FINITE STATE

Finite State Supply Chain Assessment

Huawei Technologies Co., Ltd.

CONTENTS

EXECUTIVE SUMMARY 1

KEY FINDINGS 3

RECENT SECURITY CONCERNS REGARDING HUAWEI 5

Known Security Concerns 5

Dutch General Intelligence and Security Service Investigation 5

African Union Alleged Data Exfiltration 5

Vodafone Backdoor 5

Windows Kernel Driver Vulnerability 5

Session Hijack, Heap Overflow and Stack Overflow Vulnerabilities 6

UK HCSEC Reports 6

Chinese National Intelligence Law of 2016 6

THE 5G THREAT LANDSCAPE 8

Why 5G is different than 4G 8

5G Network Architecture 9

5G Network Security Challenges 10

5G Network Suppliers 11

Huawei 5G Products and Services 11

SUPPLY CHAIN SECURITY CHALLENGES 12

Modern Electronics Supply Chains 12

The Supply Chain Attack Surface 13

Hardware Backdoors 13

Firmware and Software Backdoors 14

Compromised Firmware & Software Updates 15

Rogue Service Providers 15

Approaches to Supply Chain Security 15

Best Practices and Recommendations 16

Know Your Vendors 16

Leverage Your Buying Power 16

Verify Everything 16

Collaborate on Remediation 16

Transparency Leads to Better Security 16

RISK ASSESSMENT METHODOLOGY 17

Unpacking Firmware 17

Analyzing Firmware 17

Outdated Components 17

Presence of known vulnerabilities 17

Hard-coded credentials and cryptographic material 18

Code hygiene and safety 18

Automated vulnerability discovery 18

Similarity and Code Genealogy 18

Quality change over time 18

Limitations of the Approach 18

Firmware Analysis at Scale 18

Huawei Firmware Analysis 19

USE OF OUTDATED COMPONENTS 20

Results Summary 20

Detailed Example 20

PRESENCE OF KNOWN VULNERABILITIES 23
Discovering Known Vulnerabilities 23
Results Summary 23
CVEs per Firmware 23
CVE Severity 24
Reported vs. Derived CVEs 24
Detailed Example 25
UNDOCUMENTED AND HARD-CODED CREDENTIALS 26
Results Summary 26
Detailed Example 27
DEFAULT AND HARD CODED CRYPTOGRAPHIC KEYS 29
Results Summary 29
Detailed Example 30
SECURE CODING PRACTICES 31
Results Summary 31
Detailed Example 32
VULNERABLE CODE CONDITIONS 33
Results Summary 34
Example Unsafe Function Wrappers 34
Comparing Safe and Unsafe Function Usage 35
QUALITY CHANGE OVER TIME 37
Results Summary 37
CASE STUDY: HUAWEI CE12800 vs. JUNIPER EX4650 vs. ARISTA 7280R 39
Methodology 39
Results Summary 41
CONCLUSIONS 42
Huawei Devices Come with Serious Technical Security Risks 42

Supply Chain Risk Management is More Than Just Technical Risks 42
Managing the Risk 42
Transparency Leads to Better Security 43
STEPS TO MITIGATE YOUR RISK 44
1. Implement a True Supply Chain Security Program 44
Know Your Vendors 44
Leverage Your Buying Power 44
Verify Everything 44
2. Identify 44
3. Mitigate 45
4. Detect 46
5. Respond 46
ABOUT FINITE STATE 47
APPENDIX A: DEVICES TESTED 48

> *We will support and adopt any internationally agreed standard or best practice for cyber security in its broadest sense; we will support any research effort to improve cyber defences; we will continue to improve and adopt an open and transparent approach enabling governments to review Huawei's security capabilities, and finally, as we have done to date, we warmly welcome the assistance from our customers in enhancing our processes, our technology, and our approach to cyber security so that we can provide even greater benefits to them and their customers.*
>
> — John Suffolk, Global Cyber Security and Privacy Officer, Huawei

> *The escalation of the national security debate around Huawei has caught a number of 5G enthusiasts off guard. The United States, Australia, New Zealand, Japan and the Czech Republic, among others, have imposed restrictions on the use of Huawei 5G solutions over national security concerns; much of Europe is pondering whether to follow suit. Summed up, the nations' worries are rooted in the ties between Chinese communications technology companies and its intelligence services, reinforced by China's political and legal environment requiring cooperation with intelligence agencies. Perceived or real, fears persist that adopting Huawei 5G technology will introduce a critical reliance on equipment that can potentially be controlled by the Chinese intelligence services and the military in peacetime and in crisis.*
>
> — NATO Cooperative Cyber Defence Centre of Excellence (CCDOE) , 2019

EXECUTIVE SUMMARY

5G promises to usher in the next generation of consumer, enterprise, and industrial technology. Among the other profound benefits it offers the global economy, 5G will realize the vision and potential of the Internet of Things (IoT). Advancements in 5G technology are expected to enable exponential growth in accessible autonomy and smarter infrastructure that will likely become foundational to our way of life over the next decade and beyond.

However, increased reliance on new technologies brings with it new threats. The possibility of a smart city shutting down, autonomous vehicles crashing, or factories going dark due to a cyber attack is a frightening proposition. 5G technology is a complex system involving hundreds of vendors, winding global supply chains, and the gamut of security threats. Thus, national security, global trade, and international competitiveness are all potentially impacted. If suppliers of 5G technology were to have secret or overt access to the infrastructure they are providing, there is considerable concern that they could be persuaded to use that access as leverage in times of peace, or perhaps something far more ominous in times of conflict.

Compounding this concern is the fact that a single company has emerged as the first and most dominant provider in 5G: Huawei Technologies Co. Ltd., commonly referred to as Huawei. The lack of competition in the 5G market has been described by Dr. Ian Levy, Technical Director for the UK National Cyber Security Centre (NCSC): "The market is fundamentally broken. We can't possibly live in a world where only four or five companies provide all the critical infrastructure for a particular sector—that's insane."[1]

The lack of competition in the market and clear dominance of that market by a Chinese company, coupled with economic and national security concerns, have caused policy debates over the implementation of 5G to boil over. Countries are taking measures to limit their risks by doing everything from establishment of security verification centers to outright bans of Huawei products.

That said, much of this policy debate has been missing a key set of facts. There has been an underlying assumption that using Huawei equipment in a 5G network would provide Huawei and/or the Chinese government with access to that network, which could be used to execute espionage or military missions. But this assumption has never been concretely proven.

Huawei itself denies this possibility. As stated by Huawei's Global Cyber Security and Privacy Officer, John Suffolk, "We don't run networks, and because we don't run the network, we have no access to any of the data that is running across that network." [2]

Cybersecurity experts disagree, as discussed in a recent *Lawfare* article: "Whoever provides the technology for 5G networks will be sitting in a position of incredible access and, thus, power. All data sent and received from a mobile device, smart home or even a car will pass through a network built with Huawei devices. These devices will be remotely controlled and updated, leading to exponential vectors of attack."[3]

Without ground-truth data, it is hard to settle this debate. At Finite State, we believe greater transparency leads to better security for everyone, and that, fundamentally, policymakers should be making data-driven decisions about which risks they are, and are not, willing to take.

Finite State's mission is to defend the next generation of networks containing critical IoT devices through unprecedented visibility, proactive risk management, attack detection, and enablement of rapid responses. As a core component, we have developed advanced technology to provide deep visibility into these previously opaque devices. Our platform unpacks and analyzes device firmware at massive scale, enabling proactive risk identification and robust supply chain security, which help rebalance the power for defenders.

[1] https://www.wsj.com/articles/u-k-cybersecurity-official-says-5g-market-is-fundamentally-broken-11559839990

[2] https://www.bbc.com/news/business-48588661

[3] https://www.lawfareblog.com/risks-huawei-risk-mitigation

To that end, we have undertaken a large-scale study of the cybersecurity-related risks embedded within Huawei enterprise devices by analyzing Huawei device firmware at an unprecedented scale.

Finite State's proprietary technology platform uniquely enabled us to conduct a comprehensive, unbiased analysis of the security properties of these devices. Our automated system analyzed more than 1.5 million files embedded within 9,936 firmware images supporting 558 different products within Huawei's enterprise networking product lines. Our analysis looked for risks including hard-coded backdoor credentials, unsafe use of cryptographic keys, indicators of insecure software development practices, and the presence of known and 0-day vulnerabilities.

The results of the analysis show that Huawei devices quantitatively pose a high risk to their users. In virtually all categories we studied, we found Huawei devices to be less secure than comparable devices from other vendors.

Through analysis of device firmware, we discovered that there were hundreds of cases of potential backdoor vulnerabilities – improper default configurations that could allow Huawei or a malicious attacker to covertly access a user's device. These vulnerabilities manifested in the form of hard-coded, default user accounts and passwords, and several types of embedded cryptographic keys.

The study also found that each Huawei device had a large number of *known* vulnerabilities associated with the third-party and open-source libraries embedded within the firmware. On average, there were 102 known vulnerabilities (CVEs) associated with each firmware, a significant percentage of which were rated as high or critical in their severity.

By analyzing the embedded binary code, configuration files, and libraries, our system was also able to discern the extent to which Huawei is prioritizing security within their software development practices. In most modern software engineering organizations, standard processes are followed to minimize the number of vulnerabilities that can be introduced into a product. In fact, Huawei has pledged to invest $2B into improved security engineering for their products.[4] Despite these investments, our research uncovered a substantial lack of secure development practices resulting in significant numbers of vulnerabilities. In some cases, engineers chose to use 20-year-old versions of software libraries rather than current, secure alternatives. Huawei engineers wrote insecure functions with misleading names indicating that the function was safe from conditions such as buffer overflows when in fact it was not.

By using advanced binary analysis, we also tested these unsafely built software components for vulnerabilities. Our system found hundreds of potential 0-day vulnerabilities (each of which will undergo additional verification and, if warranted, be properly disclosed to the vendors).

Overall, despite Huawei's claims about prioritizing security, the security of their devices appears to lag behind the rest of the industry. Through analysis of firmware changes over time, this study shows that the security posture of these devices is not improving over time — and in at least one case we observed, it actually decreased. This weak security posture, coupled with a lack of improvement over time, obviously increases security risks associated with use of Huawei devices.

Security should be viewed as a risk management problem, and the goal of this report is to present actual risks clearly, in a format that policymakers can use while the debate continues. Whether those risks were introduced intentionally or accidentally is outside of the scope of a technical assessment, and thus, we cannot, and do not, draw any conclusions relating to intent.

Vulnerabilities exist in every device, but if the users of these devices are unaware, attackers have the advantage. If there is no extensive, scalable review process for devices, their supply chains, and their software, it is more likely that intentional and unintentional backdoors can slip in unnoticed. The findings in this report demonstrate that automated, scalable supply chain security reviews are possible, and when implemented properly and continuously against devices and their software updates, they can be a key factor in building out a security program.

Ultimately, the decision on whether to use Huawei devices will come down to individual risk tolerances and plans to manage that risk. Increased transparency into the devices we hope to entrust with our most critical services is paramount to achieving better security for everyone.

[4] http://www.chinadaily.com.cn/a/201902/21/WS5c6e6bb0a3106c65c34eaa2d.html

KEY FINDINGS

Summary

Huawei has been accused of maintaining backdoor access to networks, but until now, little evidence has been available to support or refute those claims. Finite State's automated system analyzed more than 1.5 million unique files embedded within 9,936 firmware images supporting 558 different products within Huawei's enterprise networking product lines — many of which could be used within the core of 5G networks. Our analysis looked for risks including hard-coded backdoor credentials, unsafe use of cryptographic keys, indicators of insecure software development practices, and the presence of known and 0-day vulnerabilities.

The results of the analysis show that Huawei devices quantitatively pose a high risk to their users. In virtually all categories we examined, Huawei devices were found to be less secure than those from other vendors making similar devices.

1. Backdoor Access Vulnerabilities

Out of all the firmware images analyzed, 55% had at least one potential backdoor. These backdoor access vulnerabilities allow an attacker with knowledge of the firmware and/or with a corresponding cryptographic key to log into the device.

- 29% of all devices tested had at least one default username and password stored in the firmware, enabling access to the device if administrators don't change these credentials.

- We identified 76 instances of firmware where the device was, by default, configured such that a root user with a hard-coded password could log in over the SSH protocol, providing for default backdoor access.

- 8 different firmware images were found to have pre-computed `authorized_keys` hard coded into the firmware, enabling backdoor access to the holder of the private key.

- 424 different firmware images contained hardcoded private SSH keys, which can enable a man-in-the-middle to manipulate and/or decrypt traffic going to the device.

2. Pattern of Security Flaws

Huawei devices were shown to have a high number of known security vulnerabilities. Despite the fact that many of these vulnerabilities have been public knowledge for years, Huawei continues to make firmware updates without addressing them. These vulnerabilities increase the likelihood that attackers can compromise these devices.

- On average, Huawei devices had 102 known vulnerabilities inside their firmware, primarily due to the use of vulnerable open-source and third-party components.

