AdapSan: Adaptive Input Sanitization in Medical Systems with eBPF

Sinyin Chang[*]
National Taiwan University
Taipei, Taiwan
b10902087@ntu.edu.tw

Ao Li
Washington University in St. Louis
St Louis, Missouri, USA
ao@wustl.edu

Evin Jaff
Washington University in St. Louis
St Louis, Missouri, USA
evin@wustl.edu

Yuanhaur Chang
Washington University in St. Louis
St Louis, Missouri, USA
c.yuanhaur@wustl.edu

Jinwen Wang
Washington University in St. Louis
St Louis, Missouri, USA
jinwen.wang@wustl.edu

Ning Zhang
Washington University in St. Louis
St Louis, Missouri, USA
zhang.ning@wustl.edu

Hsu-Chun Hsiao
National Taiwan University
Taipei, Taiwan
hchsiao@csie.ntu.edu.tw

Abstract

The current state of healthcare is challenged by the widespread use of legacy systems, which often lack the advanced security features necessary to combat modern cyber threats. However, our investigation reveals that hospitals are often reluctant to upgrade their systems, which is primarily due to the following factors: low tolerance to service downtime and patch errors, platform diversities, and the resource constraints of embedded medical devices.

Motivated by these challenges, we propose AdapSan, a framework that allows users to dynamically deploy input sanitization schemes for kernel according to the risk profiles of the target system. AdapSan utilizes eBPF as the key technique to enable dynamic reconfiguration of defense strategies, thereby avoiding downtime. Additionally, it uses eBPF's verification mechanisms to sandbox patch code, minimizing the impact of potentially buggy patches. Furthermore, the platform-agnostic nature of eBPF bytecode and its Just-In-Time (JIT) compilation facilitate the deployment of this solution across various platforms while maintaining the execution speed of native code. AdapSan incorporates two types of insertion schemes, offering a balance between usability and code complexity. Preliminary evaluations of AdapSan indicate that the patching time can be less than 19 milliseconds and the average runtime overhead post-patching can be maintained below 36%.

CCS Concepts

• **Security and privacy → Distributed systems security**; **Domain-specific security and privacy architectures**.

[*]This work was done while Sinyin was with the International Student Research Internship Program at Washington University in St. Louis.

AACD '24, October 14–18, 2024, Salt Lake City, UT, USA.
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1231-9/24/10
https://doi.org/10.1145/3689935.3690397

Keywords

Adaptive defense; Input sanitization; eBPF; Medical security

ACM Reference Format:
Sinyin Chang, Ao Li, Evin Jaff, Yuanhaur Chang, Jinwen Wang, Ning Zhang, and Hsu-Chun Hsiao. 2024. AdapSan: Adaptive Input Sanitization in Medical Systems with eBPF. In *Proceedings of the 11th ACM Workshop on Adaptive and Autonomous Cyber Defense (AACD '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3689935.3690397

1 Introduction

Healthcare organizations and medical systems have become popular targets of cyberattacks in recent years [21, 24]. Since 2020, there has been a 239% increase in large healthcare breaches involving hacking and a 278% increase in ransomware [23]. For example, the recent Black Basta ransomware has disrupted the operation of 140 Ascension hospitals across the U.S. [6]. One culprit of such trends is healthcare organization's reliance on legacy systems. It was found that 83% of imaging computers run an unsupported operating system [5], which is a direct result of modern operating systems being mostly incompatible with healthcare components [16].

The prevalence of legacy medical systems opens up a wide attack surface. However, patching healthcare systems is often challenging in light of several operational constraints in the medical environment. A primary concern for upgrading healthcare systems is the expected resource-consuming and disruptive process of patch deployment. Currently, there are no consistent mechanisms to develop and verify the reliability of a patch before deployment, risking downtime due to patch errors [8]. The fact that the medical environment comprises various interconnected systems and devices, each from different vendors and designed to support patients with different conditions, further complicates the development and testing of patches, and is sometimes even infeasible for legacy systems.

