers of an Anti-malware Campaign. Malware can be addressed at many different control points. This framework delineates the various possibilities by following the traffic of a compromised source host (A) through to either the botnet's command and control infrastructure (E) or the victim of the malicious traffic (G).

remediated here, there is no need for secondary intervention - whereas other strategies could still result in harm to the user, such as ransomware[1] locking away the user's data. Moreover, securing a host has other benefits. Secure hosts are less vulnerable to other classes of malware, such as ransomware, which will prevent user access to a machine and demand payment to return access. Although some source host solutions can be, themselves, sources of insecurity [81], this strategy remains highly effective and an extremely common approach.

Preventing malware has two components: developing a secure system and preventing malicious changes to that system. In developing a secure system, the goal is

[1]Ransomware is a class of malware that demands ransom, typically a payment in the form of Bitcoin or some other semi-anonymous payment system, after removing the victim's ability to access data, such as by encrypting the hard drive with a random key that will be sent to the user upon receipt of payment.

typically to hamper entire classes of attacks [10, 102, 92, 48]; in the security parlance, this will greatly decrease the "attack surface" [4], which is a more efficient approach than addressing vulnerabilities piecemeal. While this strategy has significantly increased security, the continuing rates of malware infestation [87] show that hoping for secure development, on its own, is insufficient. This is, in many ways, an implementation problem: Security may be discussed, even outright required, but it is often done inadequately. Short of this goal, system developers may prevent individual vulnerabilities, typically by discovering the vulnerability, developing a "patch" for the system, and then distributing that patch to the system's users [76]. Unfortunately, there can be significant delays between a vulnerability being discovered and a patch being distributed, users rarely keep their systems fully updated [25], and even security-conscious users may be wary of updates, given a desire for stable systems [66]. This shifts the effective responsibility for system security to other technologies or actors, despite many problems being solvable by simply keeping a system up-to-date.

Second, the system must be protected from malicious changes. This requirement assumes an imperfect system - that is, even a secure development process results in an exploitable system. While enterprise scenarios can entail extensive measures such as configuration management processes, home users typically eschew these in favor of an antivirus program, as these processes are costly and unwieldy. Home users even disfavor privilege management systems, such as Windows User Account Control, that prevent untrusted software from executing with the elevated privileges required for most malicious activity, as they present an unwanted disruption to users' workflow [54]. Thus, the responsibility for preventing malicious activity falls nearly entirely onto antivirus suites. These suites can operate proactively, by scanning files for infection as they are downloaded or executed, or reactively, such as a user-directed scan of the entire system. In either mode, the targeted file is scanned and compared against a list of the signatures[2] of known bad files; more advanced suites may also detect malicious activity through heuristic analysis based on rules of what "good" and

[2]In this context, a signature refers to a cryptographic hash that uniquely identifies a file. These hashes are calculated from strong, one-way hash functions that enable complex inputs to be identified by a fixed-length, pseudorandom output.

"bad" behavior entails [34].

Unfortunately, antivirus has technical and economic challenges. First, malware can easily avoid the traditional, signature-based detection schemes, and heuristic-based schemes are not developed enough to provide adequate protection [73]. This results from, principally, the efforts of malware engineers: Malware that is easily detected is not good malware, so a good malware engineer will take significant efforts to stymie detection. Typically, malware engineers will utilize specialized programs, including "packers" that introduce an element of randomness to their code prior to it being distributed, thus preventing easy signature matching [37]. Additionally, newly-developed malware can be checked against common antivirus suites, allowing the developer to verify that it will be undetected. Consequently, antivirus suites, which are often the only implemented security mechanism, are insufficient. Second, antivirus suites impose costs on their users. They have computational requirements, which while not a significant impediment to modern devices, can challenge older systems [7] - and can, even in newer systems, lead uneducated users to forego antivirus for performance considerations. More directly, antivirus typically operates on a subscription model, where the user must pay annually for access to the service [53]; while free alternatives abound, uneducated users may not be aware of their availability, especially given that subscription antivirus is often pre-installed on new machines. Antivirus suites also require regular updates, and this hassle can discourage users [25]. These challenges compound to lower the level of antivirus use below the level optimal for the security of the Internet as a whole.

Education campaigns can help these problems, from the security of software development [25] to antivirus use [79]. Typically, these campaigns encompass a wide range of "cyber hygiene" issues, including antivirus usage and best practices, such as verifying email attachments prior to downloading. The "Stop.Think.Connect" campaign, currently being conducted by the US Department of Homeland Security (DHS), has "keep a clean machine" as its first category of security advice: Users are instructed to "keep security software current" and "automate software updates" [9]. This emphasizes the unfortunate reality of host security: The more effective security measures

are, often, those that are less used.

Even with a broad campaign for public awareness, malware cannot be completedly handled by addressing infected hosts. This is a reflection of the negative externality Internet users generate in their security choices. Many of the costs of malware are not borne by those whose systems are infected. While a bot owner has to face certain consequences, such as the release of sensitive financial information, they will not have to deal with the flooded inbox that results from a spam email campaign or the crippling traffic from a DDoS. Thus, users will rationally choose a level of security that, not reflecting the total cost of such poor security, is below the optimal level for society. While increased public awareness of cybersecurity harms - through public awareness campaigns or media reports of data breaches - can help this balance, users still provide an inadequate level of security [25]. Compound this with the imperfection of existing security measures, and the resulting security provided at the level of infected hosts is clearly inadequate.

2.2 Source Network Solutions

Similarly, malware cannot be adequately handled at the level of the source network, but home networks typically lack any security measures at this layer. Such networks typically only contain a router, beyond the individual connected devices. While some home routers do contain limited security capabilities, such as preventing an infected host from scanning for open ports, generally routers are not capable of effectively remediating a malware infection. Broadly, network devices have several approaches to handling malware. They could prevent connected devices from ever downloading malware, such as by performing antivirus scans on all data flowing across the network; this is infeasible for various reasons, including encrypted communications and routers lacking the necessary processing power. Routers could also prevent the malware from "phoning home" to a C2 server or conducting malicious activity against an external destination (as they can do little to stop malware's activities against the infected host). Unfortunately, this approach requires the network device be able to

differentiate between legitimate and illegitimate traffic, in real time; this is often beyond entry-level consumer devices with limited processing power and memory [16]. Moreover, even devices that perform some analysis of traffic typically will only block a subset of malicious traffic. Blocked traffic is typically activity where the identifiers are constant, such as a port scan, compared to a bot communicating with a C2 server.

Fortunately, there are classes of devices designed to handle the complexities of detecting illegitimate traffic: Intrusion Detection Systems (IDSs), which detect malicious activity via generally the same mechanisms as antivirus suites, and Intrusion Prevention Systems (IPSs), which take that approach further and block, or otherwise handle, the detected traffic [35]. Such devices are common on enterprise networks, such as large corporations. Yet, they are rarely used in consumer networks for two reasons. First, they are expensive - significantly more than the average price of a wireless router. Second, they require effort to administer, especially given that an incorrect action by an IPS may block legitimate communications. As previously noted, users have a low willingness to pay for security, due to the misaligned incentives generated by malware, and are thus unlikely to pay for high-cost security choices such as an IDS.

There are techniques that do not require specialized hardware, however. Routers are generally capable of limiting the external connectivity of a connected device. This functionality could place a device into a "quarantine," where it could perhaps only talk to other devices on the network, but not the Internet, or no devices at all; less extremely, it could enact rate limiting, where the connectivity of the targeted device is limited, but not disabled - lowering its usefulness in a spam or DDoS campaign. These techniques are not ordinarily seen on a consumer device, but that is due to a surmountable software lack, not an insurmountable hardware one. Used effectively, quarantining a device can forestall many of the negative consequences of that device's compromise. While the user loses the Internet connectivity of the infected device, their privacy remains uncompromised, and adversaries lose access to the bot.

2.3 Internet Solutions

At the Internet level, there are many mechanisms for controlling malware. This analysis centers on Internet Service Providers (ISPs). While there are other Internet actors - such as Regional Internet Registries (RIRs) and Certificate Authorities (CAs) - that might have specialized roles to play in anti-malware campaigns, these approaches will be covered in the discussion of malware command and control, as these levers impact control of the malware network, not transmission or other facets associated with the greater Internet. Still, ISPs have several capabilities. Like devices at the source network level, they can detect and block malicious activity - and do so efficiently. This efficiency springs from economies of scale, as ISPs handle large volumes of traffic relative to the upfront costs associated with the necessary security hardware.

First, the ISP must detect the malicious activity amongst the body of legitimate traffic. This typically is done by comparing activity against known signatures of malicious activity. These signatures can be the destination of traffic, in the case of a known C2 infrastructure, or protocol information, for malware strains that have predictable communications patterns. As an example of both forms, consider the Conficker worm. It is possible to predict, in advance, what C2 servers the Conficker horde will attempt to contact, and Conficker always scans on port 53/UDP (used for the Domain Name System, DNS), 80/TCP (used for insecure Hyper-Text Transfer Protocol, HTTP, requests), and 445/TCP (Server Message Block, SMB, a Microsoft Windows network communication protocol) [66]. This form of analysis is typically known as metadata analysis, in reference to the malware communications' metadata providing the signature for detection.

Without an identifiable signature, detecting malicious activity is naturally more difficult. This is the goal of IP traceback - retroactively determining the origin of a malicious packet stream. There are several proposed schemes, each hoping to mitigate DDoS campaigns [70, 101, 18, 100]. Archetypically, a scheme works as follows: Each ISP will mark packets that traverse its network in some way. Then, the victim of an attack can contact its ISP, who will determine which other ISP generated the

malicious traffic; the victim's ISP contacts the second ISP, who iterates this process, until the source ISP is able to determine the malicious customer. Alternatively, each ISP may log packets that it sees; the request then becomes an iteration of log queries. While this approach imposes a higher cost on the ISP, it allows for the tracing of a single malicious packet, whereas marking methods may require a significant body of traffic be utilized to determine the origin [75].

Detection is challenged by IP source address spoofing, where an adversary forges the source IP address on a packet. In a DDoS campaign, this is typically done to protect the botnet infrastructure; bots will forge source addresses, so other parties cannot easily identify the bots and thus combat the botnet. [86]. IP traceback schemes generally prevent effective spoofing, in that the traceback system can determine, at each iteration, the source network. Thus, an adversary could not spoof a packet to appear from a different ISP or even network; however, a skilled adversary could spoof a neighboring address, which is important for non-residential settings such as school or corporate networks with large numbers of connected systems on one network [11].

Once the ISP has detected malicious activity, it has limited options. It can block the known malicious traffic, block all traffic from the malicious host, or rate limit the infected host - the same options possible at the source network. This filtering is possible at two points, the entrance to the ISP's network - ingress filtering - and upon exiting the ISP's network - egress filtering. While the latter is common on enterprise networks (to prevent security leaks), it is clearly inefficient for an ISP, who would still pay the cost of pushing the malicious traffic in its own network. Therefore, ingress filtering is the predominant choice [64]. It also may be more effective: A model simulating the Internet found that ingress filtering could be more than two times as effective at preventing "wicked" traffic [36]. One filtering scheme, Pushback, takes this logic further and allows for an ISP to enact ingress filtering and then send filtering requests to the source networks. On the whole, this allows an optimal outcome, the malicious traffic being blocked at the first ISP - but this requires significant coordination between ISPs and near universal adoption of Pushback [40], due to strong network effects.

Unfortunately, these mechanisms are significantly limited by modern Internet architecture. IP addresses can come in two forms, version 4 (IPv4) and version 6 (IPv6). IPv4 is older and depreciated, but the vast majority of addresses are IPv4. The total possible address space for IPv4 is highly limited, and the entirety has been allocated [44]. This address exhaustion predicated the development of intermediate measures until IPv6 could be implemented; one of these is Network Address Translation (NAT), which allows routers to map a series of internal IP addresses to an external IP address, thus allowing for many devices to share a single IP address allocation. Additionally, this system provides privacy protection for those translated addresses; an external observer cannot determine which device is the source of traffic [33]. IPv6 systems can gain equivalent privacy protection through the use of IPv6 Privacy Extensions, which randomizes the source address such that an outside observer is unable to easily connect two such random addresses from the same system [57]. These systems prevent ISPs from identifying the source devices for traffic, including through malicious traffic detection schemes such as IP traceback. Consequently, any action taken by ISPs must be at the customer level: An entire customer's network will be quarantined, not merely the infected device.

Despite these difficulties with residential source identification, ISPs have significant capabilities to combat malware for home Internet users. Unlike at the customer level, an ISP can leverage economies of scale for detection systems. They face limited misalignment of incentives regarding their security choices, due to the informal interconnection between network operators providing a coercive mechanism for positive security choices. An ISP's costs are tied to their relationships, and transit agreements, with other ISPs, which are in turn dependant on the security of that ISP, so an ISP is financially incentivized towards stronger security. Overall, the only drawback to remediating malware with ISPs is the impermanence of the solutions - quarantines do not remove the disease. Due to societal concerns over privacy, this inability is unlikely to change. The US legal tradition has long held that homes are granted significant privacy protections, in excess of the protections afforded other locations. These protections can apply to Internet communications originating in the home:

Outside observers, including ISPs and law enforcement agencies, should not be able to readily get device-level identification of the source of traffic. Thus, ISPs are unable to pursue a solution targeted at the individual infected host; they are limited to network-level, customer-level, actions. This is unfortunate for ISPs' customers, who may find themselves entirely disconnected from the Internet for the corruption of a single device on their network.

2.4 C2 Network Solutions

For certain classes of malware, affecting the command and control networks (C2) networks may be a potent approach. This is most effective against malware strains that require control, such as bots; malware strains able to effect harms without phoning home, such as ransomware, are much less vulnerable. As a further complication, many operations against C2 networks require participation from actors not traditionally involved in the malware problem, such as the Internet registries that maintain the Domain Name System (DNS) that translates from human-readable web addresses to IP addresses. As these actors can face organizational resistance in addressing areas, such as security, outside of their typical purview, campaigns against C2 networks are typically limited only to politically-charged issues with the capital to overwhelm these challenges.

The goal of these strategies is typically to either remove the malware's ability to phone home successfully - to "sinkhole" the botnet - or to intercept malware instances' requests for commands and reply with forged commands - to "hijack" the botnet [79]. While the latter is obviously preferred, modern malware can employ encrypted and authenticated communications that frustrate the forgery of botnet commands [66], but malware engineers do not uniformly utilize these measures [16, 79]. In the presence of authenticated communications, which preclude hijacking, sinkholing is the only option. This can take several forms, depending on how the targeted strain of malware communicates. There are two common variations of communication, master-slave and peer-to-peer. In a master-slave infrastructure, each bot phones home to a C2 server

(or one of a series of C2 servers); this is the stereotypical approach. In a peer-to-peer setup, this single point of failure is removed: Each bot requests commands from neighboring bots and retransmits commands to all other neighboring bots [37].

However, hijacking a botnet may be illegal. The Computer Fraud and Abuse Act (CFAA) outlaws the acquisition of information from a "protected computer."[3] Modern malware often includes dual functionalities to steal private information and create bots, and the malware often transmits this information to its C2 infrastructure. Thus, depending on the exact steps taken, a botnet hijacker may find themselves in violation of the CFAA, which does not contain a security or research exemption [1]. This could limit botnet hijacking to legal authorities, who are unwilling to utilize this approach - it is technically difficult and does not lead to convictions. Yet, academics have successfully conducted research projects that centered on hijacking botnets [79]. Without legal precedent or a clear security research exemption in the legislation, this uncertainty will remain, and this uncertainty may prevent botnet hijacking campaigns. However, given the second-order effects of such campaigns, legal protections are necessary: Botnet hijacking, while perhaps efficient, can be a form of vigilantism. Non-government campaigns lack the bureaucratic protections necessary for guarding against such effects.