- Across the firmware tested, there were 8,826 observations of vulnerabilities with a CVSS score of 10.0, the maximum severity level, indicating serious flaws in the systems.

- One tested device had a total of 1,419 known vulnerabilities in its most recent version of firmware.

3. Highly Insecure Software Development Practices

Despite claims of prioritizing security, we quantitatively demonstrate that Huawei engineers systematically made poor security decisions in building the devices we tested. This weak security engineering significantly increases the potential for serious vulnerabilities.

- Despite being a default compiler option, less than half of the binaries encountered used security features such as ASLR, DEP, and StackGuard.

- Huawei practices abysmal software configuration management as demonstrated by their use of 79 distinct versions of the OpenSSL library across their most recent firmware releases. In some cases, Huawei used 10-year-old versions of libraries containing dozens of vulnerabilities rather than selecting newer, more secure options.

- On dozens of occasions, Huawei engineers disguised known unsafe functions (such as memcpy) as the "safe" version (memcpy_s) by creating wrapper functions with the "safe" name but none of the safety checks. This leads to thousands of vulnerable conditions in their code.

- Across 356 firmware images, there are several million calls into unsafe functions. Huawei engineers choose the “safe” option of these functions less than 17% of the time, despite the fact that these functions improve security and have existed for over a decade.
- On average, each binary analyzed had more than 12 possible buffer overflows, each of which are potential 0-day vulnerabilities.
- Security is not improving over time. In at least one instance, security became quantifiably worse for users that patched their devices to the updated version of firmware.

4. Quantitatively Higher Risk than Other Similar Devices

Compared to similar devices from other vendors, we quantitatively demonstrate that Huawei has substantially worse security.

- In analysis across different dimensions of risk categorized by the Finite State Risk Matrix, a Huawei device had the highest risk in six of the nine categories when ranked against comparable Juniper and Arista devices.
- The Huawei device had substantially more known vulnerabilities and 2-8x more potential 0-day vulnerabilities than the other devices.
- The Huawei device was the only device that contained hard-coded default credentials and hard-coded default cryptographic keys.

5. Firmware Security Verification is Possible at Scale

Despite assertions that devices and their firmware updates could not be scalably tested for security properties, we demonstrate that verification can be conducted at scale, enabling increased transparency and security.

- In a matter of hours, the Finite State Platform was able to process and analyze more than 9,936 firmware images comprised of more than 1.5 million unique files.
- Through firmware analysis, the platform was able to uncover deeper vulnerabilities than comparable vulnerability scanning tools.
- By using automated analytical tools, the end users of these devices have a mechanism to enforce security requirements upon their vendors -- ultimately making networks safer for everyone.

RECENT SECURITY CONCERNS REGARDING HUAWEI

Huawei has faced significant criticism relating to security risks posed by its products, leading several western governments – including the United States, Australia, Japan, and New Zealand – to effectively ban Huawei products entirely. Nevertheless, until now, no detailed study of systemic patterns of vulnerabilities in Huawei products has been made available to the public at large.

To provide context on the current debate, this section includes a survey of some of the more prominent historical concerns, and then discusses the legal regime in China that could provide the mechanism to effectuate exploitation of cybersecurity vulnerabilities. Please note that neither the survey of historical concerns nor the legal analysis is intended to be exhaustive.

Known Security Concerns

Dutch General Intelligence and Security Service Investigation

In May 2019, Dutch newspaper *De Volkskrant* reported that Dutch intelligence agency AIVD was made aware of backdoors on Huawei equipment belonging to a Dutch carrier, and that AIVD was determining whether or not those backdoors were used for spying by the Chinese government. Huawei was said to have possible access to customer data from one of the major telecom providers in the Netherlands. It is not clear whether it is Vodafone, Ziggo, T-Mobile, Tele2, or KPN.[5]

African Union Alleged Data Exfiltration

In January 2018, African Union officials told the *Financial Times* that computer systems installed by Huawei in its headquarters had been transferring confidential information daily to servers in China between 2012 and 2017. The data theft was first reported by French newspaper *Le Monde Afrique*, which said AU technicians identified the leak after noticing unusually high levels of data transfer activity each day between midnight and 2am.[6]

Vodafone Backdoor

Vodafone, Europe's biggest phone company, identified hidden backdoors in software inside Huawei products that could have given Huawei unauthorized access to the carrier's fixed-line network in Italy, a system that provides internet service to millions of homes and businesses, according to Vodafone's security briefing documents from 2009 and 2011 seen by *Bloomberg*, as well as people involved in the situation.

Bloomberg further reported that Vodafone asked Huawei to remove backdoors in home internet routers in 2011 and received assurances from the supplier that the issues were fixed, but further testing revealed that the security vulnerabilities remained. Vodafone also identified backdoors in parts of its fixed-access network known as optical service nodes, which are responsible for transporting internet traffic over optical fibers, and other parts called broadband network gateways, which handle subscriber authentication and access to the internet.[8]

"Vulnerabilities in both the routers and the fixed access network remained beyond 2012 and were also present in Vodafone's businesses in the U.K., Germany, Spain and Portugal."[9]

Windows Kernel Driver Vulnerability

In March 2019, various media outlets covered a Huawei driver vulnerability uncovered by Microsoft. Huawei MateBook systems that are running the company's PC Manager software included a driver that would let unprivileged users create processes with superuser privileges. The insecure driver was discovered by Microsoft using some of the new monitoring features added to Windows version 1809 that are monitored by the company's Microsoft Defender Advanced Threat Protection (ATP) service.[10]

[5] https://www.reuters.com/article/us-netherlands-huawei-tech-idUSKCN1SM0UY

[6] https://www.ft.com/content/30ec5c54-83aa-11e9-b592-5fe435b57a3b

[7] https://www.bloomberg.com/news/articles/2019-04-30/vodafone-found-hidden-backdoors-in-huawei-equipment

[8] Ibid.

[9] Ibid.

[10] https://arstechnica.com/gadgets/2019/03/how-microsoft-found-a-huawei-driver-that-opened-systems-up-to-attack/

Microsoft traced this to an app from Huawei called PC Manager; a device management software for Huawei MateBook laptops that was identified by Microsoft as "exhibiting unusual behaviour," with a Huawei-written driver designed to monitor the software's performance (restarting it if it crashed), injecting code into a privileged Windows process and then running that code using an asynchronous procedure call (APC).[11]

"The Huawei driver did make some attempts to ensure that it would only communicate with and restart Huawei's own service, but improper permissions meant that even an unprivileged process could hijack the driver's watchdog facility and use it to start an attacker-controlled process with LocalSystem privileges, giving that process complete access to the local system."[12]

Session Hijack, Heap Overflow and Stack Overflow Vulnerabilities

In July 2012, Felix Lindner and Gregor Kopf gave a conference presentation at Defcon to announce that they uncovered several critical vulnerabilities in Huawei routers (models AR18 and AR29) which could be used to get remote access to the device.[13] Lindner and Kopf also criticized Huawei for its lack of transparency when it comes to security issues. The company didn't have a security contact for reporting vulnerabilities, didn't put out security advisories and didn't say what bugs have been fixed in its firmware updates, the researchers said. Huawei replied that it would be investigating the claims.

UK HCSEC Reports

In March 2019, the Oversight Board of United Kingdom's government organization Huawei Cyber Security Evaluation Centre (HCSEC) found "serious and systematic defects" in Huawei software engineering and their cybersecurity competence. HCSEC also cast doubt on Huawei's ability and competence to fix security problems that have been found.[14]

HCSEC reported that it has not seen anything to give it confidence in Huawei's ability to bring about change via its transformation program and will require sustained evidence of better software engineering and cybersecurity quality verified by HCSEC and NCSC.[15]

> **"HCSEC has continued to find serious vulnerabilities in the Huawei products examined. Several hundred vulnerabilities and issues were reported to UK operators to inform their risk management and remediation in 2018. Some vulnerabilities identified in previous versions of products continue to exist."**
>
> Huawei Cyber Security Evaluation Centre (HCSEC) Oversight Board, Report to the National Security Adviser of the United Kingdom, March 2019

Chinese National Intelligence Law of 2016

The Chinese National Intelligence Law of 2016 requires all companies "to support, provide assistance, and cooperate in national intelligence work."

Huawei has categorically denied that Chinese law would compel it to engage in insecure cybersecurity practices:

"Chinese law does not grant government the authority to compel telecommunications firms to install backdoors or listening devices, or engage in any behaviour that might compromise the telecommunications equipment of other nations. A mistaken and narrow understanding of Chinese law should not serve as the basis for concerns about Huawei's business. Huawei has never been asked to engage in intelligence work on behalf of any government."[16]

[11] Ibid.

[12] Ibid.

[13] https://www.computerworld.com/article/2505191/hackers-reveal-critical-vulnerabilities-in-huawei-routers-at-defcon.html

[14] https://techcrunch.com/2019/03/28/uk-report-blasts-huawei-for-network-security-incompetence/

[15] Ibid.

[16] https://www.aspistrategist.org.au/huawei-and-the-ambiguity-of-chinas-intelligence-and-counter-espionage-laws/

Despite this, it appears clear that for Chinese citizens and companies alike, participation in "intelligence work" is a legal responsibility and obligation, regardless of geographic boundaries.

This requirement is consistent across several laws on the protection of China's state security. For instance, Article 7 of the National Intelligence Law **(国家情报法)**[17] declares:

Any organisation and citizen shall, in accordance with the law, support, provide assistance, and cooperate in national intelligence work, and guard the secrecy of any national intelligence work that they are aware of [emphasis added]. The state shall protect individuals and organisations that support, cooperate with, and collaborate in national intelligence work.[18]

Similarly, Article 22 of the 2014 Counter-Espionage Law **(反间谍法)**[19] states that during the course of a counter-espionage investigation, "relevant organizations and individuals" must "truthfully provide" information and "must not refuse." The implementing regulations[20], released in November 2017, clarified the law's provisions:

"When state security organs carry out the tasks of counter-espionage work in accordance with the law, and citizens and organizations that are obliged to provide facilities or other assistance according to the law refuse to do so, this constitutes an intention to obstruct the state security organs from carrying out the tasks of counter-espionage work according to law."[21]

According to the Australian Strategic Policy Institute:

"The scope and parameters of what Chinese authorities might deem to be 'intelligence work' and 'counter-espionage work' are not clearly defined in these laws—which are, at best, ambiguous and open to varying interpretations."

So even if Huawei may be technically correct in saying that Chinese law doesn't explicitly "compel" the installation of backdoors, there are still reasons for concern. China's intelligence and counter-espionage activities tend to be so expansive that these provisions could be used to justify activities extending well beyond China's borders.[22]

> **"There is no way Huawei can resist any order from the (People's Republic of China) Government or the Chinese Communist Party to do its bidding in any context, commercial or otherwise."**
>
> -Jerome Cohen,
> NYU Law Professor
> Adjunct Senior Fellow,
> Council on Foreign Relations

[17] http://www.npc.gov.cn/npc/xinwen/2017-06/27/content_2024529.htm

[18] Ibid.

[19] http://www.npc.gov.cn/npc/xinwen/2014-11/02/content_1884660.htm

[20] https://www.madeirasecurity.com/detailed-regulations-for-the-prc-counterespionage-law-rush-translation/

[21] Ibid.

[22] Ibid.

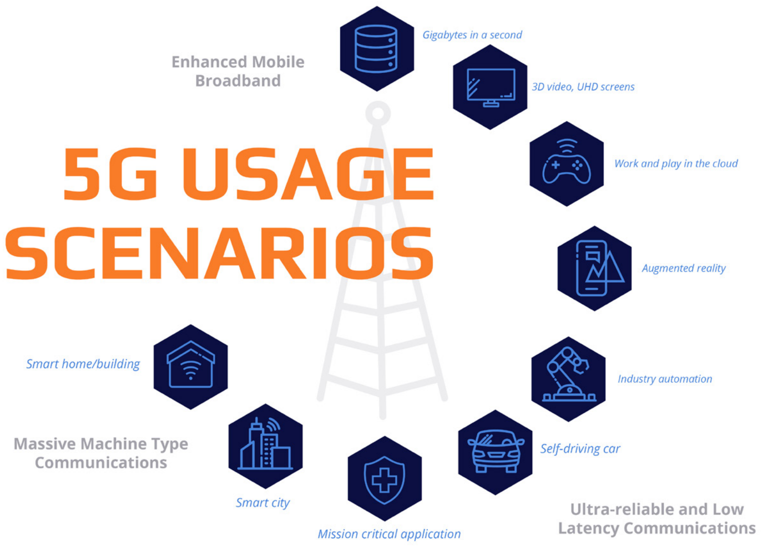
THE 5G THREAT LANDSCAPE

5G networks are highly distributed, complex to secure, and reliant upon long supply chains dominated by one Original Equipment Manufacturer (OEM), Huawei, and the consequences of even relatively isolated attacks on 5G network components could cause cascading failures leading to loss of critical services. Given this threat landscape, it is understandable that 5G has become a focal point for international policy debates. This section of the report provides a deeper dive into these topics and 5G.