Our Solutions. To address the healthcare industry's reliance on legacy systems and adapt to their unique operational requirements, we propose AdapSan, a framework that utilizes extended Berkeley Packet Filter (eBPF) as the core technique for kernel protection.

eBPF provides several key advantages. First, its dynamic instrumentation supports online updates without requiring system reboots, minimizing service downtime [13, 31] and enhancing availability [28, 29]. Second, eBPF's verifier ensures memory safety, improving patch reliability. Third, its just-in-time (JIT) compilation facilitates portability across a wide range of medical systems and devices. Finally, eBPF allows for selective code instrumentation that helps manage performance overhead more effectively. Given the constraints of writing eBPF programs, AdapSan incorporates two schemes, each designed to balance usability and code complexity.

Evaluation and Future Work. We implemented AdapSan and conducted a preliminary evaluation on a Raspberry Pi 3b and a desktop with an Intel i9-12900K processor. The evaluation shows that the patching time can be less than 19 milliseconds, and the average runtime overhead post-patching can be maintained below 36%. In future work, we plan to fully realize AdapSan's potential by enabling more sophisticated use cases and automatic generation of an eBPF program from user-provided patches.

2 Background & Motivation

2.1 State of Hospital Digital Infrastructure

Heterogeneous Hospital Networks. Digital systems in healthcare can be divided into two main categories: IoT Systems and IT systems. Medical IoT systems contain a wide range of medical devices interconnected in a hospital network. Devices such as infusion pumps (38%), imaging systems (5%), patient monitoring (19%), and point-of-care systems (10%) are the most likely to be IoT-enabled [9]. Meanwhile, medical IT systems comprise devices that are often not purpose-built and instead run applications important for hospital management on generic hardware. This includes EHR terminals for retrieving and entering patient data, PACS servers for storing patient imaging data, and HMS systems that manage operations such as discharge paperwork and billing.

Efforts to Secure Medical Systems. Previous efforts to secure medical systems started with non-adversarial safety analysis, such as the introduction of ISO 14971 [1]. Its modern revision provides a general risk management framework for assessing the risks of medical systems based on their ability to harm in the event of malfunction [3]. On the other hand, IEC 80001-1's introduction in 2010 contained more security-specific guidelines for medical device security, proposing a three-phase life cycle and a decommissioning plan in addition to network security recommendations [2]. Later, device-specific guidelines emerged, such as the NIST SP 1800-8 in response to the high adoption of IoT-enabled infusion pumps [22].

Despite the available guidelines, definitive security requirements for medical devices in the U.S. were not established until the passage of Section 3305 in the 2023 federal omnibus, which sets minimum cybersecurity requirements for devices submitted to the FDA, including accelerated premarket applications such as 510(k) [4]. The FDA's draft standard for cybersecurity requires that device manufacturers maintain procedures to provide reasonable security assurance, disclose post-market vulnerabilities, and provide a software bill of materials [12]. MITRE and the FDA have also released a white paper on cybersecurity risks for legacy medical devices that are in use but are considered unable to be patched to standard [20].

However, studies show a poor adoption of these guidelines and security measures in the medical sector. While a growing number of hospitals adopted network segmentation practices such as VLAN, 72% of these mixed medical systems with non-medical ones [5]. Even simple device-level protections, such as firewall and anti-malware software, have not yet been fully adopted, with 11% and 9% of hospitals operating without them, respectively [30]. Some of these oversights could be attributed to underfunded IT and minimal security spending, in addition to technical constraints, as 75% of hospitals believe that they do not spend enough on security, despite a predicted 1.8 million healthcare IT worker shortage in 2021 [30].

2.2 Barriers to Updating Systems

Downtime. Due to the critical nature of medical infrastructure, downtime is required to be extremely minimal. A study of EHR downtime found that a disrupted EHR system can cause the mean time to perform clinical tests to increase by 62% [17]. As a result, system maintenance is required to be short and occur during low-traffic periods. For example, a Cornell hospital issued an emergency bulletin that they would be performing an emergency maintenance update on Epic EHR at 1:00 AM, which would last 3 minutes and would restore app context to users after the update was completed [14].