To disrupt the master-slave relationship, an attacker must prevent the bots from successfully contacting the C2 server. Typically, malware calls back to its master not by hard-coded IP address, but by contacting a domain name, which is then resolved through the DNS into the C2 IP address. This process can be disrupted: Prevent (or, more rarely, remove) the registration of that domain name, and the botnet will never be able to phone home [79]. Unfortunately, malware engineers have developed "domain flux" techniques, where the malware periodically generates a list of domain names and iterates down that list; the first authenticated response is treated as the

[3] A protected computer is one "exclusively for the use of a financial institution or the United States Government, or any computer, when the conduct constituting the offense affects the computer's use by or for the financial institution or the Government; or which is used in or affecting interstate or foreign commerce or communication, including a computer located outside the United States that is used in a manner that affects interstate or foreign commerce or communication of the United States..." Practically, this means any Internet-enabled computer.

master. Modern malware can generate tens of thousands of domains per day, forcing malware responders into a lopsided game of catch-up [66]. Malware responders will have to disable the entire list of possible domains, and the work required to do this vastly exceeds the work required by malware engineers to implement domain flux. Therefore, DNS-based responses may be rendered completely infeasible.

Internet Registries, who maintain the DNS, are well positioned to assist in such efforts, but are generally neither organized nor incentivized to effectively assist. Their cooperation is essential in fighting malware with large domain flux capabilities, such as Conficker [13]. Unfortunately, "they often lack the resources, incentives, or culture to deal with the security issues associated with their roles" [79]. Thus, this approach is infeasible in the long term without a serious reorganization of the Internet registries, who historically have not focused on security operations. It has only proven effective against threats grave enough to merit broad attention, such as the Conficker worm. For less dire threats, the resource scarcity is compounded by a lack of communications channels - it can even be difficult to identify a point of contact at a registry for security takedown requests [80].

Peer-to-peer C2 infrastructures eschew the domain approach and thus demand a different response. While some malware strains have been vulnerable to sophisticated, technical attacks, a more common theme is the Sybil attack [24]. In this approach, malware responders simply introduce a massive number of pseudo-bots who insert themselves into legitimate positions in the growing botnet, but can then be utilized to limit the spread of communications. While effective, this approach has large resource requirements that prevent it from widespread usage - the pseudo-bots must be real (or virtualized[4]) computers, so inserting enough bots to affect the botnet's functionality is expensive.

Instead of focusing on the technical aspects of the C2 infrastructure, alternatively the campaign can focus on the economic infrastructure. For malware that has been

[4]Virtualized computers are computers whose operation has been "virtualized," run on another system. This approach allows for one physical computer to run multiple virtual machines. In this case, it allows the pseudo-bots to be more resource efficient, but this approach is still generally infeasible.

highly monetized, this approach can be incapacitating. A recent campaign against "stress test" services, pseudo-legal DDoS services that an adversary can purchase and use for their own aims, coordinated with PayPal to remove the payment accounts used by the DDoS peddlers, resulting in an 80% reduction in business. Unfortunately, this approach is only possible against the strains of malware that are highly commercialized through centralized services such as PayPal; modern, monetized malware that chooses a distributed approach such as Bitcoin are not vulnerable, since there is no centralized authority that can be coerced into disabling the payment infrastructure [45]. Hence, while payment interdiction strategies may see limited effectiveness currently, this will decline as cryptocurrencies rise in usage.

2.5 C2 Host Solutions

In many high profile malware cases, law enforcement officials may go directly after the C2 host and its operators. Such campaigns are typically directed at financial malware, the strains that target personal financial information such as credit card numbers and bank account details, including the 2014 Gameover Zeus takedown by the Federal Bureau of Investigation (FBI) [28]. These approaches are greatly complicated by jurisdictional issues. Malware authors often take deliberate steps to limit their creations' effects on the authors' home country to limit their home law enforcement agency's incentives to intervene [66]. Regrettably, this strategy can be highly effective, as many legal jurisdictions require that a legitimate harm to the jurisdiction's citizens be demonstrated before law enforcement action is taken. This can necessitate other approaches, such as a victimized country coordinating with the botnet herders' country through a Mutual Legal Assistance Treaty (MLAT), which are effectively extradition agreements for evidence. Unfortunately, these bilateral agreements are slow and cumbersome, and the staffs responsible for coordinating countries' activities in support of them are commonly overworked [47]. Thus, it is unlikely that a malware case will get the necessary support, when such cases compete with murders and other serious crimes. This limits law enforcement's effectiveness against malware,

as most strains are broadly transnational.

2.6 Destination Network Solutions

Malware can be mitigated by bolstering the destination network, but these approaches do not decrease many of the ecosystem-wide costs of malware. To achieve this defense, a network can be engineered to be secure and resilient. These are subtly different goals: Security prevents attacks, but a good network is engineered under the assumption that security will be imperfect, necessitating a design that allows the network to be resilient in the face of attacks. At the network level, security typically comes from a firewall [95], IDS [96], IPS [97], or some combination thereof. Load balancers [99] can increase the network's resilience, especially in the face of DDoS attempts. While these devices are infeasible in home networks for reasons discussed in section B, destination networks do not face the same incentive structure. The destinations of malware traffic are typically enterprise networks - corporations, schools, governments - that have both a much higher security budget and a stronger incentive to spend that money wisely. However, this approach is still generally ineffective for the victims of concentrated efforts. While large corporations such as Google or Facebook might be able to successfully withstand a large campaign, overall 65% of victim networks are "severely affected" [94]. Moreover, this strategy still leads to an sub-optimal result for the Internet ecosystem. Although a blocked attack may have limited visible effects, consumers and ISPs still bear the costs of their machines' corruption; consequently, it is more efficient to remediate at a lower layer.

2.7 Destination Host Solutions

Finally, individual hosts can be hardened against the effects of malware campaigns. This can be done similarly to the network-based approaches: prevent the traffic from having an impact, such as through host-based firewalls, and limit any effects through policies such as resource allocation, which limits the resources any connection can

utilize, so illegitimate traffic will hopefully have less of an impact on legitimate users' access [17]. For web servers, a common approach is the utilization of CAPTCHA systems [91] that prevent automated access to a resource. Alternatively, hosts can be hardened through protocol innovation, such that their entire interaction with all users is secured and illegitimate users have less potential impact. This can eliminate entire classes of attacks, such as the Transmission Control Protocol (TCP) sync (SYN) flooding technique - previously, one of the most popular and dangerous denial of service methods - being mitigated by an update to the Protocol [26]. As with network-based approaches, these methods provide significant benefits to potential victims, but their systemic use in favor of other approaches would not result in sufficient security for the Internet. Destination solutions are a worthwhile endeavor, and their incentive structure encourages their use, but they are not a total solution.

Chapter 3

Remediating Malware in the Internet of Things

The Internet of Things will challenge conventional malware remediation. While there have been a few known malware campaigns wielding IoT devices, such as the 2014 spam email campaign from compromised set-top boxes [103], a 2014 DDoS campaign with home routers, a 2015 DDoS campaign of home security cameras [29], and the recent record-setting DDoS campaigns against the Krebs on Security blog [14] and Dyn, a DNS provider [49]. the sample is far too small to draw adequate conclusions about the future of IoT compromise. Thus, this analysis draws on the fundamental differences between IoT and traditional devices, including how the IoT market and ecosystem differ from their traditional equivalents, and what effects these features may have on networks and the Internet.

3.1 Remediating an IoT Device

In several ways, an IoT device is critically different from a traditional device such as a personal computer or smartphone. First, these devices generally have low computational power. This results from differing requirements for low cost - necessitating less expensive and less powerful components - and low electrical power requirements (for battery-powered devices). While this might be expected to decrease IoT devices'

use to malware engineers, most strains' chief goals are information - private financial details, etc. - or Internet bandwidth with which to send spam or malicious traffic [19]. IoT devices have access to bandwidth and personal information on par with traditional devices. Thus, outside of specific strains that have high computational requirements, such as Bitcoin mining malware,[1] IoT devices will be fully capable draftees into any botnet. Accordingly, the threat of IoT botnets is in line with the size of the IoT, not diminished by the functionality of the individual devices.

IoT devices' equivalent capability frustrates malware policy. For a policy to be effective, it must not focus exclusively on the "big" devices - personal computers, smartphones, and full-feature IoT devices that may be running a traditional operating system, such as an Android-based smart television; an effective policy must also cover the numerous low-power devices. This directly imposes a cost on IoT users, who must handle the malicious behavior of many more devices - it is insufficient to simply run antivirus on the "big" devices. Moreover, this is more than a technical challenge. As previously mentioned, users have demonstrated a low willingness to pay for security, and the increased scope of their security responsibilities is likely to worsen this result. Users who were unwilling to pay the costs associated with adequate security for a handful of devices are less likely to provide adequate security for dozens. Therefore, IoT devices may be able to disproportionately contribute to the threats posed by botnets, compared to traditional hosts - equivalent capability with lower obstacles, as perceived by adversaries.

Additionally, IoT devices' low power requirements can frustrate antivirus response, as antivirus suites have a nontrivial computational requirement which may not be met by a low computational power device. Second, these devices are generally intended to be single-purpose. Many existing IoT devices are developed on top of highly-capable operating systems, typically Unix/Linux, but the development might focus exclusively on the aspects that are necessary for the device's advertised functionality. This can leave glaring security gaps elsewhere in the OS, as developers fail to address security

[1]Bitcoin is a digital currency that is created through "mining," the completion of computationally-complex tasks. Thus, a botnet herder can task the botnet with mining, thus earning revenue for the botnet herder. For an example, see [65].

concerns of aspects of the system they are not using in their device. This, in turn, imposes an unexpected cost on users, who often assume that security is the default state for a product [25].

The IoT ecosystem may also be heterogeneous, relative to the ecosystem of traditional devices, which would further confound an antivirus approach. The current Internet of Things is predominantly based on the Unix and Linux operating systems [46]. However, the future architecture is uncertain, and heterogeneity would complicate antivirus significantly. An antivirus suite has to be specifically engineered for the operating system that is being protected, so a wide variety of operating systems forces a correspondingly-wide variety of antivirus suites - which are expensive to develop and keep updated. Worse, the IoT need not necessarily encompass many operating systems: Many different versions of the same OS (from customization, obsolescence, etc.) foment the same challenges.

To further complicate this space, many of the negative effects under analysis might not result from an IoT device being infected with malware, but instead might be the simple compromise of that device's legitimate functionality. For example, a refrigerator might be capable of emailing its owner when products, such as milk, expire. Given that many IoT devices are insecure - no passwords, weak passwords, etc. - such a refrigerator could be suborned into a spam email campaign with a simple login script instead of malware affecting the operating system (OS) [69]. One of the first famous examples of IoT compromise, where baby monitor cameras were listed on the Internet and able to be remotely viewed, was caused by the absence of strong passwords combined with legitimate remote access functionality, not the successful installation of malware [50]. This same tactic exists on traditional devices, certainly, but IoT devices have specific usability concerns - such as typing passwords with a remote, controller, or some other device that's not a traditional, user-friendly keyboard - that prevent some measures of stronger security.

Worse, IoT device users may be incapable of even detecting malware. While traditional malware typically exhibits symptoms such as pop-ups, website redirection, etc., IoT malware tends to target either users' privacy or seek the compromised device's

use as a bot. Neither of these effects may be detectable by users, and malware remediation first requires detection. IoT ransomware would likely prove an exception, but this type of malware has yet to reach the IoT. Moreover, even when examining tech-savvy users, the difference between traditional and IoT devices is stark. These users may be comfortable examining files and processes on a traditional device, but IoT devices mask, or outright remove, any capability for these manual forms of malware detection.

Some entrepreneurs have sought to capitalize on these security concerns. Typically, however, such services are for critical infrastructure and other commercial applications of IoT devices [93]. In the home, there are a few solutions, typically IoT-specific intrusion detection systems that are linked into the home router, but they have extremely limited adoption so far [23]. Residential IoT owners do not face nearly the same incentive structure as commercial owners. Commercial networks are subject to increased liability and visibility, which increases the incentive to implement security. Additionally, certain classes of these networks, such as those involved in critical infrastructure, receive support from the government. Thus, residential devices are unlikely to be similarly protected. While compromising consumers' IoT devices will not have the same catastrophic consequences, an optimal Internet still requires consumers' devices be protected above what the laissez-faire incentive structure would provide.

Unfortunately, the growth of the Internet of Things is not likely to answer these challenges. This inability is not exclusively technical. While there are technical challenges, such as the potentially heterogeneous ecosystem and interconnection requirements, none of these preclude the development of an IoT device with security equivalent, or even superior, to traditional devices. Yet, policy realities can impose themselves and entirely thwart this goal. IoT developers and manufacturers will be incentivized to maintain the status quo: Security is expensive.

Government action may be possible, but considering its challenges in regulating the security of critical infrastructure [5], government is unlikely to effectively intercede to secure device development. More feasibly, IoT services may be regulated for

privacy and related concerns: For example, a smart television that responds to voice commands may be regulated such that it minimizes the voice samples it stores, encrypts those it does store, and limits the samples' use to legitimate, user-approved functions. However, even this extensive regulatory regime would do little to prevent the actual devices' usage for other malicious purposes; that smart television could still be used, potentially, to send spam email.

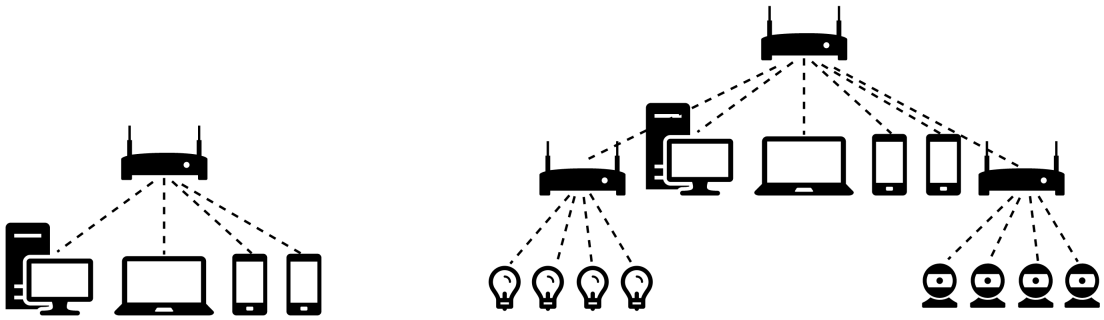
3.2 Network and Internet Solutions

Given that IoT devices operate with the same Internet and network protocols as traditional devices, malware strategies above the host layer change very little. However, many IoT devices operate on a cloud model, where the individual devices communicate with the device developer's network [85], such as a smart scale recording its users' weights to a cloud database. This model, combined with the intended single purpose functionality of many devices, may enable easy differentiation of illegitimate traffic - that is directed towards an unexpected destination, such as a C2 server - from the legitimate traffic to the expected cloud. In turn, this easy differentiation enables detection of compromise via network analysis and prevention of many symtoms through network whitelisting, which would restrict IoT devices to only communicating with their respective clouds. Unfortunately, the same privacy-protecting technologies that impeded analysis for traditional devices, Network Address Translation and Privacy Extensions, similarly prevent device-level identification.

Furthermore, the ability to detect certain classes of malicious activity through network monitoring (including the monitoring available to ISPs) may be stymied by impact resulting not from malware, but instead a corrupted legitimate function of an IoT device. Because the misbehavior is identical to an otherwise-legitimate function, the traffic may be indistinguishable at a network level. To reuse the example of a refrigerator with email capability, the traffic of refrigerators sending spam emails may appear identical to the traffic of those same devices sending legitimate milk reminders. However, as more of Internet traffic is encrypted by default, this type of insight is

growing less common. Instead, network monitoring has to rely on metadata analysis, which could still potentially detect the wayward refrigerator - for example, a spambot is likely sending a volume of traffic far exceeding the norm.