According to a 2018 CCDOE report: "5G is the next generation of wireless mobile technology, providing greater data speeds, lower latency (better responsiveness), and the possibility to simultaneously connect to more devices. These qualities will expedite the advance of robotics and automation, virtual and augmented reality, and artificial intelligence and machine learning – transforming the scene of smart devices and applications, and the entire operation of digital societies, very likely in ways unimagined today."[23]

5G networks are rolling out around the world to enable advancements to mobile communications on three fronts: increased bandwidth for high-end experiences (such as virtual reality and HD videos); lower latency, higher reliability connections for critical applications (such as autonomous vehicles); and massive numbers of simultaneous connections for IoT devices (such as in smart cities). These features of 5G will enable new advances that lead to changes to the way individuals, companies, and governments interact on a daily basis. New industries will be created.

This new, ubiquitous connectivity to the Internet will inevitably increase everyone's reliance on these networks. Most importantly, critical industries such as transportation, energy, manufacturing, and communications will rapidly become deeply tied to 5G, meaning that disruption or surveillance of these networks can have much more significant consequences including threats to national security, large-scale espionage, disruption of critical businesses, and even loss of life.


5G USAGE SCENARIOS
Enhanced Mobile Broadband
Gigabytes in a second
3D video, UHD screens
Work and play in the cloud
Augmented reality
Industry automation
Self-driving car
Ultra-reliable and Low Latency Communications
Mission critical application
Smart city
Massive Machine Type Communications
Smart home/building

Why 5G is different than 4G

The 4G era ushered in substantial growth in connected devices and the services available on those devices. Before 4G, it was impossible to stream HD videos on your phone, place video calls, have high-quality satellite maps, or have home internet provided by your cellular carrier. With the advent of 4G, there was growth in new, always-connected IoT devices ranging from smart watches to connected cars. However, most of these devices were designed to survive service disruptions, and virtually no 4G devices required constant connectivity to provide critical services.

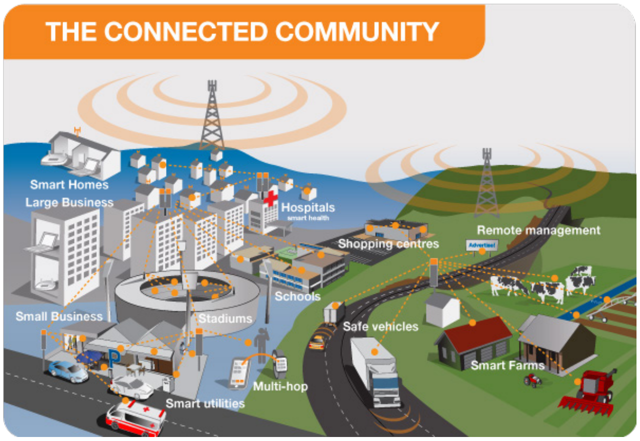
As broadband internet connectivity became ubiquitous, PC software and operating systems shifted from a design paradigm of "sometimes connected" into a rapid dependence upon reliable internet connectivity. Entire operating systems (e.g., ChromeOS) were built that simply didn't function if you were disconnected. Software developers shifted from thick desktop clients into lightweight web apps that can be constantly updated.

[23] https://ccdcoe.org/uploads/2019/03/CCDCOE-Huawei-2018-03-28-FINAL.pdf

5G will have a similar effect on all of the “things” connected to cellular networks. Autonomous vehicles will need low-latency, high-bandwidth connectivity to make sense of complex environments. Industrial robots won’t be able to perform their most advanced tasks without the assistance of a cloud-enabled AI. In short, within a decade, our most critical services could cease to function without connectivity, inextricably tying 5G to national and economic security.

As explained in a report from The European Union Agency for Network and Information Security (ENISA),

“Telecommunications are key in nowadays societies. They represent the backbone, the primary infrastructure based on which our society works and constitute the main instrument in allowing our democracy (and other EU core values such as freedom, equality, rule of law, human right) to function properly.”[24]



25

5G Network Architecture

Mobile networks have two fundamental parts, the Radio Access Network and the Core Network.

Radio Access Network. This is the portion of the network responsible for connecting wireless devices and mobile users. It includes towers and masts, as well as small cells, in-building, and in-home systems.

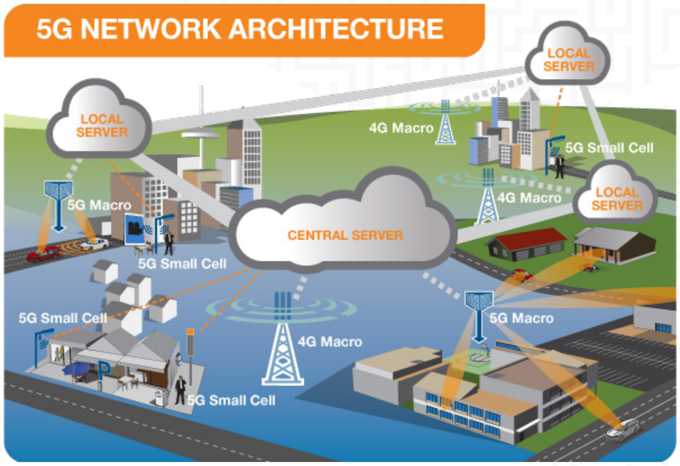
As part of 5G architectures, small cells will provide short-range connections as a complement to the wide-area coverage provided by towers. Small cells will often make use of new millimetre wave (mmWave) frequencies, which have very short connection ranges. By distributing small cells in clusters in areas of the highest need, users will still be able to maintain continuous connections.

Massive multiple input, multiple output (MIMO) antennas will be of similar size to 3G/4G antennas, but provide for a “massive” number of connections. They will act as macro cells, simultaneously connecting more devices and people, while still maintaining high data throughput.

Core Network. The voice and data connections provided by the Radio Access Network converge onto the core network. The core network is undergoing a major redesign as part of 5G architectures, to allow for more-seamless integration with the Internet and various cloud-based services.

Network function virtualization (NFV) and network slicing will allow for more flexible and efficient network deployment and management. Distributed services will improve overall network responsiveness and reduce latency.

Illustrated below: 5G and 4G working together as part of the same network. A combination of centralized and distributed local servers provides faster response times to users and devices.[26]



[24] https://www.enisa.europa.eu/publications/signalling-security-in-telecom-ss7-diameter-5g

[25] The EMF Explained Series - www.emfexplained.info, http://www.emfexplained.info/?ID=25916

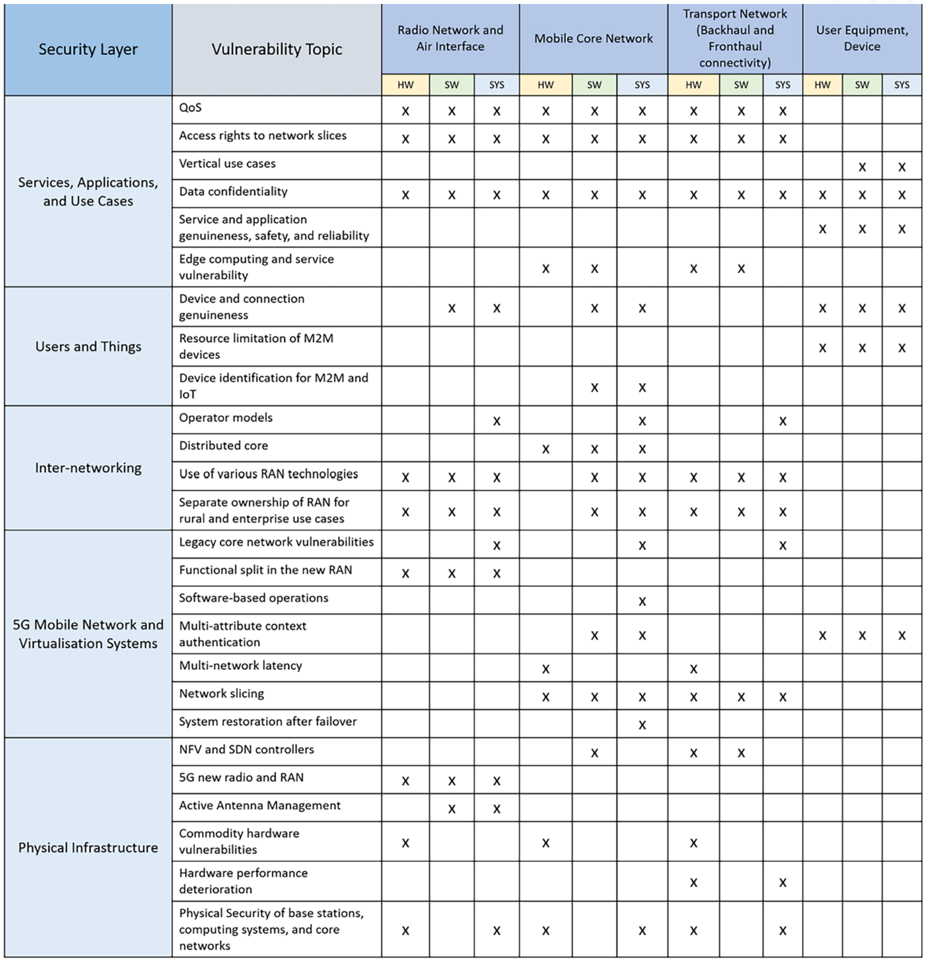
[26] Ibid.

5G Network Security Challenges

It is clear that the consequences of a security breach can be catastrophic. It is also clear that the attack surface for a 5G network is more complex than previous generations of cellular networks. In order to achieve the high-bandwidth, low-latency, massively parallel communications for 5G, several new technologies are required:

- **Software Defined Networking (SDN)** Abstracts lower-level network functions away from hardware and into software, allowing for more dynamic and adaptable management of network functionality, implementing most functions in software rather than hardware.
- **Network Function Virtualisation (NFV)** Allows multiple distinct instances of various network services (e.g., routers, load balancers, firewalls) to be run as virtual machines on top of the same shared hardware.
- **Multi-access Edge Computing (MEC)** (aka Mobile Edge Computing) is an approach to networking which moves data and services to the network edge and closer to end users. It facilitates lower-latency and higher-bandwidth connections, while also reducing congestion in the core networks.
- **Distributed Core Network** separates and virtualizes core network functions which have traditionally been centralized and distributes them geographically, reducing latency and allowing for more responsiveness to high local user demand.
- **Network Slicing** is a virtual network architecture related in concept to SDN and NFV. It allows an operator to build end-to-end, isolated "slices" of network services in support of specific business use cases or customer groups.

Each of these new technologies brings with it new security challenges. For 5G to be successful, both the Radio Access Network and the Core Network have become more distributed, complex, fragmented across suppliers, and dependent upon software. This inevitably makes security more challenging. Most notably, there is more processing happening at the edge of the network, so the traditional boundaries between the "edge" and the "core" are blurrier. Risk management steps are still possible, but they look different than older generation networks.



Security Layer	Vulnerability Topic	Radio Network and Air Interface			Mobile Core Network			Transport Network (Backhaul and Fronthaul connectivity)			User Equipment, Device		
		HW	SW	SYS	HW	SW	SYS	HW	SW	SYS	HW	SW	SYS
Services, Applications, and Use Cases	QoS	x	x	x	x	x	x	x	x	x			
	Access rights to network slices	x	x	x	x	x	x	x	x	x			
	Vertical use cases											x	x
	Data confidentiality	x	x	x	x	x	x	x	x	x	x	x	x
	Service and application genuineness, safety, and reliability										x	x	x
	Edge computing and service vulnerability				x	x		x	x				
Users and Things	Device and connection genuineness		x	x		x	x				x	x	x
	Resource limitation of M2M devices										x	x	x
	Device identification for M2M and IoT					x	x						
Inter-networking	Operator models			x			x			x			
	Distributed core				x	x	x						
	Use of various RAN technologies	x	x	x		x	x	x	x	x			
	Separate ownership of RAN for rural and enterprise use cases	x	x	x		x	x	x	x	x			
5G Mobile Network and Virtualisation Systems	Legacy core network vulnerabilities			x			x			x			
	Functional split in the new RAN	x	x	x									
	Software-based operations						x						
	Multi-attribute context authentication					x	x				x	x	x
	Multi-network latency				x			x					
	Network slicing				x	x	x	x	x	x			
	System restoration after failover						x						
Physical Infrastructure	NFV and SDN controllers					x		x	x				
	5G new radio and RAN	x	x	x									
	Active Antenna Management		x	x									
	Commodity hardware vulnerabilities	x			x			x					
	Hardware performance deterioration							x		x			
	Physical Security of base stations, computing systems, and core networks	x		x	x		x	x		x			

[28]

A report by the UK Department for Digital, Culture, Media, and Sport[27] highlights some of these security challenges. Examples include:

- Due to an increased reliance upon software for most functions of the network (rather than hardware), there is an increased attack surface size and increased likelihood of vulnerabilities,

[27] Ibid.

[28] https://uk5g.org/discover/research/technical-report-5g-network-architecture-and-secur/

- The new 5G radios are more dense and fragile, and thus, attacks originating in or targeting the air interface may have larger effects, and
- The distributed nature of the network increases the cyber and physical security challenges.