Patch Verification. Hospitals are also particularly sensitive to downtime caused by incorrect patches that break an application or system. For example, a case study from an emergency diagnostic division details how a security update to Internet Explorer 6 broke a proxy connection to their PACS server that stores images from radiology. The outage caused 10 days of downtime to fix and resulted in a statistically significant decrease in the utilization of workstations to diagnose illnesses [16]. More recently, CrowdStrike pushed a flawed update to a security module that managed named pipe execution in Windows which causes a bad dereference in forbidden memory [8]. The flaw resulted in complete kernel panic and many hospitals were left unable to operate critical systems [7].

Generalizability across Platforms. Medical systems span a wide set of different hardware and software combinations. Among a 2020 Palo Alto Networks survey of imaging systems, 9% of the systems use a custom embedded OS with 9% of systems using a variety of Linux and Unix versions (both supported and unsupported). The report did find a majority 82% of imaging systems using Windows, but this was further segmented between usage of unsupported versions like Windows 7 and XP and supported versions like Windows 8.1 and 10 [5]. A more recent Cynerio study in 2023 presents an even more scattered landscape for all medical IoT where 46% of medical IoT devices run a version of Linux, with the remaining 54% comprised of a mix of proprietary RTOSes, Windows (both embedded and desktop), and other operating systems [9].

Resource Constraints. The market size of medical microcontrollers has been increasing rapidly, projected to reach 75 billion by 2032 [10]. Microcontrollers' healthcare applications mostly include implantable devices such as pacemakers, infusion pumps, CPAP machines, etc., all of which require signal controls to be precise and real-time data processing to ensure timely responses [27]. However, compared to traditional processing units, microcontrollers have minimal memory, storage, and computational power.

3 Threat Model and System Goals

Threat Model. We assume the adversary has arbitrary memory read and write ability in user-space, but cannot alter any data in the kernel memory space. This can be achieved by installing a malicious third-party application [15] or exploiting network vulnerabilities to remotely compromise a user-space application. The primary objective of the adversary is to exploit the kernel vulnerability from user-space to corrupt the kernel's functionality. The goals could be manipulating critical medical device sensor data or tampering with actuation data to cause harm to patients. We assume that the target system's kernel is trustworthy, but may contain vulnerabilities that can potentially be exploited by user-space applications. Since AdapSan leverages eBPF [26] as a substrate, we assume that the eBPF toolchain is trustworthy and free of bugs. Hardware attacks, side-channel attacks, covert-channel attacks, and availability attacks [18] are outside the scope of this work.

System Goals. The design of AdapSan is guided by the following goals motivated by the barriers discussed in Section 2.2.

R1 - No service disruption. Defense should be able to be dynamically adjusted at runtime without having to reboot the system.

R2 - Reliability of patch. Ensuring the logical correctness of a patch is essential but depends heavily on domain knowledge and the semantics of the target application. As a system-level solution, AdapSan focuses on ensuring memory safety for the added code, preventing it from corrupting the kernel.

R3 - Agnostic to Target Platform. Given the diverse platforms used in medical devices, the solution must be compatible across platforms.

R4 - Low runtime overhead. The defense should incur minimal resource costs, introducing only manageable runtime and memory overhead. This is crucial for embedded environments, where most medical IoT devices operate.

4 Design of AdapSan

Overview. AdapSan is designed to deploy input sanitization mechanisms within the kernel at runtime, eliminating the need for a system reboot. The centerpiece of AdapSan is an eBPF-based dynamic instrumentation framework, which inherently satisfies requirement R1. Additionally, AdapSan leverages the eBPF verifier to ensure that newly added patch code is sandboxed, thereby minimizing impacts on other parts of the kernel, which meets R2. eBPF's availability on Linux, which is present in 46% of medical IoT devices, and its platform-independent bytecode, modular design, and uniform interface allow for adaptation on other systems, satisfying R3. Furthermore, dynamic instrumentation enables selective patching of the kernel without maintaining multiple instances, and the JIT compilation of eBPF converts bytecode to native machine code with minimal impact on runtime performance, meeting R4.

The workflow of AdapSan is depicted in Figure 1. (1) A user writes patch code based on the vulnerability report; (2) the user specifies the location in AdapSan to patch; (3) AdapSan compiles the user's code and verifies its correctness; (4) AdapSan notifies the user if the verification fails; (5) otherwise, AdapSan adds the compiled program into the kernel; (6) the kernel then runs the compiled bytecode at the locations predefined by the user.