Finally, future home networks may be fundamentally more complex. Most current home networks consist of a single router and access point, to which all devices connect (see 3-1a). This centralizes access, thus allowing for a single device to have visibility over the full, layer 2 and 3,[2] metadata of all devices' communications. However, future home networks may have multiple access points, combining to create a complex, multi-tier topology (see 3-1b) [38]. In such a topology, the edge router has limited visibility: NAT or IPv6 Privacy Extensions could hide the identity of devices accessing the network through lower-tier access points, such as a specialized controller and access point for smart lightbulbs or home security cameras, as shown in the example.



(a) Simple (b) Complex

Figure 3-1: Example Home Network Topologies

[2]These numbers refer to the data link and network layers of the OSI network model, an abstraction useful for discussing computer networks. Specifically, these details include the source MAC address, part of layer 2, and the source IP address, part of layer 3.

Chapter 4

Designing EDICT

ISPs and customers, acting in coordination, have a strong ability to address malware infection. ISPs have the capability and incentive to detect malicious activity, but the residential source identification problem prevents useful device identification. However, customers are positioned to provide device-level identification. EDICT (Enabling Device-level Identification of Compromised "Things") addresses this challenge by providing customers a user-friendly, privacy-preserving way to identify a compromised device on their home network, given notice of that compromise from their ISP. This chapter outlines EDICT's system architecture and analyzes several technical and policy considerations in its development and implementation.

4.1 System Architecture

At the high level, EDICT is an open-source software package designed for home router operating systems. It operates within a seven step process, as shown in figure 4-1.

1. **Devices connect to router as normal.** All devices on the home network, from traditional devices to IoT devices, connect to the home router as they would traditionally. Connection can be wired or wireless.

2. **EDICT logs devices and connections.** EDICT operates by having packets mirrored via `iptables` (for IPv4 packets) or `ip6tables` (for IPv6 packets).

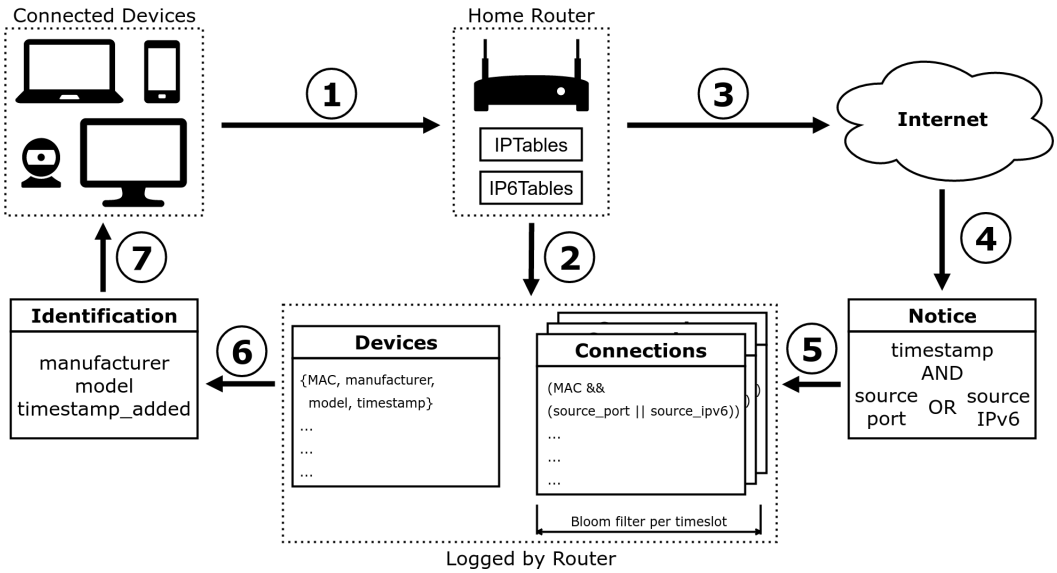


Figure 4-1: System Architecture. EDICT operates in seven steps:

1. Devices connect to router as normal
2. EDICT logs devices and connections
3. Router forwards packets
4. Customer's ISP detects malicious traffic and notifies customer
5. Customer queries EDICT based on ISP's notification
6. EDICT queries connection log to identify source for customer
7. Customer remediates infection as desired

First, EDICT examines the frame containing each packet for the source MAC address. If the MAC address is novel, EDICT creates a new entry in the device log containing the source MAC, information on the device manufacturer and model derived from the Google Fi project's Taxonomy library,[1] and the timestamp of the device's initial connection. For additional usability, EDICT could log the timestamps of known devices' connections, to provide the user with a "last seen" time stamp; however, this feature was eschewed for efficiency. Second, EDICT logs metadata about the packet into the current time slot of the connection log, a series of scalable Bloom filters.[2][3] This log is comprised of one scalable Bloom filter per time slot of logging, to enable the log to be written and queried by timestamp. While this length is tunable, EDICT defaults to a time slot length, l of 1 hour. Each entry in the connection log contains a concatenation of the MAC address and either the source port (for IPv4) or source IPv6 address.

3. **Router forwards packets.** Following EDICT's logging activities, the router will forward the packet as it would normally. NAT behavior is not affected.

4. **Customer's ISP detects malicious traffic and notifies customer.** As discussed in section 2.3, the customer's ISP will monitor the customer's traffic for malicious activity. When such activity is detected, the ISP will notify the

[1]This library, discussed in [30] and available at [31], uses MAC addresses and WiFi Probe and Association Requests to determine the manufacturer and model of an unknown, connected device.

[2]A Bloom filter [12] is a probabilistic data structure that allows for significant storage savings at the cost of a deterministic false positive rate. That is, the filter will return either "definitely no" or "probably yes" when queried about membership. The false positive rate is tunable and inversely proportional to the storage requirements. The filter operates by creating an array of m bits and passing the input to a series of k hash functions, each of which maps to a single bit in the array. Thus, an insertion flips k bits to 1. To query membership, the item is passed to the hash functions; if and only if the bits returned from all hash functions are 1, the filter returns positive. Thus, the collision of the hash functions over a series of inputs could cause a false positive, where the series of bits associated with a value have been turned to 1 by other inputs' hashes, without that value having ever been inserted into the filter.

[3]A scalable Bloom filter [8] is a complex, Bloom filter-based data structure that enables the efficiencies of a Bloom filter without requiring a priori knowledge of the required filter occupancy. It is initialized to a set, small size and maintains an estimate of the current false positive rate, determined by the current capacity and occupancy; if the false positive estimate exceeds a threshold, the filter expands by creating an additional filter, which is then used for new additions.

customer with metadata on a sample of the malicious traffic, including at a minimum the source IP address, source port, and timestamp. Note that the presence of NAT would obscure the true source address of an IPv4 host from the customer's ISP, thus requiring the use of the source port for identification. In either situation, the ISP must be capable of internal traceback to identify the source network (the customer).

5. **Customer queries EDICT based on ISP's notification.** The customer then launches EDICT's user interface (UI) and provides EDICT with the source IP address, source port, and timestamp from the notification. This interface is hosted by the router and accessible to the local network. For convenience, it is not integrated into the router's administrative interface.[4]

6. **EDICT queries connection log to identify source for customer.** First, EDICT iterates over the queue of Bloom filters to determine the correct filter, as determined by timestamp. EDICT then queries that time slot's Bloom filter on the concatenation of the MAC address and either the source port or source IPv6 address, depending on the IP version in use. If the timestamp is close to the start or end of a time slot, EDICT also queries the previous or next time slot's filter, respectively. If the queried filter(s) return positive, EDICT informs the user of the compromised device's manufacturer, model, and timestamp of first connection.

7. **Customer remediates infection as desired.** Finally, the customer receives EDICT's response and can remediate the infection in various ways, as determined by the capabilities of the compromised device: antivirus, hardware reset, power cycling, physical disconnection, or network quarantine.

[4]To accomplish this, the router hosts a separate, Node.js web server which handles the EDICT UI. The UI is not integrated with the router's administrative interface for several reasons: portability (EDICT's UI does not need to be redesigned for each router OS), accessibility (router administrative interfaces are often confusing; EDICT's goal is usability by any computer user capable of running antivirus), mobile friendliness (router UIs are often unusable on mobile devices), and delegatability (linking EDICT to the router UI means that allowing access to EDICT allows full administrative access to the router; home network administrators may wish to split these capabilities).

4.2 Technical Considerations

Operating on a home router entails many technical considerations, which in turn necessitate certain design choices for EDICT. These devices are often constrained, possessing limited processing power and memory. EDICT must not require its users to upgrade their routers; it must function on existing equipment. Furthermore, its use should have minimal interference with normal network traffic. If EDICT imposes a significant overhead on the network, it becomes less beneficial to the user. Finally, EDICT must be able to scale as networks grow in size and complexity. The future of home networks is uncertain, and EDICT should be adaptable as those networks expand.

4.2.1 Performance

EDICT is engineered to operate on modern home routers. As these devices are designed to be low cost, they often possess limited hardware capabilities. This necessitated deliberate design choices regarding EDICT's ability to perform with low processing ability and limited memory without negatively impacting the network's performance.

4.2.1.1 Processor

EDICT's architecture allows two assumptions regarding processing requirements. First, writes to the connection log will be common, but reads will be rare. Therefore, EDICT utilizes the scalable Bloom filter structure discussed in section 4.1. As an insertion into a Bloom filter requires k hash calculations, k does not vary, the scalable Bloom filter must expand when capacity c is reached, and another Bloom filter must be added once per time slot l, insertions into the connection log will run in $O(1)$ constant time:

$$O(k) + \frac{O(1)}{c} + \frac{O(1)}{l} = O(k + \frac{1}{c} + \frac{1}{l}) \equiv O(1) \tag{4.1}$$

Now, to read from the connection log, EDICT must determine the correct Bloom filter, which is determined in constant time from the notification's timestamp. Then, for the correct Bloom filter, EDICT iterates over n devices in the device log to determine which device was communicating over the relevant source port. Thus, reading from the connection log runs in $O(n)$ time, as shown below. However, n may be large: While traditional networks have a small number of connected devices, future networks, especially those under attack, may contain - or appear to contain, as discussed in **??** - a large number of devices.

$$O(1) + O(n) = O(n + 1) \equiv O(n) \tag{4.2}$$

Second and similarly, reads of the device log will be common, but writes will be rare. This log is read whenever a device sends traffic to the router, in order to determine whether to register a new device, and when EDICT is queried, to include additional information on the device for the user. This necessitates EDICT's use of a hash-based associative array. Thus, reading from the device log will run in $O(1)$ constant time, and writing when a new device connects will require $O(n)$ linear time. Additionally, to enable persistence across reboots, the device log writes each new device to disk and will recreate itself from disk when EDICT restarts. Because both new devices and reboots are rare, this has little impact on EDICT's requirements.

4.2.1.2 Memory

EDICT must require a low memory overhead per unit of time, in order to allow the overall log to persist as long as possible. There may be a significant delay between the malicious activity taking place, the ISP detecting the activity, the ISP notifying the customer, the customer receiving the notification, and finally the customer identifying the compromised device via EDICT. The storage requirements of a simple Bloom filter allow the construction of an estimate of EDICT's storage requirements per connection, S. This serves as a baseline for each timeslot; the individual timeslots' filters will scale geometrically as discussed in [8]. I estimate a single filter's baseline following [63],

where m is the number of connections tracked and ϵ is the maximum false positive rate.

$$S \approx m \times 1.44 \log_2(\frac{1}{\epsilon}) \text{ bits/filter minimum} \tag{4.3}$$

Because EDICT initializes each filter to store 15,000 connections (and will scale the filters when the estimated false positive rate exceeds 0.001), I can estimate the minimum storage required for each timeslot:

$$S \approx 15000 \times 1.44 \log_2(\frac{1}{0.001}) \approx 26.27 \text{ Kilobytes/filter} \tag{4.4}$$

In order to operate within the constraints of a modern home router using LEDE, as discussed in 5.1, EDICT fixes its maximum storage requirements to 10 Megabytes. This ceiling is within the memory available on most LEDE platforms.[5] Every $frac1\epsilon$ connections, EDICT will compute the current total size of the connection log. If that sum exceeds 10 Mb, EDICT will drop the oldest Bloom filter successively until the sum is below the threshold. This check is performed only sporadically in order to minimize impact on normal operations; in periods of high activity (e.g., a compromized device participating in a DDoS), the checks will become more frequent.

4.2.2 Scalability

As networks grow, network-centric security approaches become much more difficult. This is due to both the increase in the number of devices and thus the network's traffic, as well as the complexity of the network topology, as discussed in section 3.2. With limited additional work, EDICT can scale with growth on both axes.

Larger networks impose significant requirements on network monitoring systems, such as EDICT. As the amount of traffic increases, EDICT's router must have similar increases to its CPU and memory. However, home router capabilities have trended upwards as home Internet access has increased, and this trend is unlikely to stop -

[5]This assumes LEDE's memory requirements are equivalent those of its predecessor, OpenWRT. OpenWRT advertises a minimum requirement of 16 MBytes; most modern routers contain 32 MBytes [82]. This allows sufficient room EDICT's storage.

CPU and memory prices continue to fall. This follows from the notion that a router's capabilities must support the amount of traffic: Routing packets requires work, so a busier network requires a better router simply to operate, irrespective of security concerns.

Additionally, EDICT can scale to the complex network topologies, discussed in section 3.2, through the installation of EDICT on each access point to the network. In a simple network topology, an EDICT instance on the gateway router has full network visibility, as all devices on the network connect directly to this router. However, were a complex network topology to only have EDICT installed on the gateway router, the EDICT system would only have sufficient visibility of devices directly connected to that gateway; devices connected through secondary access points may be unidentifiable. Secondary routers would overwrite the MAC addresses of connected devices, preventing identification by an EDICT instance on a higher-level router. Moreover, secondary access points may have additional layers of NAT and DHCP, in order to allow portability of their connected devices (e.g., a smart lightbulb package may utilize this nested architecture to ensure plug-and-play usability). However, with EDICT installed at all access points, the EDICT system as a whole regains full visibility of the network. This allows for the identification of a device anywhere on the network through a recursive query:

1. The gateway router performs a standard EDICT query.
2. If the query returns a host, inform the user.
3. If the query returns another router or switch, that device performs an EDICT query.
4. Continue recursively performing EDICT queries until the compromised host is identified.

However, specialized access points (e.g., a controller for a home security system) may not be as readily upgraded with EDICT. Fortunately, specialized access points are likely to control specialized, homogeneous devices (such as an array of identical

security cameras): because these devices are identical, identifying the type of device infected (that is, the controller) should be sufficient for effective remediation.

4.3 Policy Considerations

To succeed as an anti-malware strategy, EDICT must address several policy realities, in addition to the technical ones. This section demonstrates EDICT's feasibility as a tool of policy, by investigating several of these concerns: first, costs imposed on several critical classes of actors; second, coordination requirements necessary for effective implementation and usage; third, legal questions related to ISPs' and customers' authorities to monitor communications and act on that information; and fourth, the impact to users' privacy.

4.3.1 Cost

First, EDICT will decrease the costs of malware on the key actors in its adoption, consumers and their ISPs. It will have no impact on the costs of IoT developers and manufacturers, and the costs required by router manufacturers will be acceptably low. I focus on these actors, as any malware remediation strategy may, at least indirectly, affect them. Conversely, I exclude from the analysis the victims of malicious traffic, who would be equally positively affected by any substantive effort, and the controllers of botnets, who face an equal negative outcome from all efforts bar those directed squarely against them. Across all actors, I discuss cost in the economic sense, not merely the monetary: remediation time, frustration, and additional maintenance requirements are all forms of "cost."