In summary, the report states “5G networks have the unique property of merging different types of networks and technologies under one umbrella system, requiring interoperability, efficiency, and seamless connectivity, and support for the requirements of a large number of diverse use cases.”

This complexity increases security challenges and the amount of trust we must place in the hundreds of vendors supplying hardware and software components to these networks.

5G Network Suppliers

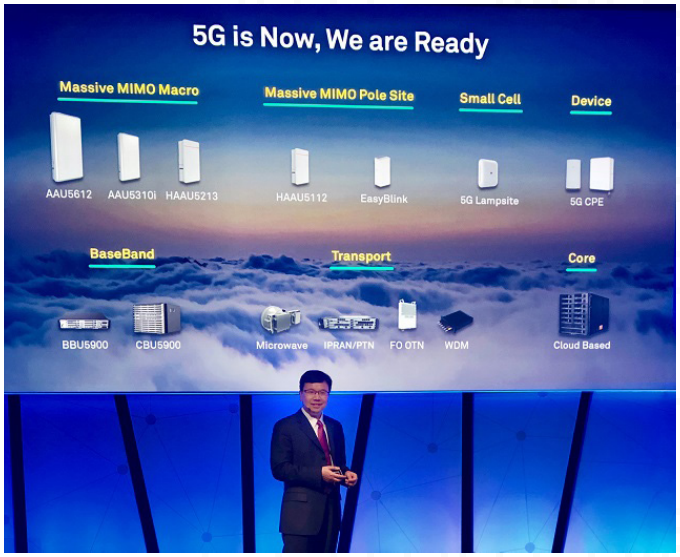
Despite the fact that 5G networks are substantially more complex than previous generations, there are still really only five primary suppliers of 5G radios and core network hardware: Huawei, ZTE, Nokia, Ericsson, and Samsung. Due to a combination of mergers and acquisitions and difficulties competing with Chinese government subsidies[29] for 5G technologies, the market has become far less competitive than it used to be.

The reality is that market competition is not only sparse, it’s also far from evenly matched. Over the past several years, by virtually all accounts, Huawei has become the dominant force in 5G equipment. According to phys.org, “[Huawei] has received hundreds of millions of dollars in grants, heavily subsidised land to build facilities and apartments for loyal employees, bonuses to top engineers, and massive state loans to international customers to fund purchases of Huawei products.”[30] This government support has enabled them to make technological advances beyond several of their competitors and gain substantial market share by undercutting the competition on price.

Huawei 5G Products and Services

Given all of the components in 5G infrastructure, the natural question is: What parts of the system does Huawei provide? The short answer is: Almost everything.

Huawei’s investments in 5G are extensive. They have solutions for virtually every part of the core and radio access network: large and small radios, microwave backhaul transport, servers to run the edge and core cloud, and of course, high-end routers and switches to handle all of the traffic. Huawei even offers tools and services to support 5G site planning, radio propagation analysis, and power consumption planning.



Yang Chaobin, President of Huawei 5G Product Line[31]

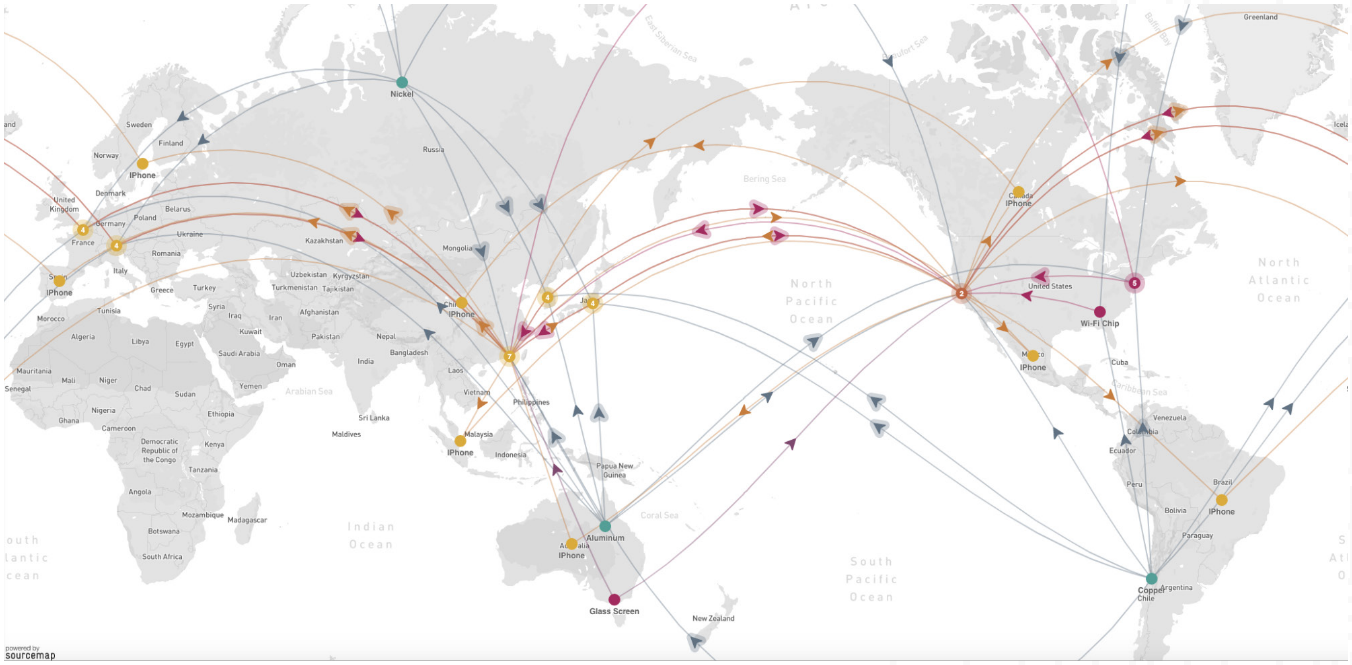
> **“Cyberspace is a new strategic domain, but it is unlike the physical territory of which we are used to. It has gradually become the ‘nervous system’ through which society operates.”**
>
> “21st Century Technology and Security – A Difficult Marriage” by John Suffolk, Global Cyber Security and Privacy Officer, Huawei

[29] https://www.bloomberg.com/news/articles/2011-04-25/huawei-counts-on-30-billion-china-credit-to-open-doors-in-brazil-mexico

[30] https://phys.org/news/2019-05-huawei-key-beneficiary-china-subsidies.html

[31] https://www.huawei.com/en/press-events/news/2018/2/Huawei-Launches-Full-Range-of-5G-End-to-End-Product-Solutions

SUPPLY CHAIN SECURITY CHALLENGES



[32]

The effects of massive globalization are clear in modern telecommunications devices, built via a long chain of hardware, software, and service providers. These components form the security foundation of every device. This section covers the complexity of supply chains, the attack surface they create, and the complexity of securing them, as well as examining the means by which cyber attacks would be possible.

Even with substantial consolidation for telecommunications equipment manufacturers, the supply chain for any single device is far more complex than it appears. Huawei, for example, lists 150 global suppliers in their supply chain[33], and a single 5G network component could have contributions from dozens of hardware and software suppliers.

This long chain of hardware, software, and service providers forms the foundation of a trust relationship that every network operator is entering into when they place equipment on their network. Any of those components might contain critical vulnerabilities or backdoors that could be exploited.

According to Simha Sethumadhavan, a professor at Columbia University, "Hardware is like a public good because everybody has to rely on it. If hardware is compromised in some way, you lose security in a very fundamental way."[34] This perspective can be broadened to apply to hardware, firmware, and any other system software deeply embedded inside devices. These components form the security foundation of every device, but to understand the complexity of securing them, you must first understand the complexity of the supply chains themselves.

Modern Electronics Supply Chains

The effects of massive globalization may be no more apparent than when looking inside a modern telecommunications device. Dozens of components made from companies around the world bounce through multiple layers of suppliers and integrators until they are placed on a board, tested, and packaged by the OEM. These components range from complex processors such as CPUs and FPGAs to small, passive components like resistors and capacitors, and they come from countries in virtually every corner of the globe.

[32] https://open.sourcemap.com/maps/5cdacebfcefd58d813b66635b

[33] https://gbtimes.com/huawei-lists-33-us-companies-among-core-suppliers

[34] https://www.technologyreview.com/s/519661/nsas-own-hardware-backdoors-may-still-be-a-problem-from-hell/

The supplier of the components also provides accompanying software with each component. This software could include microcode buried inside a processor, firmware that interfaces with the hardware, operating systems to run application software, and much more. Thus, just like hardware, various software components are passed through the supply chain and built on, layer after layer, until they make their way to the OEM. Then, the OEM integrates all of that microcode, firmware, and software with *their* custom software and software of third-party software suppliers to create a firmware or software *build* or *image*. That image is programmed to the devices as they are assembled and shipped to the end users. In the end, that image could contain software written by thousands of engineers at dozens of companies across many different countries.

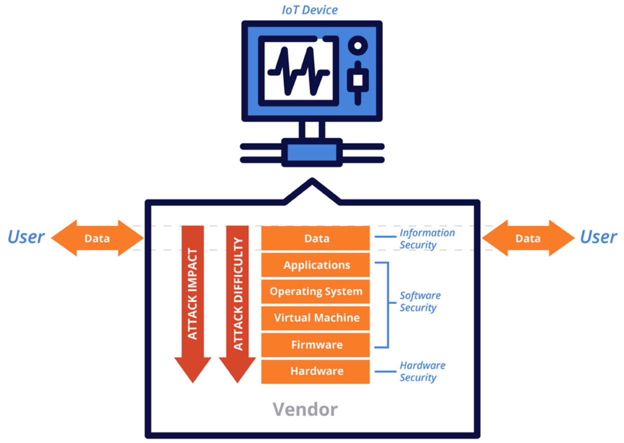
With such a complex process in place, inevitably, there will be errors in the firmware image. When this happens, what does the OEM do? They reassemble that firmware image with the new fixes in it and provide it to all of their customers through a website download or automatic update mechanism built into their product.

With entangled national and economic security issues arising from 5G, the next question is, what happens if any vendor in the supply chain intentionally or unintentionally becomes the attacker?

Through analysis of hundreds of thousands of firmware images, Finite State has found that, on average, **more than 80%** of software in a device is duplicated in other devices - illustrating just how interconnected software supply chains are.

The Supply Chain Attack Surface

Cybersecurity is hard enough as it is, and it gets substantially harder when users can't trust the vendors of the products. It's hard to secure systems from an adversary that knows substantially more about those systems than the defenders do. Not to mention, at a minimum, most vendors are in control of the software updates being applied to their systems, and that gives them a platform for causing widespread damage. The good news is that most vendors want to build more secure products. Customers are demanding better security, and compromising that through an intentional backdoor would be detrimental to their business. Regardless of intent, security vulnerabilities will still exist, and this section examines the means by which cyber attacks would be possible.



Connected devices are like black boxes, making it difficult for security teams to properly assess risk regarding the components running inside

Hardware Backdoors

Of the entire supply chain attack surface, the most devastating attacks are hardware backdoors. According to *Science Direct*, "Despite a few allegations against various governments, there are no publicly confirmed cases of backdoors in computer hardware being deployed. However, in recent years security researchers have repeatedly demonstrated the power and stealth of compromised hardware." These demonstrations range from "demonically clever" insertion of tiny analog components into a microchip[35] to hidden instructions and registers added into the processor architecture.[36]

Given demonstrations of methods for adding backdoors to hardware, there is concern and speculation that governments may already be doing this. The most widely publicized accusation is that of Edward Snowden against the National Security Agency (NSA). As Glen Greenwald explains that "routers built for export by Cisco (and probably other companies) are routinely intercepted without Cisco's knowledge by the National Security Agency and equipped with

[35] https://www.wired.com/2016/06/demonically-clever-back-door-hides-inside-computer-chip/

[36] https://github.com/xoreaxeaxeax/rosenbridge

VULNERABILITIES VS. BACKDOORS

In cybersecurity, a vulnerability is a weakness which can be exploited by a threat actor, to perform unauthorized actions within a computer system. To exploit a vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness.

A backdoor is a method of bypassing normal authentication or encryption in a computer system, a product, or an embedded device, or its embodiment. Backdoors are often used for obtaining remote access to a computer or obtaining access to plaintext in cryptographic systems. Although some are secretly installed, other backdoors are deliberate and widely known. These kinds of backdoors have “legitimate” uses such as providing the manufacturer with a way to restore user passwords.

In essence, backdoors are a type of vulnerability. For example, leaving an administrative account accessible over telnet using the password ‘12345’ is a vulnerability. An attacker with that knowledge can gain backdoor access to the device. The colloquial term “backdoor” typically means something deliberately inserted.

So what is the difference between a backdoor and a vulnerability? Intent, which is very difficult to prove. The best intentional backdoors look like security oversights, so they can be 100% deniable.

hidden surveillance tools.”[37] Similar accusations have been made against China by US national security experts.