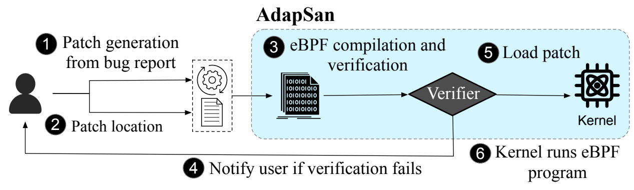
AdapSan
1 Patch generation from bug report
2 Patch location
3 eBPF compilation and verification
Verifier
5 Load patch
Kernel
4 Notify user if verification fails
6 Kernel runs eBPF program

Figure 1: AdapSan workflow.

Dynamic Input Sanitization. AdapSan incorporates two schemes for dynamic input sanitization, each with its own advantages and limitations. The first scheme involves transforming patch code directly into an eBPF program, which is then loaded into the kernel to replace the vulnerable code. This approach leverages the eBPF verifier to ensure the program is sandboxed and thus secure. However, due to eBPF's stringent requirements, the complexity of the eBPF program is limited. Highly complex patches may not pass the verification process, making this method unsuitable for patches with intricate code logic. Additionally, eBPF programs cannot be added at predefined points in the kernel, further limiting their applicability.

The second scheme involves instrumenting reference monitoring code to achieve input sanitization by inserting it into the kernel before the vulnerable code region. This is accomplished using KProbe [25], which allows adding tracing points in most kernel code regions without modifying the kernel itself. It can be used alongside eBPF to add more instrumentation points, enhancing eBPF's flexibility. However, a drawback is that KProbe does not provide security guarantees for the inserted code. This makes it unsuitable for directly deploying patch code without additional measures, such as instrumenting address masking for sandboxing.

Helper Functions. eBPF programs, unlike built-in kernel code, drivers, or loadable kernel modules, do not have access to the entire kernel address space and cannot call arbitrary kernel functions. This limitation can pose challenges for the development of eBPF programs. To facilitate the implementation, we provide specialized helper functions in eBPF tailored to networked medical systems. These helper functions fall into two main categories: (1) Data Inspection and Analysis: These functions inspect and analyze sensor and actuation data communicated by various devices, ensuring secure and reliable communication between medical devices and other systems; (2) Event Timestamping: These functions retrieve precise timestamps for events, which is crucial for monitoring and logging time-sensitive medical data. By using these tailored helper functions, eBPF programs can effectively support the secure and reliable operation of networked medical systems.

5 Preliminary Evaluation

We implemented AdapSan on Linux with kernel version 5.15 and on Raspbian with kernel version 6.6. The core functionalities are implemented as a set of Python scripts. To evaluate AdapSan, we aim to answer the question – *Does AdapSan achieve low downtime?* This will be assessed by measuring the downtime caused by the first scheme. To this end, we use an open-source SparkFun heart rate monitor (HRM) [11] as the evaluation target. The original source code runs on Arduino; we modified the code to run on Linux while maintaining the original code logic. The experiments are conducted on a Raspberry Pi 3b and a PC with an i9-12900K processor. We

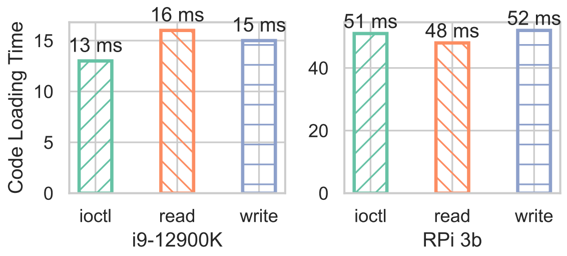

Figure 2: Performance evaluation of AdapSan.

inspected the interface of the HRM application and found that the main components required are the network stack and bus drivers. As such, we assumed vulnerabilities reside in these locations. The sanitization codes are instrumented in ioctl, read, and write syscalls.