4.3.1.1 Customer

EDICT will generally decrease customers' costs. It increases consumer choice, by allowing consumers additional alternatives for malware remediation, thus allowing consumers to choose the lowest cost option. Without EDICT, customers are faced with a binary choice: continue to bear the costs of malware or attempt some form

of brute-force remediation, which is infeasible in a large and varied network. With EDICT, customers have additional options: place the device into a network quarantine, physically disconnect it, reset the device - perhaps by simply power cycling it, or run antivirus if possible. EDICT allows customers to efficiently target their remediation efforts, thus lowering their cost. This is specially important when considering certain IoT devices that may be critically important to the user. A brute force approach (that is, manually remediating all connected devices in order to ensure the compromised device is remediated) may unduly impact important devices, such as smart door locks or smoke detectors. With EDICT in place, uncompromised devices need not be affected.

However, this analysis rests on a pair of assumptions. First, I assume that notifications are effective. Customers may fail to receive them or ignore them outright, and without an effective notice, the customer is reduced to alternative means of detecting the compromise, which are infeasible in the IoT. In support of this assumption, a 2015 study showed that notification of botnet activity decreased the mean time to remediation [56]; another study, conducted on Australian ISPs, note that most ISPs received an overwhelmingly positive response from customers who received notifications [84]. Second, I assume that the customer bears cost from malware. There are several thought experiments to the contrary: a consumer, who cares nothing for his own privacy, being infected by a strain of malware that streams his home security cameras globally, or another consumer, whose typical Internet behavior entails low bandwidth requirements, fails to even notice the network performance impact of a DDoS bot. This assumption remains valid, due to the notification scheme: The notice itself broadcasts malware's cost to the victim, thus incentivizing remediation.

Finally, this analysis rests on EDICT being usable and unobtrusive - merely running it must not impose additional cost on users. This is a function of its design and implementation. First, this rests on EDICT being high performance, as discussed in section 4.2.1 and analyzed in section 5.1: EDICT must not significantly decrease the network's performance or impede functionality of the router. Second, EDICT must be usable. It is designed to require user interaction only when identifying a device

(versus, e.g., a design requiring user interaction to connect a device for the first time), and its user interface is implemented to be as accessible as possible.

4.3.1.2 Internet Service Providers

Similarly, EDICT is expected to decrease ISPs' costs. Their costs are, in large part, a function of the performance and security of their network - so, by increasing the performance and security, costs are decreased. EDICT can improve network performance by enabling successful bot remediation, thus reducing the impact of DDoS campaigns and other malicious activity. Similarly, the reduction in malware increases security, which can lower the operating costs of the ISP. Increased security, furthermore, can increase the ISP's reputation, which can allow it to market "security as a feature" or potentially leverage better transit deals with its own providers.

With modified implementation, EDICT could provide ISPs an additional security benefit: threat intelligence. When EDICT determines the specific infected device, it could anonymously forward that information - manufacturer and model - to the ISP, who would then be able compile a database on threatened systems. This could even be used to generate revenue, through the database's resale to third party security firms. However, this approach should be opt-in for users and emphasize the anonymization of user data, to ensure that users' privacy is not breached.

This analysis assumes that ISPs will initially find a customer notification scheme in their best interest, but this assumption is validated by the precedent for both detection and notification schemes. First, ISPs worldwide are beginning to enact full notification schemes. In the United States, Comcast has introduced Constant Guard, an automated system where customers are notified of bot-like activity and given information on antivirus usage [60]. In Europe and elsewhere, several initiatives have taken a more systematic approach, with multiple ISPs implementing in concert, likely to forestall any first mover disadvantages [27, 68, 6].

Second, many more ISPs have systems in place to detect malicious activity [51]. EDICT would greatly decrease the costs to transform these systems into notification schemes, mainly by decreasing customer service costs. When informed of suspected

infection, a customer's first response will likely involve contacting customer service. Providing information on the notification and suggested remediation (as done by Comcast [60]) will assist many customers, this approach is imperfect and rests on an assumption that customers can successfully remediate the infection. Due to the residential source identification problem, this assumption is likely invalid. However, EDICT provides ISPs with a solution: When an ISP notifies a customer, it points them towards EDICT (which hopefully has already been installed on the ISP-provided router), thus allowing the customer to target their efforts towards the specific infected device.

4.3.1.3 IoT Developers and Manufacturers

IoT developers and manufacturers would bear no immediate cost from this approach, as EDICT only requires coordination between ISPs and their customers; it requires no input or additional functionality from individual customer devices. Indeed, this was a deliberate constraint for EDICT, following the assumption that IoT manufacturers are disincentivized to provide remediation capabilities. However, in the long term, a notification scheme coupled with EDICT may provide an incentive for IoT manufacturers to develop remediation capability. Given the discussed constraints on an IoT device, this may not be an antivirus suite, but perhaps the ability for a device's owner to reset that device to a known good state. As customer security awareness increases, from both an increased threat and a notification policy, IoT manufacturers may be pressured into increasing their devices' security by introducing remediation capability. Unfortunately, most current proposed standards requirements for the Internet of Things eschew this discussion: Security requirements are information-centric and seek to protect from privacy violations by outside actors, not the compromise of devices [22, 89, 72]. However, as EDICT enables customers to determine the specific infected device, customer service costs should increase for device manufacturers: Instead of contacting their ISP in response to a notification, customers can now direct their confusion and ire at the device manufacturer.

4.3.1.4 Router Manufacturers

EDICT would require a small cost from router developers, simply to implement the system on their devices. However, modern consumer router hardware already supports the necessary functionality (logging, perhaps device-level quarantines); the devices simply need additional software to implement those capabilities in a user-friendly way. Long term, home routers' security responsibilities may become a norm. This would incentivize router manufacturers to increase their devices' hardware capabilities, even for the low-end models, but these costs are low relative to the overall production costs. Moreover, this decision would be in router manufacturers' best interests: Customers would expect routers to deliver on security, and effective security requires a certain level of hardware capability. Additionally, many routers are purchased by ISPs and then re-sold or leased to the ISPs' customers. If ISPs begin to adopt notification schemes and support EDICT, this would further encourage router manufacturers to produce highly-capable devices.

4.3.2 Coordination Requirements

EDICT does not require significant coordination between competing actors. This is important, because coordination requirements between ISPs plagued the literature of IP traceback: every ISP between the source of malicious traffic and the destination of that traffic had to participate in a traceback scheme, preferably the same system for ease of automation. This is an extreme example of a network effect, where the utility of a product is based on the number of users of that product. Thus, a choice with strong network effects and a small userbase is disproportionately costly. With EDICT's approach, such effects are not a challenge. A single ISP that enacts a detection-and-notification policy will see benefits, irrespective of the choices of the other ISPs. Moreover, a single customer who chooses to follow these notifications will also see benefits, irrespective of the choices of the ISP's other customers. This is critical for overcoming the first mover disadvantage created by strong network effects, which would strongly disincentivize the first potential adopters of this policy and thus

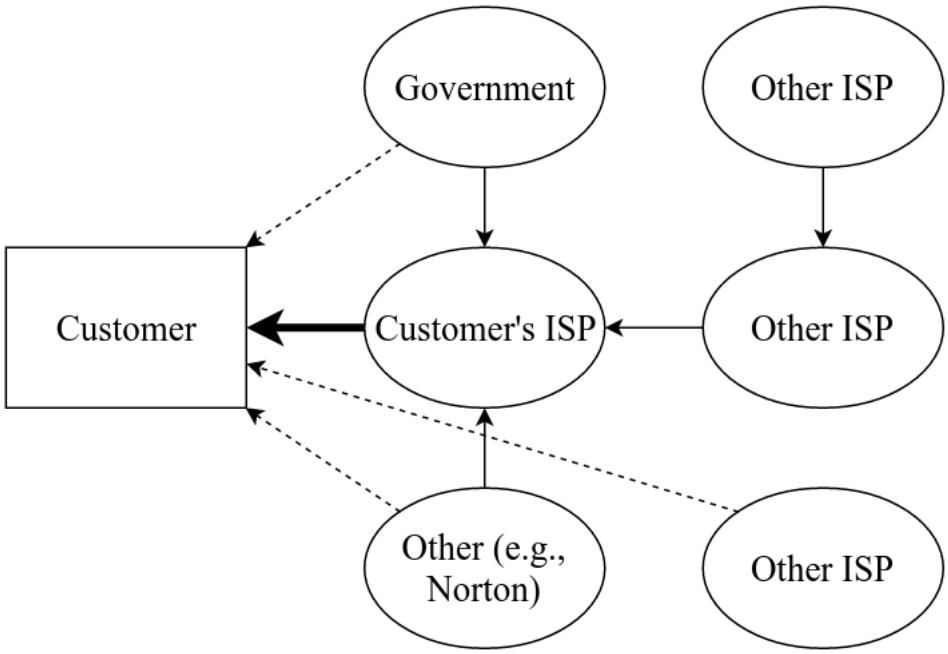


Figure 4-2: Malware Detection and Notification Routing. Various actors could have responsibility for detecting and/or notifying customers of malware infestation. Direct coordination between ISPs and their customers (the bold line) is the simplest, but other ISPs, governments, and third parties might detect malicious activity and notify the customer's ISP (solid lines) or the customer directly (dotted lines).

condemn any such scheme to non-adoption.

EDICT is engineered to focus on the customer-ISP relationship for simplicity and trust, but it is extensible to alternative notification schemes. A compromised customer's ISP may not detect the malicious traffic; instead, it might be detected by another ISP, a third party such as a government or computer security firm, or even the victim of that traffic. Figure 4-2 shows the possible routes a notification could travel to reach the compromised customer. However, because these routes lack the preexisting business relationship between an ISP and its customers, the alternative approaches elicit questions regarding trust - how does the customer trust the notification's validity and authenticity? - and legality. The legal issues are treated further in section 4.3.3. The trust issue is sufficiently complex to warrant a separate paper, but the broad methodology would likely involve cryptographic signatures.

4.3.3 Legality

EDICT raises several legal questions, which are broadly organized around the activities that the router's owner and the customer's ISP can take. While these questions cover EDICT's implementation and usage, they also discuss the nuances introduced by future additions to EDICT, such as an optional network quarantine for the devices that EDICT identifies.

Can EDICT's owner log traffic or quarantine devices? Yes. The majority of these cases will be simple home networks, where there are no requirements to announce the terms of service; it is implicit in connecting to the network. However, there is a possible exception when routers with EDICT are used in non-home settings, such as hotels or small businesses. These scenarios typically have a terms of service agreement and/or an acceptable use policy that would protect the interests of the router's owner, thus allowing the router to monitor and log traffic for this purpose.

Can an ISP monitor traffic and notify a customer of infection? Yes. ISPs have terms of service that protect their ability to monitor traffic. ATT [83], CenturyLink [2], Comcast [3], Time Warner Cable (TWC) [43], and Verizon [58] all reserve the right to monitor customers' traffic for security reasons. Additionally, the Federal Communication Commission's (FCC) 2015 Open Internet Order [41] and 18 U.S.C. §§ 2510-22 [88] provide for ISPs to take measures to secure their networks. Per the Open Internet Order, ISPs must simply disclose their network security practices; this is typically done through the aforementioned terms of service. Finally, there is significant precedent: Several US ISPs already monitor traffic to detect malicious activity [51]. Unfortunately, were EDICT utilized with an alternative coordination scheme, there may be confusion. The FCC's Notice of Proposed Rulemaking regarding the application of the Communications Act's privacy protections to ISPs forbids information sharing outside the traditional activities of an ISP [42]. While this was intended to prevent advertising and other commercial activities, it may unintentionally prevent an ISP or other actor from sharing notification of an infection. That is, the infected customer's ISP fails to detect the infection, but another ISP does; that

other ISP may be unable to inform the customer's ISP about the infection.

Can an ISP quarantine traffic or take other actions in response to infection? Yes. ISPs' terms of service also protect their ability to control their networks. The terms of service examined above [83, 2, 3, 43, 58] all prohibit malicious activity, spam email, denials of service, or some combination thereof; they all also reserve the right to suspend or terminate customers' service for these violations. Moreover, the Federal Trade Commission, in a letter to ISPs, recommends quarantining infected customers, although this recommendation was specific to spam email - not DDoS campaigns or other threats [77]; academic arguments, supported by official Microsoft statements [71], have suggested this extend to all compromised machines. Finally, while this example is international and thus less applicable to a question of US legality, the Australian Internet Security Initiative allows ISPs to determine their own response strategy for infected customers, and some ISPs choose to quarantine repeat offenders to protect the overall network as well as draw the customer's attention [84]. Examples of ISPs quarantining customers, however, remain the minority; most ISPs, in the US and elsewhere, absorb the cost of malware infections, even for customers who have been repeatedly notified.

4.3.4 User Privacy

EDICT was designed to protect users' privacy. However, any Internet logging system evokes privacy concerns, especially one like EDICT which can be used to de-anonymize specific devices and whose logs persist for a usable period of time. Worsening these fears, EDICT is designed to run on home routers, which are notoriously insecure and often targeted by modern adversaries [74]. Yet, EDICT's design prevents any gross breaches of privacy, beyond those already available through a compromised router - such as sniffing[6] all traffic or performing a man-in-the-middle.[7] EDICT's

[6] "Sniffing" traffic refers to the monitoring of traffic on a network, typically by placing a network card into "promiscuous mode," where the card will process all network frames, instead of reading only those frames directed at that device.

[7] "Man-in-the-middle" is a class of attacks where the attacker places himself between a pair of actors, typically a server and a client, in order to view or modify the communications between those actors. A compromised router provides an adversary with man-in-the-middle positioning.

logging eschews any information on the destination of traffic, forcing an adversary to breach both an individual's home router (and thus EDICT) and that individual's ISP (assuming their ISP keeps persistent logs) and cross-reference those two logs to generate a device-level log of traffic. While for highly sensitive behavior, such as political activism, it might be sufficient to simply determine the traffic's destination and some idea on its source (i.e., which customer network), EDICT does not add any capability to an adversary.

While EDICT's design protects users' online privacy, router compromise may enable attacks on users' real life privacy. Logged source information is not enough to determine the full details of Internet browsing, but it can establish a "pattern of life," which can then be used to determine when a specific user goes to sleep or wakes up, leaves for work or arrives back home, is gone for a long period of time (e.g., vacation), etc [15]. While this analysis is certainly possible for a compromised router without EDICT, EDICT's logs bootstrap it by providing data on the time prior to router compromise.

Chapter 5

Analysis

To demonstrate that EDICT meets the design intent outlined in the previous chapter, we analyzed several characteristics in more depth. First, EDICT was subjected to a performance analysis of its impact on CPU and memory, in order to show that it meets its technical constraints. Second, EDICT was examined for security and privacy flaws, to ensure its effectiveness and address concerns over the privacy of users in the face of logging of potentially-sensitive Internet behavior.

5.1 Performance Analysis

To analyze EDICT's performance, we developed an experimental setup designed to test EDICT's memory usage, CPU utilization, and impact on network throughput.[1] We tested memory usage, as memory is both a significant technical requirement for EDICT (see section 4.2.1.2) and routing and switching require a base level of memory, such that extremely high memory usage by EDICT may impact the router's core functionality. Similarly, we tested CPU utilization, both to directly measure EDICT's impact on the router, as well as determine the overhead required for EDICT. As routers become more featureful - perhaps being used to protect children from adult content or screen traffic for viruses - EDICT should not explicitly preclude the router's

[1]Memory and CPU utilization were tested using the vmstat system monitoring utility, which was executed remotely on the router hosting EDICT. Network throughput was tested using the iperf network measurement tool.

use for other tasks. Finally, we tested network throughput to measure EDICT's overall impact. These metrics are in line with performance analyses found in the IP traceback literature (e.g., [75]). While previous analyses often included tests on the performance and efficacy of queries to the traceback log (e.g., false positive rate), this experiment eschews these measurements, as queries to EDICT will be relatively rare.