The reasons these attacks are so devastating are:

- They are notoriously difficult to detect, especially given that a few transistors and capacitors are all it takes (amongst billions), and
- No software defenses can truly overcome a hardware backdoor, and they cannot be patched after detection.

The “good” news about hardware backdoors is that they are very challenging to implement correctly and covertly, and thus, they are uncommon.

Firmware and Software Backdoors

As we have discussed in this section, device firmware tends to be full of vulnerabilities already, and thus, adding additional code to enable some type of access is easy to hide. Attackers know this, and because they can get most of the benefit of a hardware backdoor with a fraction of the effort, plus substantially more deniability, firmware has emerged as the attack surface of choice. If a backdoor can be inserted in firmware, it is typically undetectable from the operating system or applications running on the system (i.e. endpoint security software). Given that thousands of developers may contribute to a final firmware image, it is also notoriously difficult to attribute where in the chain a backdoor was inserted.

Firmware and software backdoors can take many forms. One of the simplest backdoors is adding a known, default username and password to a device. Given its simplicity, it would seem this technique might be rarely observed, but, in fact, default credentials are commonly discovered in network and IoT devices. One example is the Mirai botnet DDoS attack in 2016, which co-opted 600,000 devices with backdoor accounts into a botnet that generated 623 Gbps of traffic, which was used to shut down entire sections of the Internet.

There are many other backdoor techniques seen in a variety of other devices. In 2013, D-Link routers were found to contain a backdoor allowing remote access by setting a browser’s user agent string to “xmlset_roodkcableoj28840ybtide.”[38] In 2018, four different models of Android phones were found by the German government to have an unremovable backdoor

[37] https://www.infoworld.com/article/2608141/snowden--the-nsa-planted-backdoors-in-cisco-products.html

[38] https://www.computerworld.com/article/2486450/d-link-issues-fixes-for-firmware-backdoor-in-routers.html

embedded inside that could collect intelligence on the user.[39] Of most relevance was a Vodafone report that it found backdoors in Huawei home gateways and larger network devices in 2011.[40]

The further upstream the backdoor is in the supply chain, the more serious the consequences. In 2018, Finite State researchers found a backdoor embedded in firmware associated with a chipset from Senao Networks that allows any user to escalate to root by simply entering the command "*[command redacted due to ongoing disclosure]*" Because of the pervasiveness of the chipset, this backdoor affects hundreds of products.

Compromised Firmware & Software Updates

One of the most challenging aspects of supply chain security for devices is that the supply chain doesn't end the moment a device is placed on the network. As previously mentioned, most device manufacturers are continuously updating their firmware and software and making that available to users through the Internet. *This is generally a good thing,* as most firmware updates are designed to patch vulnerabilities. The most responsible OEMs will issue these patches regularly to ensure their devices are secured against newly disclosed vulnerabilities. The problem is that each firmware update could completely change the software in the device.

Without a robust security regime in place at the OEM, a single developer or a single supplier upstream in the supply chain could insert malicious code that makes its way into a firmware image undetected. Even worse, the update servers themselves could be compromised and files modified by a malicious third party. In fact, that exact scenario occurred earlier this year.[41] Taiwanese electronics OEM ASUS inadvertently sent malware to hundreds of thousands of PCs due to a compromised software update server.

This same technique was used by the Russian threat actor group known as Energetic Bear in 2014.[42] During that operation several developers of Industrial Control System (ICS) software were targeted in an operation that trojaned software updates destined for critical industrial and energy networks. More than 250 companies were affected. "These infections not only gave the attackers a beachhead in the targeted organizations' networks, but also gave them the means to mount sabotage operations against infected ICS computers," according to Symantec.

> **"The researchers estimate half a million Windows machines received the malicious backdoor through the ASUS update server, although the attackers appear to have been targeting only about 600 of those systems."**
>
> Kim Zetter, *Motherboard* [40]

Rogue Service Providers

Unlike consumer devices on home networks, when buying expensive telecom equipment, the hardware frequently comes with a support contract. This equipment is complex to install, configure, secure, update, and troubleshoot, so representatives from the vendor are best suited to service these devices. This makes logical sense and is the standard way business-to-business technology acquisition works. But, when dealing with critical networks, there are obvious security concerns associated with these services.

For example, firmware updates in telecom networks are quite complex and often require vendor assistance. Periodically, a new "release" will be patched into the network using a combination of proprietary tools and vendor personnel who are often physically present in these events. While the risks can be managed through extensive testing, verification, and monitoring, these types of services are obvious ways that an attacker could gain access to a sensitive network and install malicious firmware updates

Approaches to Supply Chain Security

In short, there is an enormous attack surface that exists through the modern, complex hardware and software supply chains that enable modern electronics systems. Just like any other security decision, buyers

[39] https://www.zdnet.com/article/germany-backdoor-found-in-four-smartphone-models-20000-users-infected/

[40] https://www.bloomberg.com/news/articles/2019-04-30/vodafone-found-hidden-backdoors-in-huawei-equipment

[41] https://www.vice.com/en_us/article/pan9wn/hackers-hijacked-asus-software-updates-to-install-backdoors-on-thousands-of-computers

[42] https://www.nytimes.com/2014/07/01/technology/energy-sector-faces-attacks-from-hackers-in-russia.html

of these systems should apply a risk management approach to supply chain security. A comprehensive security program can mitigate many of the risks posed by supply chain threats. In particular, with software and firmware validation enabled by new technology, these risks can be identified and remediated before devices are added to the network.

To that end, there are examples of supply chain security efforts that are worth noting. The most notable is The UK Huawei Cyber Security Evaluation Centre.

Founded in 2010, the UK HCSEC may be the most comprehensive approach to telecom supply chain security. The Centre exists "to mitigate any perceived risks arising from the involvement of Huawei in parts of the United Kingdom's (UK) critical national infrastructure" and provides security evaluations of an extensive range of products from Huawei. Through this arrangement, HCSEC has access to the source code of these devices, which enables them to perform code audits along with many other verification activities. Over eight years, HCSEC has identified and reported hundreds of vulnerabilities to Huawei with the goal of improving the security of these devices.

Beyond source-code analysis, HCSEC is able to audit Huawei engineering practices, maintain open communications with its security team, and conduct extensive testing of Huawei devices. While this approach is comprehensive, it has limitations (primarily a lack of binary equivalence), and in particular, it is hard for other countries and private enterprises to replicate.

However, with HCSEC as a model, there are many steps that other organizations can take to bolster their supply chain security.

Best Practices and Recommendations

Know Your Vendors

The first step in supply chain security is simply understanding your organization's supply chain. Believe it or not, most enterprises don't really know what devices they have on their networks. Security teams should generate an inventory of all the devices they have, and they should work with procurement to understand more about each device and its supply chain.

Leverage Your Buying Power

When it comes to expensive telecommunications equipment, buyers have a lot of power. Buyers should insist on adding language to contracts that allows them to conduct independent security testing of every device and their corresponding security updates. In addition, buyers should establish a channel with the vendor to report these findings.

Verify Everything

Especially in critical infrastructure environments, every device should be thoroughly tested before deployment, and more importantly, the firmware should be analyzed using automated analysis software. Vulnerability testing of most devices will report back a list of possible defects. Firmware testing will go far beyond that and provide a deep understanding of how secure the software and firmware is. While comprehensive firmware analysis was infeasible a few years ago, the technology now exists. The rest of this report shows just how powerful that can be.

Collaborate on Remediation

Most device manufacturers actually do want to build better, more secure products, and they want to know about vulnerabilities that are discovered. Most vulnerabilities identified will be new information to the vendor, so it's advisable for buyers to try to build a helpful relationship by reporting what they find. Remember, just because there is a vulnerability in a product doesn't mean that the OEM is the originator. Devices are built through complex supply chains, and vulnerabilities could have been introduced by one of the suppliers.

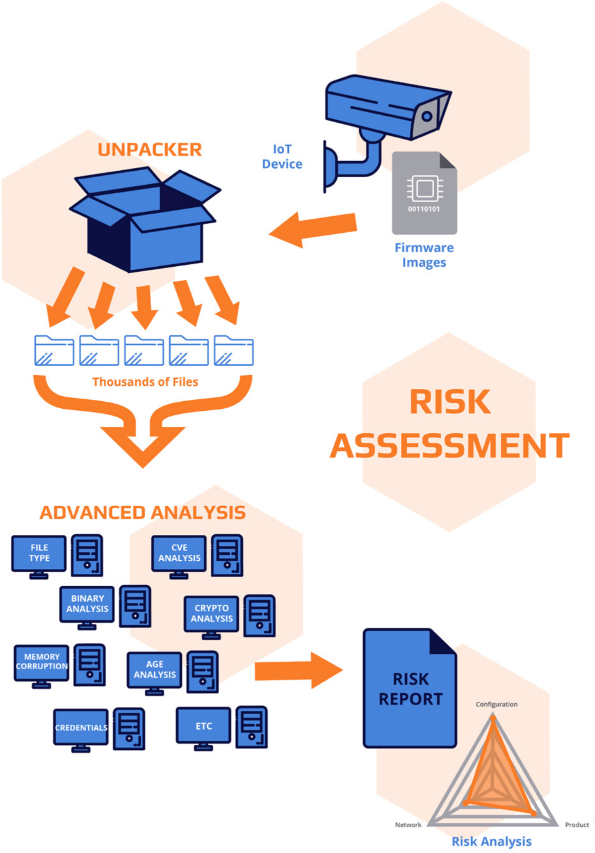
Transparency Leads to Better Security

In just about every example in history, increased transparency directly leads to better security. The more eyes that are able to look at a device or its source code, the more likely someone will spot a defect. At Finite State, transparency is core to our mission. The rest of this report details how increased transparency, enabled by our firmware analysis technology, can provide clarity around the true risks of these devices rather than relying upon potentially politically charged accusations.

RISK ASSESSMENT METHODOLOGY

Finite State has developed the world's largest automated firmware analysis platform, Iotasphere, which deeply analyzes embedded device firmware images at massive scale. This section outlines the analysis process, including firmware unpacking and the attributes we analyze to ultimately measure security risk.

Finite State has developed the world's largest automated firmware analysis system to support our mission of protecting the next generation of networks. Our approach leverages a fusion of automatically extracted firmware risk data with passively generated network inventory to allow organizations to continuously see every device on their network, assess their risks, detect intrusions, and respond to attacks.



Iotasphere, ingests firmware images which then triggers automated analysis. Analysis begins by unpacking the firmware using our highly extensible library of dozens of different unpacking modules. Once the firmware is unpacked, each file runs through a pipeline of static and dynamic analyzers.

Unpacking Firmware

Iotasphere includes dozens of custom firmware unpackers. These unpackers allow our system to search through monolithic binary firmware images and break them into component parts which are then fed through various analysis modules.

Analyzing Firmware

As files are unpacked from a firmware image, they are processed by a series of file-level analysis modules. These modules perform static, dynamic, and symbolic analysis on files as they pass through stages of the analysis process.

Outdated Components

Our analyzers use signatures contained within the binary of an application or library to determine the specific software name and version. Once identified, this information is used to map the software component and version to the effective date the software was released. The system uses these application dates to compute an "average age" of the software components in the firmware. This metric is used to determine if the vendor is updating components over the lifetime of a product or if they are simply patching bugs as they are reported.

Presence of known vulnerabilities

Software names and versions, along with file paths from the extracted filesystem, are used to construct queries against the National Institute of Standards and Technology (NIST) National Vulnerability Database (NVD) to determine the set of Common Vulnerabilities and Exposures (CVEs) that affect a given firmware image. This metric is used to catalog the complete set of known vulnerabilities that might affect a given product rather than relying on vulnerabilities that have been disclosed specifically against that product.

Hard-coded credentials and cryptographic material

We use a signature-based approach to identify and extract credentials and cryptographic material from files within a firmware. The system attempts to recover plaintext passwords for all hashed passwords it encounters. Cryptographic material is categorized based on the type of material, signing properties, and the encryption algorithms used.

Code hygiene and safety

We analyze binary code for uses of functions that commonly lead to software vulnerabilities as well as their safe equivalents. We derived our list of "safe" and "unsafe" functions based on those described in the Common Weakness Enumeration Specification[43], specifically CWE-242 (Use of inherently dangerous function) and CWE-676 (Use of potentially dangerous function). This analysis counts the number of places a dangerous function can be invoked in each binary. It does the same for each safe replacement function. A score is then computed based on the occurrence of safe versus unsafe function utilization.

Automated vulnerability discovery

Binaries that utilize a high number of dangerous functions or are determined to take input directly from a user are further analyzed to determine if any of the suspected vulnerabilities can be verified. This verification is very computationally expensive, so this analysis is only run on a subset of the extracted files. The result of this analysis is a proof-of-vulnerability (PoV), which is essentially a proof-of-concept exploit of a specific vulnerability.

Similarity and Code Genealogy

Iotasphere performs file-level similarity analysis to determine the relationship between the file being analyzed and all previously analyzed files. These relationships reveal shared code between product lines and across manufacturers due to the use of open source software, system on a chip (SoC) software development kits (SDK), and vendor frameworks. Mining the dataset for these relationships enables a much deeper and faster risk analysis due to the transitive property of vulnerabilities in shared code.