Figure 2 presents the evaluation results. The loading time, measured in milliseconds, remains within 16 milliseconds on i9-12900k and 52 milliseconds on Raspberry Pi 3b. This duration is generally within acceptable limits for medical device deadlines. The loading time may vary with the size of the eBPF program, but this accounts for a relatively small part of the overall latency and does not exceed 100 ms in our experiments. In terms of runtime overhead, intercepting the target system calls and performing reference monitor checks incur a higher cost compared to static instrumentation techniques. In our experiments, this overhead can reach up to 36% because the target workload is relatively simple such that its baseline execution time is small. However, the execution time for each check remains at the microsecond level, making the overhead tolerable for scenarios where I/O interactions are infrequent.

6 Limitations and Future Work

AdapSan faces several limitations in terms of compatibility, capability, and usability. First, it relies on the eBPF infrastructure, which may not be available on legacy systems. However, as eBPF becomes more widely supported in modern systems, this issue is expected to diminish [19]. Second, the protection scope of AdapSan is confined to the kernel space. We argue that kernel code is particularly critical in medical IoT environments, as it primarily interacts with peripherals and its vulnerabilities are more challenging to address in practice. Extending protection to user-space programs is left for future work. Finally, the programmability of eBPF poses usability challenges, as converting patch code into eBPF programs requires significant effort. Future work will focus on enabling AdapSan to support more advanced use cases and automatically generate application- and environment-specific patches.

Acknowledgment

Hsu-Chun Hsiao was supported in part by the National Science and Technology Council of Taiwan under grants NSTC 112-2223-E-002-010-MY4 and NSTC 113-2634-F-002-001-MBK. Sinyin Chang was supported in part by Washington University in St. Louis.