This analysis was conducted on a simple experimental setup. EDICT was implemented on a customized build of the Linux Embedded Development Environment (LEDE) operating system, a successor to the OpenWRT operating system, and installed, together with LEDE, on an ASUS RT-N66U consumer router. A "client" personal computer was connected to this router; all testing was initiated from this client. The LEDE router was then connected to a second router, to simulate the Internet routing infrastructure. Also connected to this second router, a "server" personal computer simulated the destination of the client's traffic. Both routers were isolated from other networks, including the Internet.

To measure memory usage, CPU utilization, and network throughput, EDICT was tested under two scenarios. First, normal network usage was simulated with multiple TCP flows generated with the iperf3 network measurement tool. This approach was designed for maximum network utilization, in order to determine the overhead that may be required by EDICT in day-to-day network use (e.g., video streaming). Second, the first scenario was combined with a rapid network scan generated from the nmap network scanner, configured to scan as quickly as possible.[2] This is designed to simulate what we judge as the worst case scenario, a device infected with a worm that is trying to rapidly find other vulnerable devices and infect them, thus creating a high number of unique connections and stressing EDICT.

For each scenario, we established two experimental groups: EDICT enabled, without and with its query web interface enabled. As EDICT's interface was developed to operate separately from LEDE's normal interface, the experimental groups were split thusly to isolate the impact of EDICT's critical components. Both of these groups were compared against a control, a LEDE router with EDICT disabled, using

[2]This was accomplished using nmap's "insane" timing template.

a pairwise t-test. Finally, the experimental group with enabled UI was compared against the group with disabled UI, to identify if the UI had any significant effects on performance.

All groups (experimental and control) were tested over 60-minute periods, in order to ensure a change of sublog periods. Thus, a full round of testing consisted of 6 blocks of testing, as listed below:

1. EDICT disabled, without portscan
2. EDICT disabled, with portscan
3. EDICT enabled, EDICT UI disabled, without portscan
4. EDICT enabled, EDICT UI disabled, with portscan
5. EDICT enabled, EDICT UI enabled, without portscan
6. EDICT enabled, EDICT UI enabled, with portscan

5.1.1 Memory

As shown in tables 5.1 and 5.2 and figures 5-1 and 5-2, EDICT imposed a small, but statistically significant, memory requirement, exactly as expected. While the experimental design only required an hour of traffic be logged, continued use would lead to an increase in EDICT's memory requirements to approximately 10MB more than currently used (as 10MB is the total memory allocated for EDICT's connection log). Further testing could determine the average duration of this log capacity, in order to allow users to more accurately allocate memory to meet their expected duration requirements (e.g., a week's worth of logs might require 15 MB of memory).

There is one minor anomaly, noticeable on the portscanned graphs: All three groups have a period, approximately 15 minutes long, with a constant increase in free memory. This is an artifact of the nmap scanner, which was configured to scan the entire subnet of the "server" machine and restart upon completion. Each of these scans took approximately 15 minutes. For an unknown reason, perhaps caching, one iteration of this scan took fewer resources for each iteration.

Figure 5-1: Free Memory without Portscan (KB) over Time

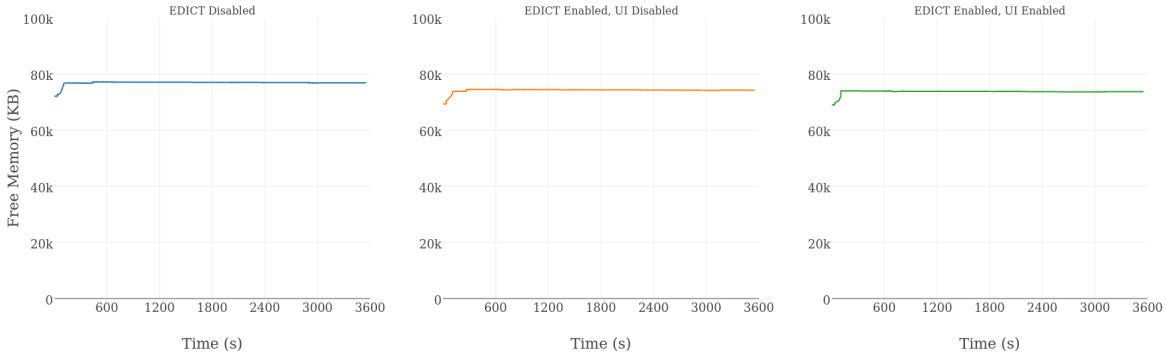


Figure 5-2: Free Memory with Portscan (KB) over Time

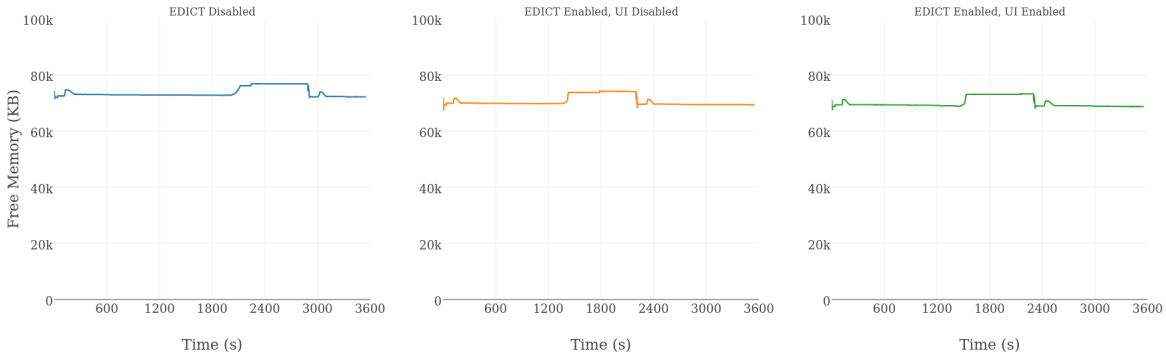


Table 5.1: Free Memory without Portscan (KB)

Group	Mean	SD	*t*-test v. G1		*t*-test v. G2	
			Δ	p	Δ	p
1. EDICT disabled	76,871	690	-	-	-	-
2. EDICT enabled, UI disabled	74,217	648	-2,654	0.00	-	-
3. EDICT enabled, UI enabled	73,667	667	-3,204	0.00	-550	0.00

Table 5.2: Free Memory with Portscan (KB)

Group	Mean	SD	*t*-test v. G1		*t*-test v. G2	
			Δ	p	Δ	p
1. EDICT disabled	73,706	1,676	-	-	-	-
2. EDICT enabled, UI disabled	70,698	1,782	-3008	0.00	-	-
3. EDICT enabled, UI enabled	70,059	1,672	-3647	0.00	-639	0.00

5.1.2 CPU Utilization

EDICT imposes no appreciable impact on CPU utilization, as shown in tables 5.3 and 5.4 and figures 5-3 and 5-4 . However, the router appears to be CPU-limited, as the CPU utilization is consistently maximized, despite memory remaining available and the network infrastructure supporting gigabit speeds. While the lack of impact on network throughput, as discussed in section 5.1.3, implies that EDICT's impact on CPU utilization is limited, further testing is required to understand the true impact. This could be accomplished on more capable hardware, such that the network hardware (e.g., cabling) becomes the limiting factor, or the network simulation (iperf) could be reduced to a level that does not maximize the CPU in normal operations.

Figure 5-3: CPU Utilization without Portscan (%) over Time

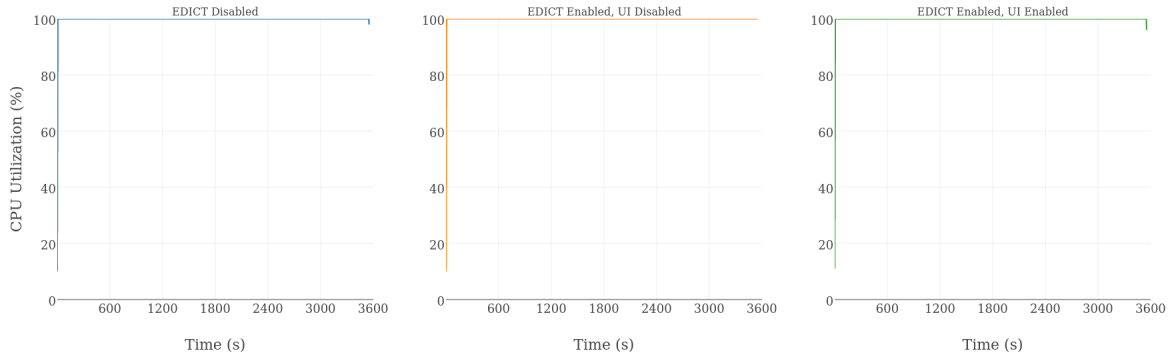


Figure 5-4: CPU Utilization with Portscan (%) over Time

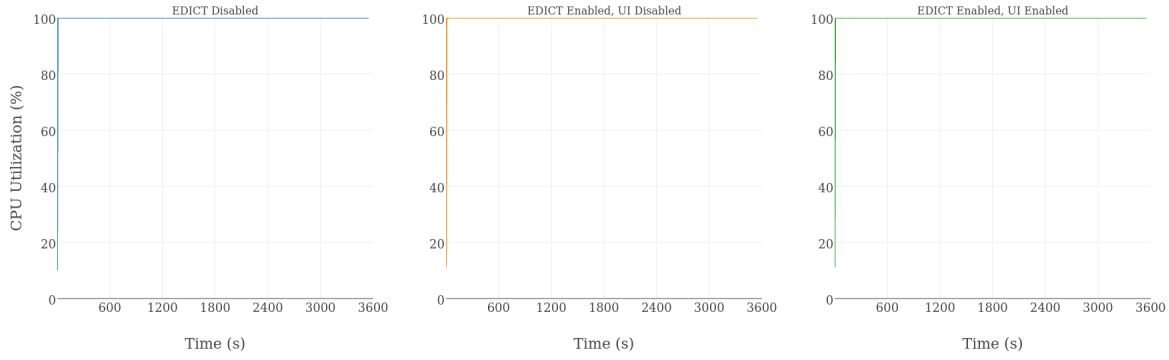


Table 5.3: CPU Utilization without Portscan (%)

Group	Mean	SD	t-test v. G1		t-test v. G2	
			Δ	p	Δ	p
1. EDICT disabled	99.97%	1.51	-	-	-	-
2. EDICT enabled, UI disabled	99.97%	1.51	+0%	0.32	-	-
3. EDICT enabled, UI enabled	99.97%	1.49	+0%	0.65	+0%	0.47

Table 5.4: CPU Utilization with Portscan (%)

Group	Mean	SD	t-test v. G1		t-test v. G2	
			Δ	p	Δ	p
1. EDICT disabled	99.97%	1.51	-	-	-	-
2. EDICT enabled, UI disabled	99.97%	1.49	+0%	0.32	-	-
3. EDICT enabled, UI enabled	99.97%	1.49	+0%	0.32	+0%	1.00

5.1.3 Throughput

Similarly, EDICT imposes no statistically-significant impact on network throughput, as shown in tables 5.5 and 5.6 and figures 5-5 and 5-6. In the groups subject to portscanning, there is a noticable increase in throughput for approximately 15 minutes, due to nmap as discussed in section 5.1.1.

Figure 5-5: Network Throughput without Portscan (Mbits/sec) over Time

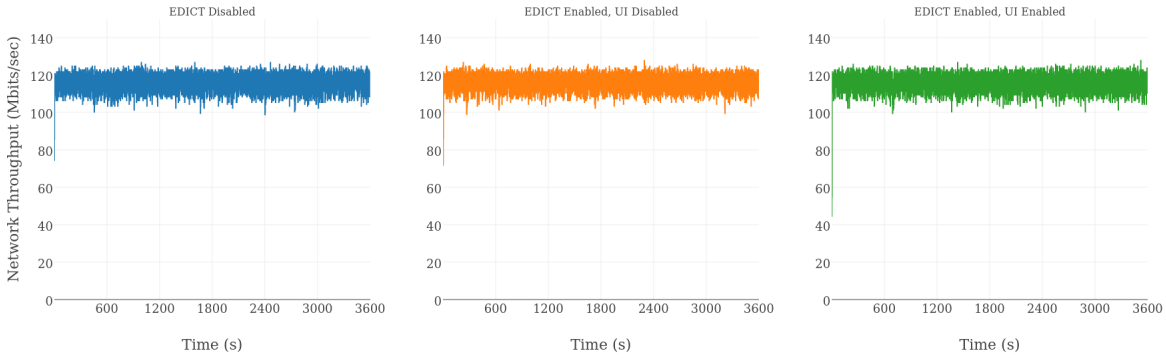


Figure 5-6: Network Throughput with Portscan (Mbits/sec) over Time

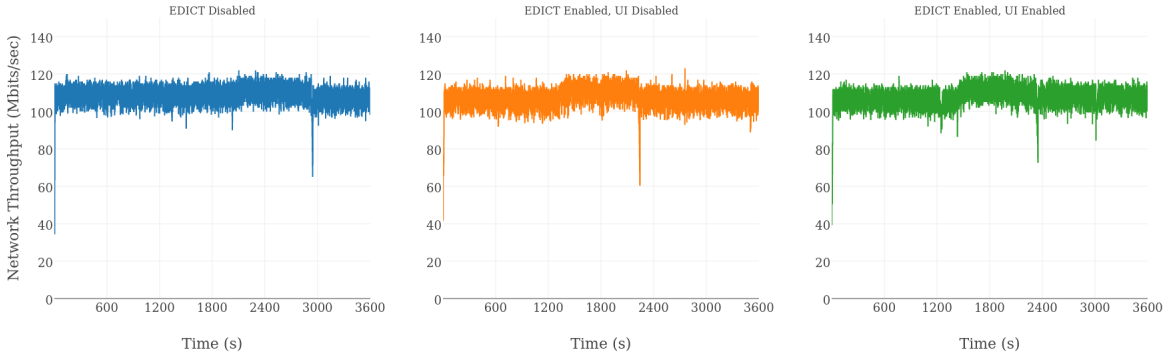


Table 5.5: Network Throughput without Portscan (Mbits/sec)

Group	Mean	SD	*t*-test v. G1		*t*-test v. G2	
			Δ	p	Δ	p
1. EDICT disabled	115.91	5.46	-	-	-	-
2. EDICT enabled, UI disabled	116.04	5.19	+0.13	0.28	-	-
3. EDICT enabled, UI enabled	115.86	5.60	-0.05	0.74	-0.18	0.16

Table 5.6: Network Throughput with Portscan (Mbits/sec)

Group	Mean	SD	*t*-test v. G1		*t*-test v. G2	
			Δ	p	Δ	p
1. EDICT disabled	108.95	5.99	-	-	-	-
2. EDICT enabled, UI disabled	107.38	6.08	-1.57	0.32	-	-
3. EDICT enabled, UI enabled	108.02	6.05	-0.93	0.32	0.64	1.00

5.2 Security Analysis

While EDICT is engineered to contribute to anti-malware efforts, it is fundamentally unable to address certain types of malware. This section discusses four of these gaps: MAC randomizers/spoofers, IPv6 randomizers/spoofers, source port manipulators, and malware that targets the router itself. While the former three types are viewed as threats to the benefits of EDICT - that is, they would reduce or eliminate EDICT's ability to identify a compromised device - the latter is instead viewed as a threat to both EDICT's efficacy and users' privacy and security. In this analysis, the threat model assumes the adversary can utilize the full capabilities of the compromised

device, to include the production of arbitrary packets and/or frames.

5.2.1 MAC Spoofing and Randomization

MAC spoofing, broadly, refers to the forgery of the source MAC address on frames a device sends over a network. This can be done by either generating a pseudorandom MAC address for the source or by sniffing the network's traffic, detecting a MAC address used by another connected device, and using that MAC. For the remainder of this analysis, MAC spoofing refers exclusively to the latter capability; MAC randomization refers to the former.