Quality change over time

The system compares the output of the previously described analysis across multiple firmware versions of the same product. This results in a metric that quantitatively describes how the product's security is improving, declining, or staying stagnant over time.

Limitations of the Approach

Iotasphere performs analysis on individual components extracted from a firmware image. Our analyzers produce metrics based on features extracted from these components. It is not currently possible to infer context such as how this component is used by the system, if the component is stale code left behind, or if it is part of the critical function of the device.

Incomplete extraction of a firmware image results in false negatives during analysis since the system is only analyzing the components of the firmware it is able to extract. This may happen due to encryption or proprietary packing.

A subset of CVEs are linked to a firmware based on a heuristic that places some degree of trust in the names of files contained within the extracted file system and the version information compiled into the binary. The correlation to a firmware also takes into account only the presence of vulnerable code, not the application of vulnerable code.

Firmware Analysis at Scale

Iotasphere performs a very detailed analysis of embedded device firmware images, and the capabilities of the system improve as the quantity of firmware it has processed grows. For those reasons, we set out to build a system that enables us to perform this type of analysis on a massive scale.

Iotasphere is built on an extensible architecture using modern, cloud-native design paradigms and advanced analytical software. To date, Iotasphere has processed more than 250,000 firmware images – a collection of more than 35 million extracted files. Because of this massive data set at our disposal, our analysis is more comprehensive. Finding a vulnerability, signature, or risk in one firmware image generally benefits the analysis of many other firmware images.

Due to comprehensive de-duplication and a massively scalable processing architecture, we have benchmarked the system processing and analyzing more than 1,000 firmware images per hour. Analysis results for even the most complex firmware images are generally available to the user in a matter of minutes.

[43] https://cwe.mitre.org

Huawei Firmware Analysis

Finite State's Huawei firmware dataset includes 9,936 firmware images covering 558 products. This report focuses on approximately 10% of the total firmware images that we believe is most relevant to the ongoing Huawei security discussion (based upon device function and release date of firmware). The results presented here were derived from the automated analysis of firmware for products used by enterprises and infrastructure providers.

The analyzed firmware represents an evolution of Huawei devices over a period of approximately 18 months. The most recent images are current as of April 2019. It includes firmware images for legacy and current products spanning 14 years. In preparation for this report, we fully reprocessed all of the Huawei firmware in our collection to ensure it was processed by the latest version of our pipeline. Reprocessing the entire Huawei dataset took approximately 36 hours and resulted in approximately 31 million analysis tasks. All of the analysis is automated, but the results were manually checked for accuracy.

USE OF OUTDATED COMPONENTS

In the *Supply Chain Security Challenges* section, we described the complexity and size of the hardware and software supply chains leading to the release of a device. Huawei products, like other networking products, are complex systems that comprise software components from many different manufacturers and open source projects. Complexity increases with the diversity of software components used in the device, and vulnerabilities increase with complexity. To assess this risk, our system first generates a software bill of materials (SBOM) for each firmware image that includes each component and its release date. This SBOM helps form the foundation of understanding the complexity of the system and the supply chain through which it was built.

One of the universal truths in cybersecurity is that the older the software is, the more vulnerabilities you are going to find in it. This is due to a combination of security researchers having more time to find vulnerabilities and the fact that older software doesn't have the advantage of new security features created since it was implemented. In the IT world, that is why it is so important to always install software updates. Enterprises have entire departments that focus solely on this problem.

The challenge with embedded systems, like network equipment and IoT devices, is that end users are dependent on the vendor to keep their devices updated whenever a software update affecting their firmware is released. The timeliness with which OEMs respond to individual component software updates is a key contributor to the overall security for a device. Many device manufacturers are not accountable to this activity, because without a system like Iotasphere, the user of the device has no way of verifying how up to date the components inside their device firmware really are.

To analyze how responsive Huawei is to third-party software updates, our system computed the average age of the software components in each firmware image.

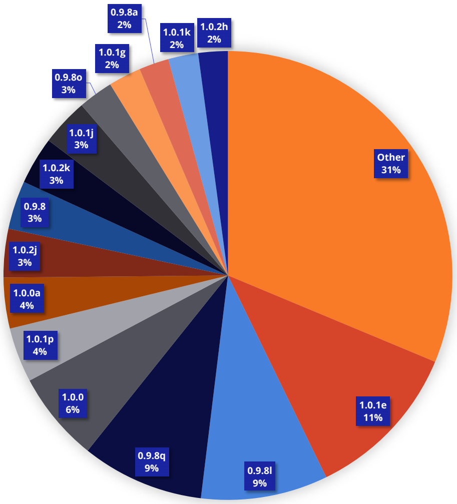
Across our entire Huawei firmware dataset, the average age of third-party components in the latest firmware versions was 5.36 years.

Results Summary

Across the dataset, we found that Huawei does not keep their components up-to-date. When just looking at the *most recent* version of each firmware image, the average age of third-party components is 5.36 years. There were thousands of instances of components that are more than 10 years old.

We found full and partial copies of 79 distinct OpenSSL versions in 3,062 unique files present in Huawei firmware images. The oldest version was released in 1999. We found no evidence of Huawei backporting security patches into their older binaries, as is the practice of security-conscious vendors when packaging binaries.

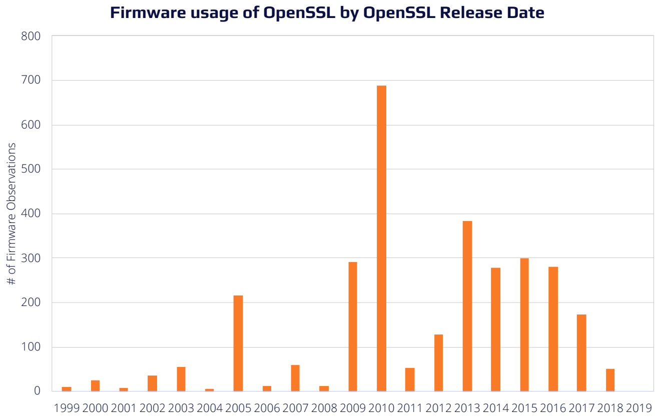
Top OpenSSL Versions



Minor patch revisions to OpenSSL are not applied. Versions such as 0.9.8d and 1.0.1c are still found, rather than the latest versions such as 0.9.8zh or

1.0.1u. This leaves these packages vulnerable to well-known attacks with active exploit packages, such as Heartbleed.[44] Indeed, 389 binaries installed on Huawei firmware are vulnerable to Heartbleed. When considering only CVSS scores of 10.0, the highest possible score, 1,405 OpenSSL binaries found on Huawei firmware are at risk.

Firmware updates for Huawei's CH221 released in November 2018 included code from OpenSSL 0.9.8l. This version of OpenSSL was first released in February of 2010 and contains several maximum-severity CVEs. Prior firmware releases for the device contain newer, more secure versions of OpenSSL up to version 1.0.2j. Huawei's security practices should not allow such regression in device security to occur, especially a device expected to be used in critical infrastructure.



So many distinct versions of OpenSSL, especially with older versions, suggests Huawei has tightly coupled application code with specific binary versions. One example is a commonly reused subsystem Huawei maintains called the "Cable Modem Termination System." This contains statically linked code fromOpenSSL 0.9.7a, which was originally released in April 2003. This version of OpenSSL is highly vulnerable to well documented attacks and has multiple active exploit kits that can target it.

We see similar behavior when considering other binaries commonly installed across firmware. OpenSSH provides an SSH server, which is typically used for securely accessing the device. For OpenSSH, we observed 29 distinct binary versions ranging from version 4 to 7.6 spread across 113 firmware images. Samba allows integration with Windows devices to provide file sharing and print services. For Samba, there are 58 distinct binary versions with 1,858 installations and a version range of 1.9.18 to 4.7. The pattern continues with web content hosting services such as Lighttpd. These are 18 different binary versions, ranging from 1.4.11 to 1.4.45 for Lighttpd with 5,046 binaries installed.

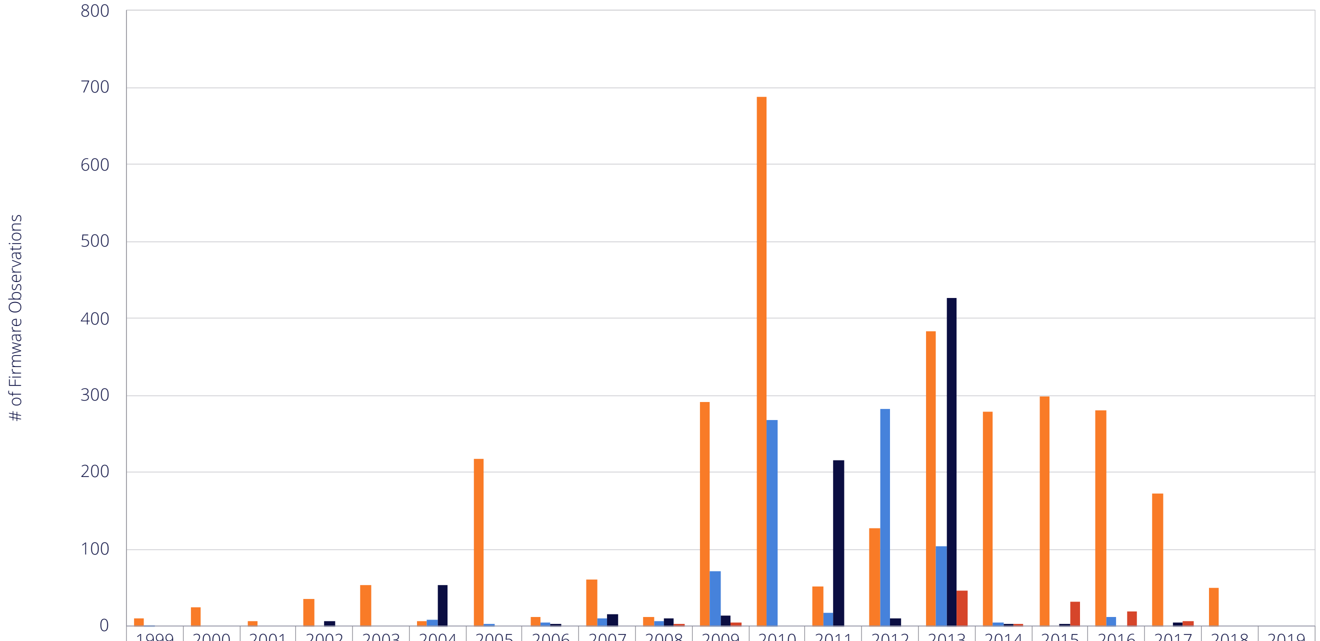
The tight coupling of Huawei-developed code to these old versions demonstrates a lack of maintainability and general poor product architecture. This has been observed across many different critical packages. Combined with poor patching hygiene relating to the administration of these devices, we can conclude this is a pervasive problem with a minimum amount of oversight.

Detailed Example

We investigated the latest version of firmware available for the Huawei AR1200 series enterprise router, V200R007C00SPCc00, and found an average component age of 12.8 years across the tested components at the time the firmware was released. The AR1200 firmware used in this example was released April 13, 2017. It appears to include firmware for the main routing platform as well as several optional interface modules. Within the firmware, our analysis found two distinct OpenSSL versions 1.0.1k and 0.9.7f. These were released in 2015 and 2005 respectively. Our analysis found three version of the Linux kernel, 2.6.34, 2.6.30, and 2.6.16. These were released in 2010, 2009, and 2006. Three distinct versions of BusyBox were also found in the firmware, 1.18.4, 1.2.1, 1.00. These versions were released 2011, 2006, and 2004.

[44] https://www.us-cert.gov/ncas/alerts/TA14-098A

Top Component Usage By Date



	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
🟧 OpenSSL	10	25	7	36	54	6	217	12	60	11	292	688	52	128	384	278	299	280	173	50	0
🟦 Linux Kernel	1	0	0	0	0	9	2	4	10	6	71	267	18	282	104	4	0	12	0	0	0
⬛ BusyBox	0	0	0	6	0	54	0	3	16	10	14	0	215	10	427	3	3	0	4	0	0
🟥 OpenSSH	0	0	0	0	0	0	0	0	0	3	4	0	0	0	46	3	31	19	7	0	0

PRESENCE OF KNOWN VULNERABILITIES

Product firmware security is equal to the sum of its parts. Complex systems include many different software components from a diverse set of manufacturers and open sources. Each of these components may contain publicly known vulnerabilities. The NVD maintains the CVE list which catalogs publicly known cybersecurity vulnerabilities. Vulnerabilities are reported against the hardware models, operating systems, or applications they are known to affect but this is rarely a comprehensive list. It is common for additional products, other than the products listed in a CVE, to be affected by the same vulnerability.

Our automated analysis examines product firmware and identifies the versions of component software. CVEs that apply to these components are associated to the firmware that contains them in order to provide an accurate accounting of the known vulnerabilities that may affect a given firmware version.

On average, each fully up-to-date firmware image has approximately 102 known, reported vulnerabilities just due to insecure third-party libraries included in the firmware.