References

[1] 1998. Medical devices - Risk Management. https://www.iso.org/standard/25799.html
[2] 2010. IEC 80001-1:2010: Application of risk management for IT-networks incorporating medical devices.
[3] 2019. Medical devices – Application of risk management to medical devices. https://www.iso.org/standard/72704.html
[4] 2023. Consolidated Appropriations Act, 2023. Public Law No: 117-328. https://www.congress.gov/117/bills/hr2617/BILLS-117hr2617enr.pdf
[5] Unit 42. 2020. 2020 Unit 42 IoT Threat Report. https://unit42.paloaltonetworks.com/iot-threat-report-2020/
[6] Christine Barry. 2024. Black basta's nasty tactics: Attack, assist, attack. https://blog.barracuda.com/2024/05/18/black-basta-nasty-tactics
[7] David Cox. 2024. Hospitals Around the World are Struggling after the Great IT Meltdown. https://www.wired.com/story/hospitals-crowdstrike-microsoft-it-outage-meltdown/
[8] Crowdstrike. 2024. Technical Details: Falcon Content Update for Windows Hosts. https://www.crowdstrike.com/blog/falcon-update-for-windows-hosts-technical-details/
[9] Cynerio. 2022. Cynerio IoT report. https://www.cynerio.com/landing-pages/the-state-of-healthcare-iot-device-security-2022
[10] Aarti Dhapte. 2024. Medical microcontrollers market size, Share report and trends 2032. https://www.marketresearchfuture.com/reports/medical-microcontrollers-market-12610
[11] SparkFun Electronics. 2024. AD8232 Heart Rate Monitor. https://github.com/sparkfun/AD8232_Heart_Rate_Monitor/tree/master. Accessed: 2024-07-23.
[12] Food and Drug Administration. 2024. Select Updates for the Premarket Cybersecurity Guidance: Section 524B of the FD&C Act.
[13] Yi He, Zhenhua Zou, Kun Sun, Zhuotao Liu, Ke Xu, Qian Wang, Chao Shen, Zhi Wang, and Qi Li. 2022. RapidPatch: firmware hotpatching for Real-Time embedded devices. In *31st USENIX Security Symposium.* 2225–2242.
[14] Weill Cornell Hospital IT. 2022. 12/17: Epic system update - Emergency Maitenance. https://its.weill.cornell.edu/alerts/1217-epic-system-update
[15] Piergiorgio Ladisa, Henrik Plate, Matias Martinez, and Olivier Barais. 2023. Sok: Taxonomy of attacks on open-source software supply chains. In *2023 IEEE Symposium on Security and Privacy (SP).* IEEE, 1509–1526.
[16] Adam B. Landman, Saira S. Takhar, S. L. Wang, A. Cardoso, J. M. Kosowsky, Ali S. Raja, Ramin Khorasani, and Eric G. Poon. 2013. The hazard of software updates to clinical workstations: a natural experiment. *Journal of the American Medical Informatics Association* 20 (2013). https://doi.org/10.1136/amiajnl-2012-001494
[17] Ethan Larsen, Daniel Hoffman, Carlos Rivera, Brian M. Kleiner, Christian Wernz, and Raj M. Ratwani. 2019. Continuing Patient Care during Electronic Health Record Downtime. *Applied Clinical Informatics* 10, 3 (2019), 495–504. https://doi.org/10.1055/s-0039-1692678
[18] Ao Li, Marion Sudvarg, Han Liu, Zhiyuan Yu, Chris Gill, and Ning Zhang. 2022. Polyrhythm: Adaptive tuning of a multi-channel attack template for timing interference. In *2022 IEEE Real-Time Systems Symposium (RTSS).* IEEE, 225–239.
[19] Microsoft. 2021. Making eBPF Work on Windows. https://opensource.microsoft.com/blog/2021/05/10/making-ebpf-work-on-windows/. Accessed: 2024-09-07.
[20] MITRE. 2023. Next Steps Toward Managing Legacy Medical Device Cybersecurity Risks. https://www.mitre.org/news-insights/publication/next-steps-toward-managing-legacy-medical-device-cybersecurity-risks
[21] The Hacker News. 2024. NextGen Healthcare Mirth Connect Under Attack - CISA issues urgent warning. https://thehackernews.com/2024/05/nextgen-healthcare-mirth-connect-under.html
[22] Gavin O'Brien, Sallie Edwards, Kevin Littlefield, Neil McNab, Sue Wang, and Kangmin Zheng. 2018. *Securing Wireless Infusion Pumps in Healthcare Delivery Organizations.* Special Publication (NIST SP) 1800-8. National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.SP.1800-8
[23] U.S. Department of Health and Human Services. 2023. HHS' office for civil rights settles ransomware cyber-attack investigation. *HHS.gov.* https://www.hhs.gov/about/news/2023/10/31/hhs-office-civil-rights-settles-ransomware-cyber-attack-investigation.html
[24] Emily Olsen. 2024. Ascension says cyberattack may have compromised protected health data. https://www.cybersecuritydive.com/news/ascension-cyberattack-health-data-exposed/718978/
[25] Marina Papatriantafilou and Philippas Tsigas. 2000. KProbe: A low overhead kernel probe tool. In *Proceedings of the 14th international parallel and distributed processing symposium.* IEEE, 169–178.
[26] Alexei Starovoitov et al. [n. d.]. eBPF: Extended Berkeley Packet Filter. https://ebpf.io. Accessed: July 20 2024.
[27] Vemeko. 2024. Microcontroller Applications in Medical Devices. https://www.vemeko.com/blog/microcontroller-applications-in-medical-devices.html
[28] Jinwen Wang, Ao Li, Haoran Li, Chenyang Lu, and Ning Zhang. 2022. Rt-tee: Real-time system availability for cyber-physical systems using arm trustzone. In *2022 IEEE Symposium on Security and Privacy (SP).* IEEE, 352–369.
[29] Jinwen Wang, Yujie Wang, and Ning Zhang. 2023. Secure and timely gpu execution in cyber-physical systems. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security.* 2591–2605.
[30] Liat Wasserman and Yair Wasserman. 2022. Hospital cybersecurity risks and gaps: Review (for the non-cyber professional). *Frontiers in Digital Health* 4 (2022), 862221. https://doi.org/10.3389/fdgth.2022.862221
[31] Yuhao Wu, Jinwen Wang, Yujie Wang, Shixuan Zhai, Zihan Li, Yi He, Kun Sun, Qi Li, and Ning Zhang. 2024. Your Firmware Has Arrived: A Study of Firmware Update Vulnerabilities. In *33rd USENIX Security Symposium.* 5627–5644.