5.2.1.1 MAC Spoofing

MAC spoofing is typically used to avoid MAC-based network authentication. Some networks, particularly wireless, are configured with a whitelist of known MAC addresses; connections are refused from devices not on this list. Thus, an adversary wishing access to this network may sniff for MAC addresses in use and use an observed address as a forged source for access to the network. However, it can also be used more maliciously, wherein an adversary forges a victim's MAC address, conducts some unauthorized activity, and hopes the forged MAC leads to the victim being blamed for the effects.

If a compromised device utilizes a spoofed MAC, EDICT will be unable to accurately identify the device. The following scenario demonstrates this vulnerability. First, a connected device is compromised, but takes no overt malicious acts; it simply monitors the normal network traffic, recording other MAC addresses it sees. Second, the device begins to act maliciously, but forges a legitimate address as the source of its traffic. If EDICT is queried to identify the source of this traffic, the innocent device would be identified, and there would be no easy way to determine the forgery had occured.

To fully prevent MAC spoofing, EDICT would require a network authentication scheme that can uniquely identify connected devices. Typically, this involves

certificate-based authentication. With unique certificates installed on each device, the router can cryptographically verify the identify of each device. However, the current approaches to certificate-based authentication rely on significant user intervention, making this approach infeasible for large home networks. Alternatively, the router could authenticate devices based on their network signature, much as Google's taxonomy library does, but the validity of this approach under continuous communications (as opposed to initial connections only, which is used by Google's library) requires further study.

5.2.1.2 MAC Randomization

MAC randomization, conversely, is typically used to preserve users privacy. In the last several years, this approach has become more common. Modern iPhone and Android smartphones use a randomized MAC while in wireless discovery mode[3] to prevent traffic analysis: Each wireless access point (WAP) is presented with a unique, pseudorandom MAC address each time it is discovered [90]. Additionally, some Windows 10 devices now support MAC randomization. These devices allow a different approach, where the device either generates a new pseudorandom MAC every 24 hours (and the same MAC is presented to all devices encountered during this period) or generates a new MAC for each WAP it encounters (and remembers those pseudorandom MACs, such that each time a WAP encounters the Windows 10 device, it receives the same MAC)[39].

As long as a pseudorandom MAC address does not collide with another device's MAC, EDICT will still be able to accurately identify the compromised device. However, MAC randomization allows for EDICT to be victimized by resource exhaustion attacks. Because EDICT has to record the device details of each new connection, an adversary could execute a malicious port scan, or any other network sweep that generates many unique connections, with each connection coming from a different pseu-

[3]Wireless discovery mode is the default, unconnected state of a wireless device. In this mode, the device is actively scanning for nearby WAPs in order to determine their Service Set Identifier (SSID). This requires broadcasting wireless frames, which contain the source MAC address, thus potentially allowing for attacks on privacy through tracking the appearance of a MAC across a WAP or multiple WAPs.

dorandom MAC. This could exhaust the router's CPU and/or memory, as EDICT would first confirm each of these connections as originating from a novel device, and then store information on that device. Later, this could also interfere with EDICT's identification capabilities: As the device log is not optimized for reading (identification is relatively rare), a large log could slow down the identification process to an unacceptable speed.

However, EDICT could mitigate MAC randomization through the use of a walled garden for access control, where new devices have limited connectivity until manually authenticated through a captive portal or side channel. While this approach is typically seen only on public or academic wireless networks, it would also prevent compromised home devices from using novel MAC addresses. This modification could also be used to verify information on the connected device generated by the Taxonomy library: novel devices are placed into a walled garden, and the device owner must enter identifying information (via a captive portal or a side channel web interface) in order to grant the device connectivity. Importantly, this addresses the inability of EDICT to easily differentiate for a user between multiple devices of the exact same make and model. However, this approach would not prevent MAC spoofing, as EDICT would confuse the malicious traffic with legitimate traffic and allow it through.

5.2.2 IP Spoofing and Randomization

Similarly, IP spoofing is the forgery of a packet's source IP address, and it is used to hide the true source of a packet and/or falsely attribute a packet to another source. This section adopts a similar convention to the previous: IP spoofing refers to the forgery of another connected device's legitimate source address (which can be detected similarly to other devices' MACs), and IP randomization refers to the usage of a pseudorandomly-generated source address.

5.2.2.1 IP Spoofing

While IP spoofing may be intended to misdirect attribution, its efficacy depends on two factors. First, what version of the IP is in use? For IPv4, the impact is limited, as 1) EDICT eschews the use of IPv4 addresses for identification and 2) EDICT assumes that NAT would replace the forged source address with the router's IP address. For IPv6, a spoofed address can cause EDICT to identify legitimate devices in addition to the compromised device, as a query for the spoofed address would return a listing of all devices that used that address at that time.

Second, is the spoofed address on the same subnet? The above analysis assumes the spoofed address is within the same subnet as the true address, as spoofed addresses outside of the subnet would either be filtered out by their ISP following the IETF's Best Current Practice (BCP) 38 [62] or cause the ISP to contact the wrong customer (as the packets would appear to come from somewhere else on the Internet).

Spoofed addresses within the same subnet could be easily mitigated by implementing a fixed IP addressing scheme and filtering exceptions,[4] but this approach has significant drawbacks. It imposes the overhead of network administration as devices as connected and disconnected from the network. Beyond being time-consuming, this is likely beyond the abilities of many consumers. Additionally, it limits inter-network mobility: To connect to a new network (such as a friend's network while visiting), the network administrator must now add the new device.

Conversely, there is no simple solution to cross-subnet spoofed addresses. Generally, it would require broad adoption of BCP 38 filtering (to block the false addresses) and/or IP traceback capability (to determine the true addresses), but the specific challenges are beyond the scope of this research.

[4]Under this scheme, every device connected to the network would have a fixed IP address instead of being leased one through the Dynamic Host Configuration Protocol (DHCP). Then, traffic originating from a Network Interface Card (NIC) that does not map to the assigned IP address would be dropped, thus preventing devices from spoofing IP addresses.

5.2.2.2 IP Randomization

IP randomization also varies in its efficacy between the two IP versions. For IPv4, the set of possible addresses is relatively small, creating a high probability of collisions, where the source address pseudorandomly picks the address of another legitimate device, either on the same subnet or elsewhere on the Internet. In effect, this is the same as a directly-spoofed address, which is discussed in section 5.2.2.1.

However, IPv6 randomization has less of an impact, as EDICT will store the source address for IPv6 communications, allowing for even a pseudorandom address to be uniquely identified as a device. This approach is less vulnerable to a resource exhaustion attack, compared to MAC randomization, as the source addresses are being stored in a space-efficient scalable Bloom filter. This is important, as IPv6 randomization is increasingly common: IPv6 Privacy Extensions allow for an IPv6 host to anonymize itself by using a unique, pseudorandom host identifier for each communication; because the network identifier is preserved, the source will still receive replies, but the anonymized host identifier prevents outside observers from tracking the Internet activity of individual users or devices.

5.2.3 Source Port Manipulation

While EDICT is somewhat vulnerable to source port manipulation, its design mitigates the threat. Because source ports are typically randomized by the source device, EDICT's usage of source ports for identification is not significantly impeded by collisions, where multiple devices use the same source port and this may appear indistinguishable to EDICT. However, clever malware may fix the source port(s) of its traffic, such that EDICT collides. This could be done through observation of legitimate network traffic, wherein the malware observes a source port in use and uses it within the same log timeslot, or it could be done by observing a protocol that utilizes a fixed source port, such as the Dropbox LAN synchronization protocol [98], and using that port. EDICT's design mitigates the impact of such collisions, as it will return the list of all devices that used the queried source port at that time. Thus, the compromised

device can only hide itself in the subset of legitimate devices using that source port, which should be small as legitimate devices will be randomizing their source port or using specific protocols.

Moreover, EDICT could be modified to further mitigate this threat, by allowing the home network's owner to optionally configure EDICT to store information on the destination of traffic. This would force a compromised device attempting to hide to only target malicious traffic against the destinations of traffic from legitimate devices, vastly decreasing the malware's utility. However, following the privacy concerns discussed in section 4.3.4, EDICT eschewed this approach, and even the modified version would default to this option being disabled. To further assuage privacy concerns, EDICT could only log partial information on the destination, such as the most specific 16 bits of the destination IP address, preventing even loss of this data from readily compromising privacy.

5.2.4 Router Compromise

Finally, we consider the impacts of the router hosting EDICT being compromised. This analysis centers on two questions. First, what impact does router compromise have on EDICT's ability to identify devices? Second, what impact does router compromise have on EDICT's protections of user privacy? Notably, these questions do not cover the full scope of the threats posed by a compromised router. Fully analyzing those threats is beyond the scope of this thesis, so this analysis focuses on the problems directly related to EDICT.

Router compromise can entirely negate EDICT's ability to identify devices, both for devices whose communications were logged prior to the router's compromise and to devices communicating post-compromise. Adversaries can prevent identification of pre-compromise communications by simply deleting the log files. Alternatively, they can prevent identification by manually editing the log. In a scenario where an adversary has compromised device A and wishes to prevent EDICT from identifying this device:[5]

[5]In this scenario, the adversary desires to corrupt EDICT such that it returns a negative result

1. Adversary can remove device A's entry from the device log, so EDICT will never attempt to query its connection log for device A, or

2. Adversary can manually set 1+ bit(s) in the appropriate Bloom filters to 0. Bit(s) should be in the set of results from device A's details being passed into the k-hashes. When the filter is queried for device A, it will not return $\{1\}^n$, thus indicating a negative result.

Alternatively, an adversary might wish to compromise device A and frame device B for any malicious traffic originating from A:

1. Adversary can swap the device log details of devices A and B, such that EDICT will return the taxonomy and "first seen" timestamp of device B when queried for device A, or

2. Adversary can manually set 1+ bits in the Bloom filter to 0 for device A and set all bits for device B to 1.

To address these vulnerabilities, EDICT must be able to ensure the integrity of its logs. There are various approaches, each with strengths and weaknesses. First, whenever EDICT writes a log to disk (on each novel device connection or at the end of a sublog time period), it could compute a cryptographically-secure hash value for that log. This hash value could be stored in the cloud or locally. Both approaches are still vulnerable to an availability attack, where the attacker simply deletes the logs (or the hashes, if stored locally). This motivates the second approach, where EDICT stores its logs externally, either on the home network (e.g., the user's desktop computer) or in the cloud. This could be used to backup a copy of the logs for security concerns or to address the memory concerns discussed in section 4.2.1.2. Because of the high availability requirements for this approach, a cloud-based architecture is preferred; section **??** will elaborate.

An adversary can also prevent EDICT from identifying communications post-compromise. This can be done by simply disabling EDICT, directly attacking EDICT

when queried for device A's details. This is a false negative error.

as with pre-compromise communications, or attacking the router's underlying architecture. There is a wide variety of methods an adversary could employ to achieve the latter. For example, an adversary could corrupt the router's NAT functionality, such that it either randomizes the source port (for IPv4) or source IP address (for IPv6), thus anonymizing to EDICT all communications. While this could prevent any responses to the traffic from successfully making it to the source, many types of malicious traffic - e.g., a UDP-based DDoS against the DNS - do not require bidirectional communication for success.

To fix this problem at the router, the router's OS needs to be fundamentally secure. Routers must be able to read, modify, drop, and forward packets to perform routing, so fixing this problem is not as easy as removing or securing individual functionalities. Broadly, this has three steps: one, secure the operating system, removing vulnerabilities, etc.; two, prevent the OS from being compromised during use, by requiring strong authentication mechanisms, etc.; and three, prevent an attacker from sidestepping the OS's protections, by using some form of software integrity checking such as a Trusted Platform Module (TPM).[6]

Additionally, end-to-end cryptographic methods can ameliorate many of the consequences of router insecurity, but they are unable to directly ensure EDICT is able to identify devices. By authenticating one another, the source and destination can be assured that routers have not redirected their traffic to allow for an adversary to masquerade as one of the parties. And, by encrypting their traffic, the parties can ensure that routers, or other parties in the middle of the communications path, are unable to read the contents of these communications. However, because these methods are end-to-end, the router, and thus EDICT, does not benefit from any provided assurances. While there might be room for a host-to-router communication protocol

[6]A Trusted Platform Module is a hardware chip used for cryptographic functions such as secure key storage or hardware verification. For high security applications, these functions are shifted from the CPU, where they traditionally are accomplished through software, into the TPM, in order to protect the results from corruption by malware. In this use case, the TPM would store a hash of the router's operating system from a known good state and, prior to each boot of the OS, recompute this hash; if the computed hash does not match the stored hash, the system is determined to have been corrupted or compromised, and the TPM will prevent the OS from booting. Google's home router, the OnHub, uses a TPM to secure its OS [59].

to enable a higher level of assurance, no such method currently exists.

Conversely, router compromise has little impact on EDICT's privacy protections. While an adversary with access to the router would be able to read the stored device and connection logs, the adversary would only be able to determine what devices have communicated with the router. While an adversary gaining knowledge of which devices communicated at what times is not ideal, it is insufficient for a true breach of users' privacy, as the adversary would have no knowledge of the destination or content of any communications. Moreover, the logs could be locally encrypted, thus preventing even this small privacy concern. However, EDICT has not taken this approach, as local encryption would require an authenticated login process to identify devices, which complicates use.

Chapter 6

Conclusions

The rise of the Internet of Things challenges malware remediation on home networks. Millions of devices, from refrigerators to lightbulbs, will be connected to the Internet and open to exploitation and corruption. This exacerbates existing threats from bots, who send the majority of spam email, contribute to crippling DDoS campaigns against major financial and political institutions, and breach millions of individuals' privacy. Worse, the traditional answer of antivirus is unsuited to the IoT environment and insufficient to address this problem, as are alternative schemes such as law enforcement campaigns or defensive strategies.

However, increasing coordination between ISPs and their customers can facilitate the remediation of infected devices at low cost to all stakeholders and with no requirements for coordination between ISPs and no significant legal or privacy concerns. Importantly, this benefits both ISPs and their customers. ISPs gain a more secure, higher performance network, as well as the associated intangible reputational benefits, increasing their profits. Customers, too, increase security: Their devices can be more readily remediated, lowering the costs of malware. While this will not entirely solve the problem of malware, it will protect individuals and the Internet from much of the negative consequences. It should also increase IoT security awareness, which in turn may incentivize the development of legitimate remediation capability on IoT devices.

Because of these benefits, many ISPs have implemented customer notification sys-

tems, but the residential source identification problem stymies malware remediation in expanding home networks. The privacy protections afforded by NAT and IPv6 Privacy Extensions limit an ISP's ability to trace back a communication to a specific device. This is for good reason: individuals' Internet behaviors should remain unknown to an ISP.

EDICT solves the residential source identification problem. It logs each flow leaving the home network, allowing the customer to identify any malicious traffic, given notification from their ISP. Their ISP, in turn, is able to leverage their position and economies of scale to efficiently and effectively detect malicious traffic. EDICT further logs all devices as they connect to the router and identifies the manufacturer and model, enabling user-friendly identification of compromise. By designing EDICT around a series of scalable Bloom filters, EDICT is able to efficiently run on modern routers and protect users' privacy.

Importantly, EDICT provides this benefit even when individually adopted. Unlike solutions to the broader IP traceback problem, EDICT does not require inter-ISP coordination for successful identification; it relies on notification that can be accomplished with only coordination between an ISP and its customers. Thus, a single ISP implementing notification can see benefit, and a single customer adopting EDICT can see benefit.

EDICT is readily implementable on modern routers, as it has low requirements for memory and no significant impact on network performance. When tested on a network designed to simulate containing an aggressive compromised device, EDICT had only a minor impact on the router's memory usage and no significant impact on the router's CPU utilization or the network's throughput.

Were EDICT widely adopted, malware authors may adapt by designing malware that spoofs MAC addresses. There are other vulnerabilities in the architecture, specifically related to spoofed IPv6 addresses and source ports, but they are mitigated by EDICT's design: It will return a list of all matching queries, thus affording a spoofing device only a small anonymity set, instead of true anonymity. However, by spoofing MAC addresses, a compromised device can ensure a legitimate device is blamed for

the traffic. This can only be solved by advanced network authentication methods, including certificate-based and signature-based schemes.