The CVE system associates a vulnerability to a product by linking a CVE to one or more Common Platform Enumerations (CPE). A CPE dictionary entry encodes vendor, product, and version information in a URI string. A given CVE lists the set of CPEs it is known to affect.

Traditional approaches to tracking the impact of CVEs do not work well for embedded device firmware, because end users are unaware of the components inside the device's firmware. Firmware is a complex software system that consists of many components including software from the product vendor, third-party software vendors, and open-source software.

A vulnerability in a common open-source library like OpenSSL may be found, reported, and remediated but it is incumbent on the embedded device manufacturers to apply the update and issue a new firmware release. When a firmware consists of hundreds of third-party software components, the manufacturers must remain vigilant to ensure their products are not impacted by known vulnerabilities.

Vulnerabilities that have been disclosed but remain unpatched in firmware are devastating to product security because of the strong likelihood that public exploit code for the vulnerability exists and is being used to target vulnerable systems on the Internet

Discovering Known Vulnerabilities

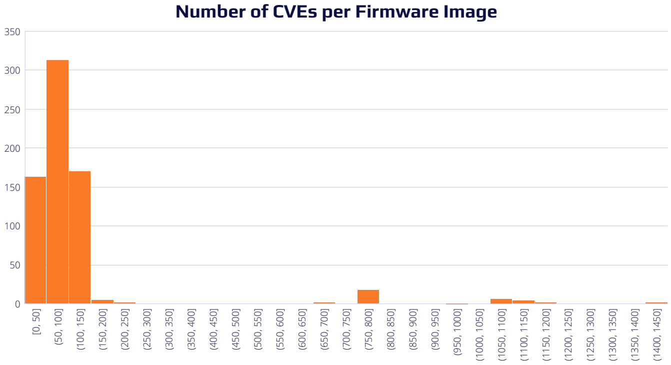
Iotasphere unpacks a product firmware image and produces a Software Bill of Materials (SBOM) based on the components used in the device firmware. CPE URIs are constructed for each entry in the SBOM and queries are issued against the CVE database for each of the CPEs. The results of those queries are rolled up to the product level in our system. The SBOM derived CVEs are combined with the product-specific CVEs and the result becomes the set of known vulnerabilities that impact a given product and firmware version. This approach allows us to construct a complete mapping of known vulnerabilities to products.

Results Summary

Our analysis found a large number of known vulnerabilities inside the *latest* versions of Huawei firmware. Many of these vulnerabilities are at high or critical severity levels. At the publish date of this report no known patch is available.

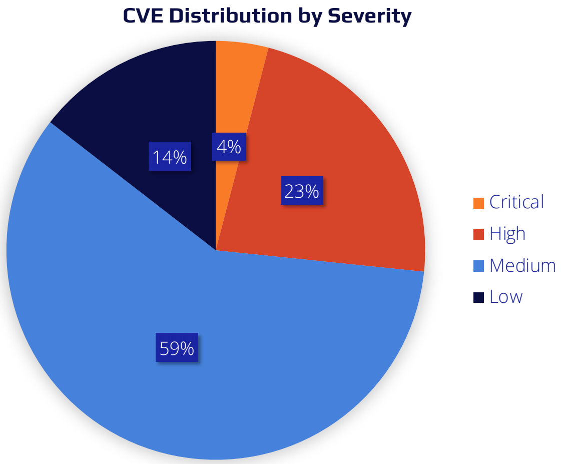
CVEs per Firmware

Huawei shares software components across multiple products and firmware versions. Vulnerabilities that apply to these shared components are present across the Huawei ecosystem. Our analysis has identified several high-risk firmware images that contain in excess of 1,400 unique CVEs. The median number of CVEs per firmware analyzed is 102. By all standards, this number is high.

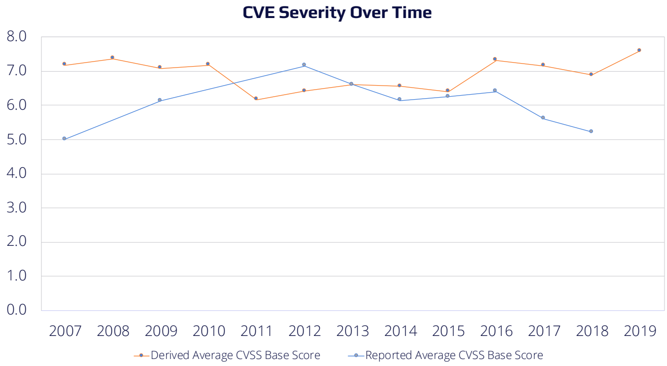


CVE Severity

A CVE is given a score from 0 to 10 according to the Common Vulnerability Scoring System (CVSS). This score is computed based on factors such as exploitability and impact of the vulnerability. The higher the CVSS score, the greater the perceived impact to security.



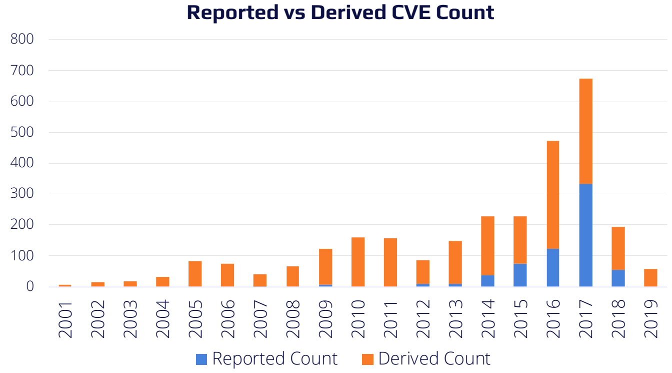
27% of the CVEs associated with Huawei firmware have a CVSS score that qualifies as 'High' (7.0-8.9) or 'Critical' (9.0-10.0). These vulnerabilities can lead to a total compromise of the affected systems. Critical vulnerabilities are given that rating because the complexity to exploit is relatively low and they often are remotely exploitable. That means an attacker just needs to be on the same network as the impacted device to take control of it.



The chart illustrates the average CVSS score per year for Huawei (reported and derived) CVEs. As security organizations and secure development processes improve, this average is expected to decline. In Huawei's case, the average severity is remaining consistent year over year.

Reported vs. Derived CVEs

The NVD tracks 655 unique CVEs that are explicitly reported against Huawei products. This number is based on a search for all CPEs with `huawei` as the vendor and includes results for categories not included as part of this report such as mobile phones and home network equipment.



The chart shows a dramatic uptick in CVEs reported against Huawei products over the past several years. When we include unique CVEs that impact subcomponents of the firmware running on Huawei products we see a more even distribution of known vulnerabilities but the recent trends still show the situation is getting worse instead of better. Note: 2018-2019 are low, because the components inside Huawei firmware are, on average, 5.36 years old.

Detailed Example

The NVD tracks 23 CVEs that apply specifically to the Huawei AR3600 series enterprise router. Our automated analysis determined an additional 1,148 CVEs apply to components contained in the V200R007C00SPCb00 firmware for the device. The additional CVE coverage includes 87 critical severity and 356 high severity CVEs. This represents a dramatic shift in the risk profile of the product. Major contributors of the additional CVE information include three versions of the Linux kernel contained within the firmware (3.10.19, 2.6.34, and 2.6.30), five versions of the OpenSSL library (0.9.7f, 0.9.8o, 1.0.0, 1.0.1e, and 1.0.1k).

CRITICAL	HIGH	MEDIUM	LOW
87	356	534	171

The versions of OpenSSL included with the firmware are susceptible to several well-known (and notoriously exploitable) vulnerabilities such as Heartbleed, DROWN, FREAK, and POODLE. The version of the Linux kernel predates patches for several critical severity, remotely exploitable, vulnerabilities such as CVE-2016-10229 which is triggered remotely via a malformed UDP packet.

Well-Known Vulnerabilities

Name	CVE	Observations	CVSS
DROWN	CVE-2016-0800	865	4.3
FREAK	CVE-2015-0204	1351	4.3
POODLE	CVE-2014-3566	1453	4.3
Heartbleed	CVE-2014-0160	737	5
Quadrouter	CVE-2016-2059	162	7.2
Quadrouter	CVE-2016-5340	162	7.2
Linux Kernel	CVE-2016-5696	282	5.8
Linux Kernel	CVE-2016-0728	282	7.2

Results Across Select Firmware

Number of Unique CVEs Applied to Firmwares	2218
Median Number of CVEs Applied to Firmware	102
Most Common CVE	CVE-2016-7055: Applied to 2692 Firmware
Number of Critical CVEs	125 unique - generating 16,142 occurrences
Highest Number of CVEs Applied to a Firmware	1419

UNDOCUMENTED AND HARD-CODED CREDENTIALS

Cybersecurity popular culture has taught us that a software "backdoor" must be something heavily obfuscated deep within the application code of a product. In reality, backdoors are often undocumented credentials and services stored in plain sight alongside the documented credentials. These include service accounts that have default passwords and can spawn a login shell. Depending on the type of service account, these default credentials can provide a remote adversary with easy access to the device. Embedded devices make locating even these shallow backdoors challenging due to the need to acquire and fully unpack the firmware for a product.

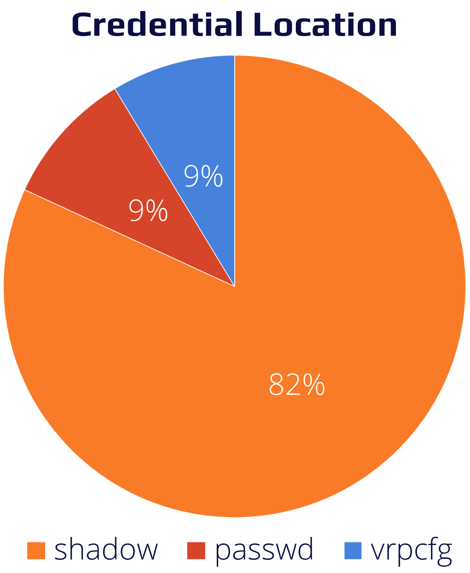
29% of all Huawei firmware analyzed had at least one default credential present.

Vodafone was criticized for a Bloomberg report[45] in April 2019 that described "hidden backdoors in Huawei equipment" they found. The security community was critical of Vodafone when the backdoor was later described as a telnet server left enabled for debugging purposes. In reality, it doesn't matter whether telnet is enabled or some sinister, custom code is installed, the results are the same: remote privileged access to Huawei devices performing critical network infrastructure tasks. Intent is the only differentiator between a maintenance function and a backdoor. We cannot easily determine Huawei's intent for these features, but we can assess the robustness of end-user documentation and configuration abilities. If a telnet server exists that cannot be disabled or is not documented in the operations manual, it is effectively a backdoor. If there are hard-coded, default credentials that a user does not or cannot change, that, too, is a backdoor.

Our automated analysis locates, extracts, and attempts to recover plaintext credentials for all accounts on the system. Having a full accounting of the credentials in a firmware leads to discovery of these potential backdoors that are hidden in plain sight.

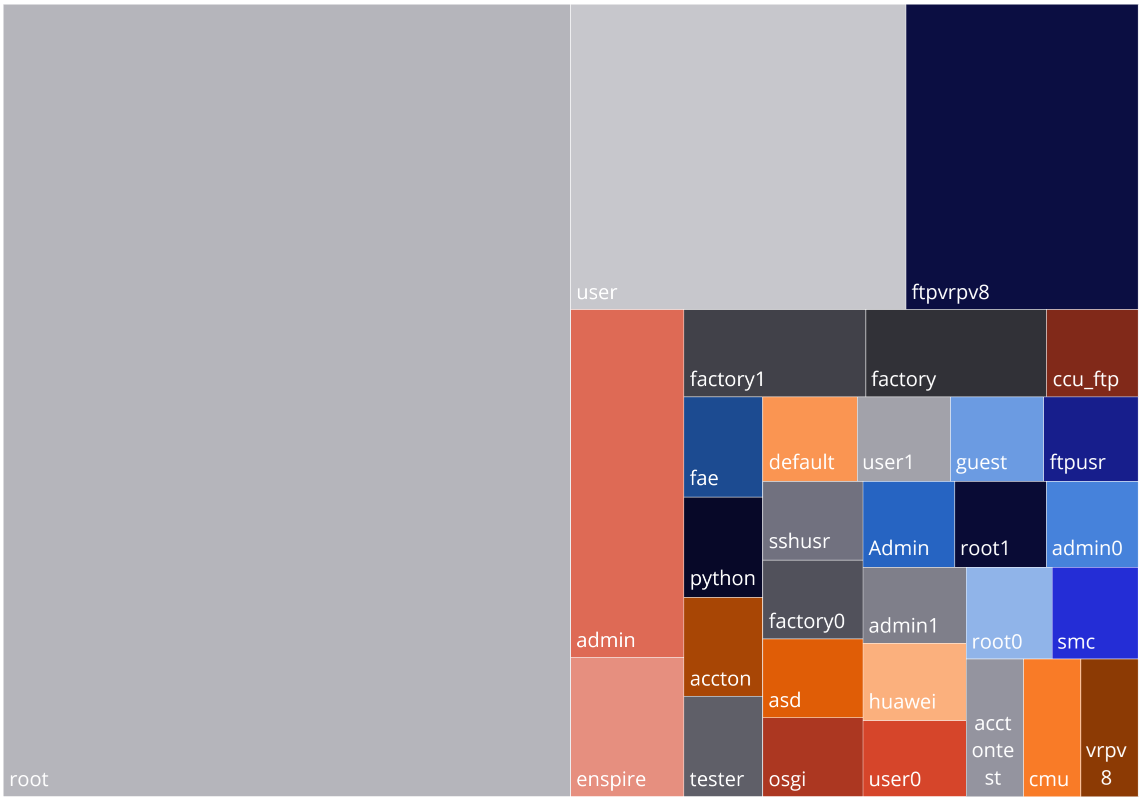
Results Summary

We performed default credential analysis on a subset of our Huawei dataset that included 1,162 firmware images. These firmware images apply to Huawei products such as service routers, enterprise switches, 4G LTE devices, IP phones, blade chassis controllers, and several other categories of equipment. Our default credential analysis of the Huawei dataset reveals that 343 firmware images (29% of analyzed firmware) contain one or more default credentials. 227 of those contain a default password for the root user. It is common for embedded devices to ship with a default password enabled for the primary account, "root" in this case, as long as the password can be changed and is documented as part of the standard operating procedure of the device.