Yet, EDICT is only a single point in a rising argument, that home routers are now the center of a home's digital life. Increasingly, they are being called on for capabilities far beyond the simple routing and switching of earlier models. Certain technologies, such as network storage capability, has become common. Presaging wider adoption, higher-end home routers have introduced additional functionality, from Virtual Private Networks (VPNs) to media servers. Moreover, advanced network management is becoming commonplace, such as the addition of "guest" networks for convenient access or network optimizations for gaming consoles. EDICT, and other router-based security mechanisms such as DNS filtering,[1] are another part of this movement.

6.1 Future Work

In seeking to understand the IoT malware remediation ecosystem and solve the identified residential source identification problem, this thesis evoked a series of research questions. Several of these were answered in the course of this research, but many more remained outside the scope of this effort. Broadly, I divide these topically: ISP security incentives, IoT malware, the future of the IoT, and additions to EDICT.

6.1.1 ISP Security Incentives

EDICT was designed with policy assumptions regarding the incentive structure surrounding ISPs. Broadly, this assumption is simple: ISPs can economically benefit from detecting malicious traffic and notifying their customers. In order to investigate this assumption, research could measure either the direct impact of this security on ISPs' costs or examine alternative economic levers relating security to ISPs' revenue.

While the direct link is straightforward, it itself rests on an assumption: customer

[1]By filtering DNS requests, a router can deny access to unwanted domain names. Typical use cases include the filtering of adult material and known sources of spam email or advertisements.

notification increases security. That is, decreased malicious traffic will decrease costs, but will customer notification decrease malicious traffic? Initial research hints this is true [56], but more data - especially from ISPs - is needed. Tangentially, this research effort could also answer other questions: Do customers follow notifications of malicious activity? How can these notifications be improved? Do notifications cause improvement primarily by lowering the time to remediation (assuming the problems would be found eventually) or by alerting customers to compromise that would go otherwise undetected? Indirectly, ISPs' revenue might be impacted in several ways. First, as consumers become more security-conscious, their choice in ISP might become a reflection of their belief in the ISP's security. Thus, a secure ISP can attract additional customers, raising revenue. Second, ISPs' "layer 8" and "layer 9"[2] connections might provide a mechanism linking security and cost. Network operators' informal interconnections might be able to benefit ISPs that are seen as more secure or better stewards of the Internet (for example, by proactively handling sources of malicious traffic on their network that are victimizing other ISPs' clients). Alternatively, the peering or transit agreements between ISPs might provide a coercive mechanism to encourage ISPs to behave in a pro-security manner. While inter-ISP agreements are notoriously undisclosed, previous research on interdomain routing has succeeded in surveying network operators [32], suggesting this may be a valid research design for this question.

Finally, this analysis of ISPs has assumed a traditional model of the ISP. It has eschewed a critical modern development, Content Delivery Networks (CDNs). Given their increasing responsibility for serving data to customers, CDNs have an important role to play in any anti-malware strategy. Like an ISP, they can be positioned in front of either source or destination, but as an important difference, most CDNs operate as a single hop: Content providers pay the CDN to deliver content directly to customers. Without the ISP-to-ISP-to-ISP hopping of the "traditional" Internet, alternative schemes may be possible. Additionally, because CDNs can spread widely

[2] "Layer 8" and "layer 9" refer to fictional additional layers to the 7-layer OSI model: people and organizations.

across the Internet - though interconnection with disparate ISPs - they could play an important role in alternative detection and notification schemes.

6.1.2 IoT Malware

IoT malware is very much in its infancy. With uncertain development ahead for IoT malware, this research raised two primary questions. First, will IoT malware follow a similar trajectory to "traditional" malware? By this, I refer to two phenomena that have targeted traditional devices: malware being spread by social engineering, instead of exclusively self-replicating worms reliant on system or protocol vulnerabilities, and ransomware. Given the difficulty with communicating trust to a user of an IoT device, both approaches are worrisome. Additionally, ransomware is especially frightening, as home IoT devices may play an important role that traditional devices never did: What if a front door is infected with ransomware and refuses entrance to the homeowner without payment? Second, will IoT malware continue to focus on breadth instead of depth? Current strains of malware have focused on single features - use this device to send malicious traffic, access this camera, etc. As a point of comparison, Zeus, a modern and full-featured strain of "traditional" malware, allows its controllers to send malicicous traffic, use devices as a proxy, and steal sensitive and/or financial information. Additionally, the most prevasive malware strains on today's IoT have lived on device memory, instead of attempting to gain persistance[3] by infecting flash memory. There are many possible explanations: IoT devices rarely reboot, so in-memory malware is "enough," or perhaps in-memory is simply easier.

6.1.3 Future of the IoT

Similarly, the Internet of Things as a whole has an uncertain future. This work has raised a litany of questions. First, will the IoT be relatively homogeneous or heterogeneous - will there be a wide range of operating systems or, perhaps led by a natural

[3]Persistance, in this use, refers to malware's ability to resume operations despite the device rebooting, losing power, etc. This is typically done by infecting the permanent, flash storage. On a traditional device, this would mean writing the malware to the hard drive and including a component, such as a malicious hardware driver, that allows it to be reloaded when the machine boots up.

oligopoly of manufacturers, few? Second, will the devices themselves be intended as single purpose or multipurpose? Securing a multi-purpose device is significantly more difficult, but the added capability brings additional security benefits - such as the possibility for antivirus. While there are research efforts underway to develop lightweight, secure operating systems engineered for the IoT, the future is still uncertain. Additionally, there is a gap between the intent of the device and the underlying OS: a "single purpose" smart lightbulb might, at its foundation, have a multi-purpose OS. This is the worst case, as single purpose devices are less likely to have additional features on the OS disabled or secured. Third, will IoT manufacturers continue to lack adequate incentives for security? While the situation is currently dire, there are several mechanisms for change. Government regulation, perhaps originating from the newly-formed Senate Cybersecurity Caucus [20], might mandate security, or consumer choice might encourage it, perhaps aided by the (TODO: Mudge's lab). Fourth, an outstanding technical question is IoT device authentication. Traditional approaches, including certificates or user-in-the-loop password entry, are unsuitable. Alternative approaches, such as the network-based methods EDICT utilizes, are insufficient for high-integrity applications (e.g., home security).

6.1.4 Additions to EDICT

Lastly, throughout the development of EDICT, I identified several goals for future improvement:

First, how can EDICT operate on low-storage devices? While EDICT is designed to run on general routers, certain future use cases might dictate its use on lower-capability devices - wireless range extenders, dedicated access points for a family of IoT devices, etc. In order to remove much of the burden of storage and querying from these devices, EDICT could utilize cloud storage. Logs could be locally encrypted, stored in the cloud, and then decrypted locally for queries. Alternatively, the logs could be homomorphically or functionally encrypted, allowing the server to maintain user-encrypted logs but return plaintext queries. While this approach allows for the EDICT instance to obviate responsibility for queries and fully protect users' privacy,

additional work is required to integrate this class of cryptography into EDICT. Moreover, all encryption and decryption operations impose computational requirements that may be beyond low-capability devices, thus making off-device storage more useful for long term backup of logs from full-capability routers.

Second, how can EDICT secure its stored logs? As noted in section 5.2.4, EDICT's stored logs are vulnerable on a compromised router. This could be accomplished by securing the logs on the router, perhaps by digitally signing logs with a password-derived key or a key stored on a TPM installed on the router. However, the logs would remain vulnerability to an availability attack (i.e., they could still be deleted). To protect against this threat, the logs could be duplicated off-router, either on another device on the home network or on an Internet server.

Third, how could EDICT work under an alternative coordination scheme, as discussed in section 4.3.2? As noted there, there is a significant trust issue. Absent the existing customer-ISP relationship, a customer may have little reason to trust the notification's validity, thus impeding successful remediation.

Fourth, could EDICT be automatically integrated into the ISP notification process? Currently, EDICT operates as a human-in-the-loop system, where an ISP notifies a customer, and the customer then manually queries EDICT with metadata from that notification. Alternatively, EDICT could be operated remotely by the ISP, wherein instead of notifying the customer, they notify the EDICT instance, which then notifies the customer of the device-specific information (the result of the query to EDICT). This approach would yield significant benefits to the user experience, but additional work is required to ensure the notification channel is secure and resistant to forgery.

Bibliography

[1] *18 U.S. Code § 1030 - Fraud and related activity in connection with computers.* URL: https://www.law.cornell.edu/uscode/text/18/1030 (visited on 05/05/2016).

[2] *Acceptable Use Policy.* URL: http://www.centurylink.com/aboutus/legal/acceptableuseCTL.html (visited on 05/08/2016).

[3] *Acceptable Use Policy for High-Speed Internet.* URL: https://www.xfinity.com/Corporate/Customers/Policies/HighSpeedInternetAUP.html (visited on 05/08/2016).

[4] *Attack Surface Analysis Cheat Sheet.* July 2015. URL: https://www.owasp.org/index.php/Attack_Surface_Analysis_Cheat_Sheet (visited on 04/19/2016).

[5] Philip Auerswald et al. "The Challenge of Protecting Critical Infrastructure". In: *Issues in Science and Technology* XXII.1 (2005). URL: http://issues.org/22-1/auerswald/ (visited on 05/05/2016).

[6] *Australian Internet Security Initiative (AISI).* Australian Communications and Media Authority. URL: http://acma.gov.au/Industry/Internet/e-Security/Australian-Internet-Security-Initiative/australian-internet-security-initiative (visited on 10/09/2016).

[7] *AVG system requirements and supported operating systems.* URL: https://support.avg.com/SupportArticleView?l=en_US&urlname=What-are-AVG-system-requirements-and-supported-opearting-systems (visited on 05/04/2016).

[8] Carlos Baquero, Paulo Sérgio Almeida, and Nuno Preguiça. "Scalable bloom filters". In: (2007). URL: http://repositorium.sdum.uminho.pt/handle/1822/6627 (visited on 08/15/2016).

[9] *Basic Tips and Advice*. Tech. rep. Stop.Think.Connect. URL: https://staysafeonline.org/download/datasets/2144/BasicTipsAndAdvice.pdf.

[10] Andrew Baumann, Marcus Peinado, and Galen Hunt. "Shielding applications from an untrusted cloud with haven". In: *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2014. URL: https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-baumann.pdf?utm_content=buffercbd94&utm_medium=social&utm_source=app.net&utm_campaign=buffer (visited on 01/20/2016).

[11] Robert Beverly and Steven Bauer. "The Spoofer project: Inferring the extent of source address filtering on the Internet". In: *Proceedings of USENIX SRUTI workshop*. 2005. URL: http://static.usenix.org/legacy/events/sruti05/tech/full_papers/beverly/beverly_html/ (visited on 05/04/2016).

[12] Burton H. Bloom. "Space/time trade-offs in hash coding with allowable errors". In: *Communications of the ACM* 13.7 (1970), pp. 422–426. URL: http://dl.acm.org/citation.cfm?id=362692 (visited on 09/08/2016).

[13] Mark Bowden. *Worm: The First Digital World War*. New York: Atlantic Monthly Press, Sept. 2011.

[14] Brian Krebs. *KrebsOnSecurity Hit With Record DDoS*. Krebs on Security. Sept. 16, 2016. URL: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/ (visited on 11/26/2016).

[15] Brownlee, Lisa. *The $11 Trillion Internet Of Things, Big Data And Pattern Of Life (POL) Analytics*. Forbes. URL: http://www.forbes.com/sites/lisabrownlee/2015/07/10/the-11-trillion-internet-of-things-big-data-and-pattern-of-life-pol-analytics/#5db8a5117a46 (visited on 12/10/2016).

[16] Armin Büscher and Thorsten Holz. "Tracking DDoS attacks: insights into the business of disrupting the web". In: *Presented as part of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats*. 2012. URL: https://www.usenix.org/conference/leet12/workshop-program/presentation/buscher (visited on 04/11/2016).

[17] Abhishek Chandra, Weibo Gong, and Prashant Shenoy. "Dynamic resource allocation for shared data centers using online measurements". In: *Quality of Service—IWQoS 2003*. Springer, 2003, pp. 381–398. URL: http://link.springer.com/chapter/10.1007/3-540-44884-5_21 (visited on 05/05/2016).

[18] Yu Chen, Kai Hwang, and Wei-Shinn Ku. "Collaborative detection of DDoS attacks over multiple network domains". In: *Parallel and Distributed Systems, IEEE Transactions on* 18.12 (2007), pp. 1649–1662. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4359399 (visited on 04/13/2016).

[19] Evan Cooke, Farnam Jahanian, and Danny McPherson. "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets." In: *SRUTI* 5 (2005), pp. 6–6. URL: https://www.usenix.org/legacyurl/usenix-sruti-05-technical-paper-10 (visited on 04/11/2016).

[20] *Cybersecurity - Mark R. Warner*. URL: http://www.warner.senate.gov/public/index.cfm/cybersecurity (visited on 11/27/2016).

[21] Dave Plonka. "Measurement and Analysis for Internet of Things". Internet Engineering Task Force (IETF) 96, Measurement and Analysis for Protocols Research Group (MAPRG). July 18, 2016. URL: https://www.ietf.org/proceedings/96/minutes/minutes-96-maprg.

[22] *Dependability Assurance Framework For Safety-Sensitive Consumer Devices Request For Proposal*. Tech. rep. sysa/13-03-20. Needham, MA: Object Management Group, Mar. 2013. URL: http://sysml.org/docs/refs/UML-for-SE-RFP.pdf (visited on 05/08/2016).

[23] Ben Dickson. *4 devices that can help secure your home's IoT*. URL: http://thenextweb.com/insider/2016/01/04/4-devices-that-can-help-secure-your-homes-iot/ (visited on 03/02/2016).

[24] John R. Douceur. "The sybil attack". In: *Peer-to-peer Systems*. Springer, 2002, pp. 251–260. URL: http://link.springer.com/chapter/10.1007/3-540-45748-8_24 (visited on 04/18/2016).

[25] Michel J.G. van Eaten and Johannes M. Bauer. *Economics of Malware: Security Decisions, Incentives and Externalities*. Tech. rep. 2008/1. OECD, May 2008.

[26] W. Eddy. *RFC 4987: TCP SYN Flooding Attacks and Common Mitigations*. Aug. 2007. URL: https://datatracker.ietf.org/doc/rfc4987/ (visited on 05/01/2016).

[27] Gadi Evron. *Dutch ISPs Sign Anti-Botnet Treaty*. Sept. 2009. URL: http://www.i-policy.org/2009/09/dutch-isps-sign-anti-botnet-treaty.html (visited on 05/08/2016).

[28] *GameOver Zeus Botnet Disrupted*. June 2014. URL: https://www.fbi.gov/news/stories/2014/june/gameover-zeus-botnet-disrupted/gameover-zeus-botnet-disrupted (visited on 04/06/2016).

[29] Ofer Gayer, Or Wilder, and Igal Zeifman. *CCTV Botnet In Our Own Back Yard*. Oct. 2015. URL: https://www.incapsula.com/blog/cctv-ddos-botnet-back-yard.html (visited on 05/01/2016).

[30] Denton Gentry and Avery Pennarun. "Passive Taxonomy of Wifi Clients using MLME Frame Contents". In: *CoRR* abs/1608.01725 (2016). URL: http://arxiv.org/abs/1608.01725.

[31] *Gfiber Taxonomy*. Google Git. URL: https://gfiber.googlesource.com/vendor/google/platform/+/master/taxonomy (visited on 10/09/2016).

[32] Phillipa Gill, Michael Schapira, and Sharon Goldberg. "A survey of interdomain routing policies". In: *ACM SIGCOMM Computer Communication Review* 44.1 (2013), pp. 28–34. URL: http://dl.acm.org/citation.cfm?id=2567566 (visited on 11/27/2016).

[33] S. Guha et al. *RFC 5382 - NAT Behavioral Requirements for TCP*. Oct. 2008. URL: https://tools.ietf.org/html/rfc5382 (visited on 03/22/2016).

[34] *Heuristic analysis in Kaspersky Anti-Virus 2013*. Mar. 2013. URL: https://support.kaspersky.com/us/8641 (visited on 05/04/2016).

[35] Kelly Jackson Higgs. *IDS/IPS: Too Many Holes?* July 2006. URL: http://www.darkreading.com/attacks-breaches/ids-ips-too-many-holes/d/d-id/1128186 (visited on 05/04/2016).

[36] Steven Hofmeyr et al. "Modeling Internet-Scale Policies for Cleaning up Malware". en. In: *Economics of Information Security and Privacy III*. Ed. by Bruce Schneier. New York, NY: Springer New York, 2013, pp. 149–170. ISBN: 978-1-4614-1980-8 978-1-4614-1981-5. URL: http://link.springer.com/10.1007/978-1-4614-1981-5_7 (visited on 08/10/2016).

[37] Thorsten Holz et al. "Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm." In: *LEET* 8.1 (2008), pp. 1–9. URL: https://www.usenix.org/event/leet08/tech/full_papers/holz/holz_html (visited on 04/13/2016).

[38] *Home Networking Working Group*. Internet Engineering Task Force. URL: https://tools.ietf.org/wg/homenet/charters (visited on 11/26/2016).

[39] Christian Huitema. "Experience with MAC address randomization in Windows 10". In: *93th Internet Engineering Task Force Meeting (IETF)*. 2015. URL: https://www.ietf.org/proceedings/93/slides/slides-93-intarea-5.pdf (visited on 09/08/2016).

[40] John Iaonnidis and Steven M. Belloven. "Implementing Pushback: Router-Based Defense Against DDoS Attacks". AT&T Labs Research, 2002.

[41] *In the Matter of Promoting and Protecting the Open Internet*. Tech. rep. GN Docket No. 14-28. Federal Communications Commission, Mar. 2015.

[42] *In the Matter of Protecting the Privacy of Customers of Broadband and Other Telecommunications Services*. Apr. 2016.

[43] *Internet Acceptable Use Policy*. URL: http://help.twcable.com/twc_misp_aup.html (visited on 05/08/2016).

[44] *IPv4 Addressing Options*. URL: https://www.arin.net/resources/request/ipv4_countdown.html (visited on 05/04/2016).

[45] Mohammad Karami, Youngsam Park, and Damon McCoy. "Stress Testing the Booters: Understanding and Undermining the Business of DDoS Services". In: *arXiv preprint arXiv:1508.03410* (2015). URL: http://arxiv.org/abs/1508.03410 (visited on 04/06/2016).

[46] Andrey Katsman. *Linux and the Internet of Things*. Sept. 2015. URL: https://www.linuxjournal.com/content/linux-and-internet-things (visited on 05/05/2016).

[47] Vivek Krishnamurthy. "Cloudy with a Conflict of Laws". In: *Berkman Center Research Publication* 2016-3 (2016). URL: http://papers.ssrn.com/sol3/Papers.cfm?abstract_id=2733350 (visited on 04/30/2016).

[48] Maxwell N. Krohn. "Building Secure High-Performance Web Services with OKWS." In: *USENIX Annual Technical Conference, General Track*. 2004, pp. 185–198. URL: http://www0.cs.ucl.ac.uk/staff/B.Karp/gz03/f2007/okws.pdf (visited on 01/20/2016).

[49] Kyle York. *Dyn Statement on 10/21/2016 DDoS Attack*. Dyn Blog. Oct. 22, 2016. URL: https://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/ (visited on 11/26/2016).

[50] Lisa Vaas. *Warning issued over baby monitor, webcam, IoT security... again!* Naked Security. July 19, 2016. URL: https://nakedsecurity.sophos.

com/2016/07/19/warning-issued-over-baby-monitor-webcam-iot-security-again/ (visited on 10/04/2016).

[51] *M3AAWG Bot Metrics Report*. Messaging, Malware and Mobile Anti-Abuse Working Group, Sept. 2014. URL: https://www.m3aawg.org/sites/default/files/document/m3aawg_bot_metrics_report1_2012-2013.pdf.

[52] James Manyika et al. *Unlocking the potential of the Internet of Things*. June 2015. URL: http://www.mckinsey.com/business-functions/business-technology/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world (visited on 05/01/2016).

[53] *McAfee SMB Online Store*. URL: http://www.shopmcafee.com/store/mfesmb/en_US/pd/productID.306911700/categoryID.66300400 (visited on 05/04/2016).

[54] Mark Minasi. *Getting to Know User Account Control*. URL: https://technet.microsoft.com/en-us/library/cc512679.aspx (visited on 04/23/2016).

[55] Jelena Mirkovic, Janice Martin, and Peter Reiher. *A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms*. Technical Report 020018. University of California, Los Angeles, 2001.

[56] Tyler Moore. *Increasing the Impact of Voluntary Action Against Cybercrime*. US Department of Homeland Security Cyber Security Division, Feb. 2015.

[57] T. Narten and R. Draves. *RFC 3041 - Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. Jan. 2001. URL: https://tools.ietf.org/html/rfc3041 (visited on 05/05/2016).

[58] *Network Management Guide*. URL: https://www.verizon.com/about/terms-conditions/network-management-guide (visited on 05/08/2016).

[59] *OnHub Features*. Google. URL: https://on.google.com/hub/features/ (visited on 11/12/2016).

[60] Jay Opperman. *Security Scene: Introducing Constant Guard*. URL: http://corporate.comcast.com/comcast-voices/security-scene-introducing-constant-guard (visited on 05/08/2016).

[61] *Overview of the Internet of things*. Tech. rep. Y.2060. ITU-T, 2008. URL: http://www.cttl.cn/itu/itubz/itut/201203/P020120319759043977765.doc (visited on 04/18/2016).

[62] P. Ferguson and D. Senie. *RFC 2827 - Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*. IETF Network Working Group. May 2000. URL: https://tools.ietf.org/html/rfc2827.html (visited on 12/10/2016).

[63] Anna Pagh, Rasmus Pagh, and S. Srinivasa Rao. "An optimal Bloom filter replacement". In: *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2005, pp. 823–829. URL: http://dl.acm.org/citation.cfm?id=1070548 (visited on 10/07/2016).

[64] Vern Paxson. "An analysis of using reflectors for distributed denial-of-service attacks". In: *ACM SIGCOMM Computer Communication Review* 31.3 (2001), pp. 38–47. URL: http://dl.acm.org/citation.cfm?id=505664 (visited on 04/11/2016).

[65] Abigail Pichel. *Cybercriminals Unleash Bitcoin-Mining Malware*. Oct. 2011. URL: https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/93/cybercriminals-unleash-bitcoinmining-malware (visited on 05/05/2016).

[66] Phillip A. Porras, Hassen Saïdi, and Vinod Yegneswaran. "A Foray into Conficker's Logic and Rendezvous Points." In: *LEET*. 2009. URL: https://www.usenix.org/legacy/event/leet09/tech/full_papers/porras/porras_html/ (visited on 04/13/2016).

[67] Gil Press. *Internet of Things By The Numbers: Market Estimates And Forecasts*. URL: http://www.forbes.com/sites/gilpress/2014/08/22/internet-of-things-by-the-numbers-market-estimates-and-forecasts/#528c5f082dc9 (visited on 04/06/2016).

[68] *Project participants*. Anti-Botnet Advisory Centre. URL: https://www.botfrei.de/en/projektteilnehmer.html (visited on 10/09/2016).

[69] *Researchers Discover Weak Security in Many Home Security Systems*. Feb. 2015. URL: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/researchers-discover-weak-security-in-home-security-systems (visited on 05/05/2016).

[70] Stefan Savage et al. "Practical network support for IP traceback". In: *ACM SIGCOMM Computer Communication Review*. Vol. 30. ACM, 2000, pp. 295–306. URL: http://dl.acm.org/citation.cfm?id=347560 (visited on 04/11/2016).

[71] Scott Charney. *The Need for Global Collective Defense on the Internet*. Microsoft on the Issues. Oct. 5, 2010. URL: https://blogs.technet.microsoft.com/microsoft_on_the_issues/2010/10/05/the-need-for-global-collective-defense-on-the-internet/ (visited on 01/03/2017).

[72] *Security Guidance for Early Adopters of the Internet of Things (IoT)*. Tech. rep. Cloud Security Alliance, Apr. 2015.

[73] Tom Simonite. *The Antivirus Era Is Over*. June 2012. URL: https://www.technologyreview.com/s/428166/the-antivirus-era-is-over/ (visited on 05/04/2016).

[74] Chris Smith. *Wireless router security and hacking: Avast 2015 proposes new defenses*. URL: https://bgr.com/2014/11/05/wireless-router-security-and-hacking/ (visited on 05/08/2016).

[75] Alex C. Snoeren et al. "Single-Packet IP Traceback". In: *ACM SIGCOMM Computer Communication Review*. Vol. 31. ACM, 2001, pp. 3–14. URL: http://dl.acm.org/citation.cfm?id=383060 (visited on 01/20/2016).

[76] *Software Patches & OS Updates*. URL: https://ist.mit.edu/security/patches (visited on 04/19/2016).

[77] *Spam Zombies Letter*. URL: https://www.ftc.gov/sites/default/files/documents/public_statements/letters-isps-about-spam-zombies/spam_zombies_letter.pdf.

[78] Steve Bauer. "Net.info: A Provider-to-Subscriber Secure Communication Channel". MIT Communications Futures Project Plenary Session. Oct. 27, 2015. URL: http://cfp.mit.edu/events/27-OCT-2015/27-OCT-2015-Slides.shtml.

[79] Brett Stone-Gross et al. "Your botnet is my botnet: analysis of a botnet takeover". In: *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 635–647. URL: http://dl.acm.org/citation.cfm?id=1653738 (visited on 04/13/2016).

[80] Brett Stone-Gross et al. "The Underground Economy of Spam: A Botmaster's Perspective of Coordinating Large-Scale Spam Campaigns." In: *LEET* 11 (2011), pp. 4–4. URL: http://static.usenix.org/events/leet11/tech/full_papers/Stone-Gross.pdf (visited on 04/11/2016).

[81] *Symantec and Norton Security Products Contain Critical Vulnerabilities*. July 2016. URL: https://www.us-cert.gov/ncas/alerts/TA16-187A (visited on 08/09/2016).

[82] *Table of Hardware*. OpenWrt Wiki. URL: https://wiki.openwrt.org/toh/start (visited on 10/07/2016).

[83] *Terms of Service*. URL: https://www.att.com/legal/terms.aup.html (visited on 05/08/2016).

[84] *The Australian Internet Security Initiative: Interviews with Industry Participants*. Australian Communications and Media Authority, Oct. 2015. URL: http://acma.gov.au/~/media/Cyber%20Security%20and%20UCE/Research/pdf/The%20Australian%20Internet%20Security%20Initiativeprovider%20responses%20to%20securitycompromised%20computers.pdf (visited on 10/11/2016).

[85] *The Internet of Things-Ready Infrastructure*. F5. Feb. 2015. URL: https://f5.com/resources/white-papers/the-internet-of-thingsready-infrastructure (visited on 05/05/2016).

[86] Vrizlynn L. Thing, Morris Sloman, and Naranker Dulay. "A Survey of Bots Used for Distributed Denial of Service Attacks". In: *New Approaches for Security, Privacy and Trust in Complex Environments*. Ed. by Hein Venter et al. Vol. 232. Boston, MA: Springer US, 2007, pp. 229–240. ISBN: 978-0-387-72366-2 978-0-387-72367-9. URL: http://link.springer.com/10.1007/978-0-387-72367-9_20 (visited on 04/11/2016).

[87] *Threats Report*. Tech. rep. McAfee Labs, Mar. 2016. URL: http://www.mcafee.com/us/resources/reports/rp-quarterly-threats-mar-2016.pdf (visited on 05/04/2016).

[88] *Title III of The Omnibus Crime Control and Safe Streets Act of 1968*. U.S. Department of Justice. URL: https://it.ojp.gov/PrivacyLiberty/authorities/statutes/1284 (visited on 10/14/2016).

[89] *Unified Component Model for Distributed, Real-Time and Embedded Systems Request For Proposal*. Tech. rep. mars/2013-09-10. Needham, MA: Object Management Group, Sept. 2013.

[90] Mathy Vanhoef et al. "Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms". In: ACM Press, 2016, pp. 413–424. ISBN: 978-1-4503-4233-9. DOI: 10.1145/2897845.2897883. URL: http://dl.acm.org/citation.cfm?doid=2897845.2897883 (visited on 11/14/2016).

[91] Luis Von Ahn et al. "CAPTCHA: Using hard AI problems for security". In: *Advances in Cryptology—EUROCRYPT 2003*. Springer, 2003, pp. 294–311. URL: http://link.springer.com/chapter/10.1007/3-540-39200-9_18 (visited on 05/01/2016).

[92] Robert NM Watson et al. "Capsicum: Practical Capabilities for UNIX." In: *USENIX Security Symposium*. 2010, pp. 29–46. URL: http://static.usenix.org/legacy/events/sec10/tech/full_papers/Watson.pdf (visited on 01/20/2016).

[93] *Webroot Secures the IoT by Protecting Critical Devices, Gateways and Systems*. Sept. 2015. URL: http://www.darkreading.com/analytics/-webroot-secures-the-iot-by-protecting-critical--devices-gateways-and-systems/d/d-id/1322013 (visited on 05/05/2016).

[94] Arne Welzel, Christian Rossow, and Herbert Bos. "On measuring the impact of DDoS botnets". en. In: ACM Press, 2014, pp. 1–6. ISBN: 978-1-4503-2715-2. DOI: 10.1145/2592791.2592794. URL: http://dl.acm.org/citation.cfm?doid=2592791.2592794 (visited on 04/11/2016).

[95] *What is a firewall?* Nov. 2013. URL: https://kb.iu.edu/d/aoru (visited on 05/05/2016).

[96] *What is an intrusion detection system?* URL: https://www.paloaltonetworks.com/documentation/glossary/what-is-an-intrusion-detection-system-ids (visited on 05/05/2016).

[97] *What is an intrusion prevention system?* URL: https://www.paloaltonetworks.com/documentation/glossary/what-is-an-intrusion-prevention-system-ips (visited on 05/05/2016).

[98] *What is LAN sync?* Dropbox. URL: https://www.dropbox.com/help/137 (visited on 12/18/2016).

[99] *What Is Network Load Balancing? Network Load Balancing (NLB)*. Mar. 2003. URL: https://technet.microsoft.com/en-us/library/cc779570(v=ws.10).aspx (visited on 05/05/2016).

[100] Yang Xiang, Ke Li, and Wanlei Zhou. "Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics". In: *IEEE Transactions on Information Forensics and Security* 6.2 (June 2011), pp. 426–437. ISSN: 1556-6013, 1556-6021. DOI: 10.1109/TIFS.2011.2107320. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5696753 (visited on 04/11/2016).

[101] Abraham Yaar, Adrian Perrig, and Dawn Song. "Pi: A Path Identification Mechanism to Defend against DDoS Attacks". In: 2003.

[102] Bennet Yee et al. "Native client: A sandbox for portable, untrusted x86 native code". In: *Security and Privacy, 2009 30th IEEE Symposium on*. IEEE, 2009, pp. 79–93. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5207638 (visited on 01/20/2016).

[103] *Your Fridge is Full of SPAM: Proof of An IoT-driven Attack*. URL: https://www.proofpoint.com/us/threat-insight/post/Your-Fridge-is-Full-of-SPAM (visited on 04/06/2016).