[45] https://www.bloomberg.com/news/articles/2019-04-30/vodafone-found-hidden-backdoors-in-huawei-equipment

OBSERVATIONS OF DEFAULT CREDENTIALS IN FIRMWARE

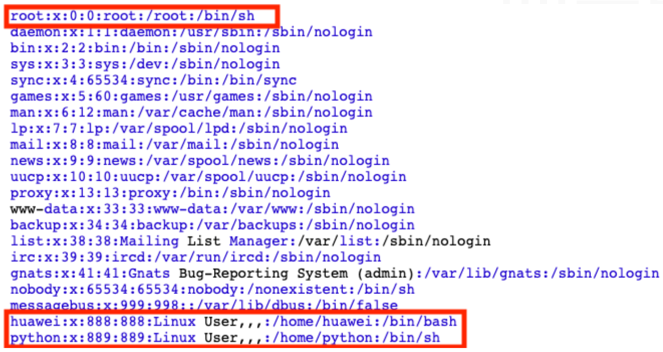


Our analysis recovered credentials from several locations on the filesystem including Linux default locations such as `/etc/passwd` and `/etc/shadow` and a Huawei-specific configuration file, `vrpcfg.cfg`. Huawei is using standard Linux-based authentication, but they have shown patterns of neglect by not locking down service accounts to the `/sbin/nologin` shell and by adding users to groups that provide root-level privileges instead of enforcing a least-privilege model for these service accounts.

The chart shows the frequency of default credentials our analysis discovered in the subset of Huawei firmwares we analyzed. The existence of a root account and instructions to secure it is often described in Huawei configuration guides. The other accounts in the table are undocumented. It is unclear if the end user is able to change the passwords associated with these accounts. The usernames associated with the accounts indicates that many of these are test accounts either left in the firmware by Huawei or by a third-party software vendor. These are exactly the type of accounts that can be leveraged to provide a malicious actor with privileged access to a device on the network.

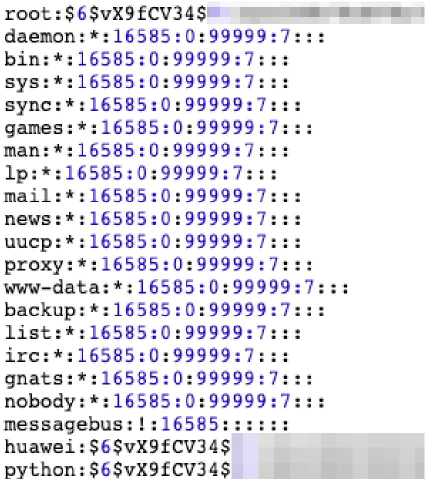
Detailed Example

Our system analyzed the Huawei AR3600 firmware version V200R007C00SPCb00 which was released on November 18, 2016. The AR3600 is advertised as an enterprise router by Huawei. The hard-coded credential analysis discovered three user accounts in the firmware that can be used to log in to the system.



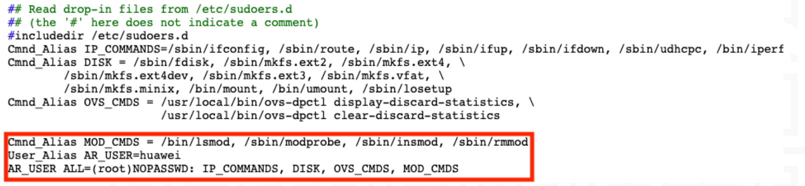
```
root:x:0:0:root:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/sbin/nologin
bin:x:2:2:bin:/bin:/sbin/nologin
sys:x:3:3:sys:/dev:/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/sbin/nologin
man:x:6:12:man:/var/cache/man:/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/sbin/nologin
mail:x:8:8:mail:/var/mail:/sbin/nologin
news:x:9:9:news:/var/spool/news:/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/sbin/nologin
proxy:x:13:13:proxy:/bin:/sbin/nologin
www-data:x:33:33:www-data:/var/www:/sbin/nologin
backup:x:34:34:backup:/var/backups:/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
messagebus:x:999:998::/var/lib/dbus:/bin/false
huawei:x:888:888:Linux User,,,:/home/huawei:/bin/bash
python:x:889:889:Linux User,,,:/home/python:/bin/sh
```

The `root` account is a privileged user (UID 0) while the `huawei` and `python` accounts are unprivileged (UID 888, 899). The root account has the same password hash as the `huawei` and `python` accounts so recovering a plaintext password for any account gives access to all three accounts.



```
root:$6$vX9fCV34$[redacted]
daemon:*:16585:0:99999:7:::
bin:*:16585:0:99999:7:::
sys:*:16585:0:99999:7:::
sync:*:16585:0:99999:7:::
games:*:16585:0:99999:7:::
man:*:16585:0:99999:7:::
lp:*:16585:0:99999:7:::
mail:*:16585:0:99999:7:::
news:*:16585:0:99999:7:::
uucp:*:16585:0:99999:7:::
proxy:*:16585:0:99999:7:::
www-data:*:16585:0:99999:7:::
backup:*:16585:0:99999:7:::
list:*:16585:0:99999:7:::
irc:*:16585:0:99999:7:::
gnats:*:16585:0:99999:7:::
nobody:*:16585:0:99999:7:::
messagebus:!:16585::::::
huawei:$6$vX9fCV34$[redacted]
python:$6$vX9fCV34$
```

The sudo configuration allows the `huawei` user to execute specific commands as a privileged user. In this example the `huawei` user is able to execute `/sbin/modprobe` and `/sbin/insmod` commands which allow that user to insert code into the kernel process of the running system.



```
## Read drop-in files from /etc/sudoers.d
## (the '#' here does not indicate a comment)
#includedir /etc/sudoers.d
Cmnd_Alias IP_COMMANDS=/sbin/ifconfig, /sbin/route, /sbin/ip, /sbin/ifup, /sbin/ifdown, /sbin/udhcpc, /bin/iperf
Cmnd_Alias DISK = /sbin/fdisk, /sbin/mkfs.ext2, /sbin/mkfs.ext4, \
        /sbin/mkfs.ext4dev, /sbin/mkfs.ext3, /sbin/mkfs.vfat, \
        /sbin/mkfs.minix, /bin/mount, /bin/umount, /sbin/losetup
Cmnd_Alias OVS_CMDS = /usr/local/bin/ovs-dpctl display-discard-statistics, \
                      /usr/local/bin/ovs-dpctl clear-discard-statistics

Cmnd_Alias MOD_CMDS = /bin/lsmod, /sbin/modprobe, /sbin/insmod, /sbin/rmmod
User_Alias AR_USER=huawei
AR_USER ALL=(root)NOPASSWD: IP_COMMANDS, DISK, OVS_CMDS, MOD_CMDS
```

This is an example of a potential privilege escalation backdoor that enables the undocumented, unpriviledged `huawei` user to execute code in the kernel.

DEFAULT AND HARD-CODED CRYPTOGRAPHIC KEYS

When looking for vulnerabilities and backdoors in devices, one of the most critical aspects is analyzing how cryptography is used within the device. Even subtle changes to the random number generator can lead to nearly undetectable access to communications.[46]

More concerning are relatively obvious security issues created by improper use of cryptographic algorithms, improper storage of cryptographic keys on devices, or improper generation of unique keys for each device. Something as trivial as choosing the wrong cipher or forgetting an initialization vector could lead to backdoor access to a device. In the world of embedded devices, it is unfortunately common to see these types of flaws.

Some of these flaws are quite obvious. For example, if a device has an `authorized_keys` file generated by the manufacturer, that is a clear backdoor for the manufacturer. Most of the time, these backdoors are created unintentionally -- they are engineering tools used during the development process. Of course, intent is hard to discern.

Our research discovered 8 unique instances of `authorized_keys` files, which can facilitate backdoor access to the device.

Results Summary

SSH keys are used as a means of identifying a user to a remote computer using public key cryptography. In this method, public keys are disseminated widely and private keys are known only to the owner. Key-based authentication is considered superior to password-based authentication because it cannot be brute-forced by submitting many possible user and password combinations. However, poor security hygiene can leave SSH servers vulnerable.

While most keys left on firmware images are likely to be benign artifacts of the build process, the presence of a private key and code referencing it suggests that keys are likely not generated on a unique basis. In that case, if a malicious user obtains a private key and access to the network via SSH, the malicious user can obtain full command-line access to the machine. For example, malicious tooling in the Mirai botnet malware scans the Internet constantly for devices listening for SSH traffic and attempts to use hard-coded credentials and cryptographic keys known to the maintainers of this malicious software. As a security measure, it is expected that any device manufacturer would not include any public or private keys within a firmware image and would automatically detect the presence of such. Yet our analysis confirmed the presence of 252 Huawei firmware images with private keys.

For devices that operate as SSH servers, a unique *host key* is expected to be maintained. This key is typically generated as part of the first initialization of the SSH server and must be unique to ensure cryptographic security. The presence of a *host key* in a build strongly indicates that all devices using this firmware image share the same key, creating the potential for man-in-the-middle attacks. Our analysis discovered 62 unique firmware images containing *host key* files.

The `authorized_keys` file, if present, lists SSH keys that can be used for remotely logging into a computer. The keys in this file are granted permanent access to the device for anyone possessing the private key. `authorized_keys` files should be maintained using extremely tight controls. The presence of this file in a firmware image indicates a method in which anyone in possession of a private key may now obtain access to all devices running that firmware image. In essence, the manufacturer (who possesses the private key) would have backdoor access to the device if it is configured to run its SSH server. Our analysis discovered 8 firmware images with such `authorized_keys` files.

It should be noted that the vulnerabilities discussed here typically require a number of factors to happen simultaneously to be exploitable. An `authorized_`

[46] https://www.theregister.co.uk/2013/09/23/rsa_crypto_warning/

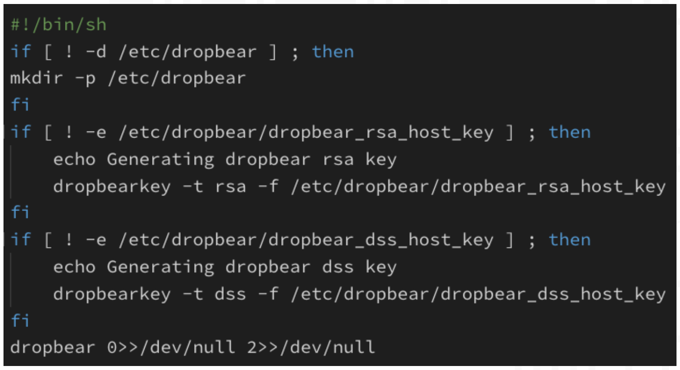
`keys` file being present on a firmware image requires an SSH configuration pointing to that file, a running SSH service on the device, and permissible network rules allowing a malicious actor to connect. SSH may very well be disabled as a service on the device and the network operator likely will block SSH access to the network operator's own management devices. In the course of adding new services or troubleshooting, an engineer may not be aware of this lurking time bomb, as the individual would likely assume that good security hygiene is being practiced by a major vendor. SSH may be turned on and network traffic permitted from additional sources to enable detailed troubleshooting. This may be safe in the right context, but it is highly risky with the presence of widely disseminated cryptographic materials.

Detailed Example

Huawei describes their SmartAX MA5800 as a "Full-service Virtualized OLT". This is a device that sits at the edge of a distribution network and multiplexes traffic over a variety of physical transport mechanisms to a passive fiber optic network. We analyzed the latest known version of firmware available for the device which was released on September 11, 2018. We detected the presence of an `authorized_keys` file for the superuser account on this firmware image. This amplifies the risk from baking this file into the firmware image, as this provides a shorter path to obtaining full control of the device to a malicious user. Further, we detected an SSH *host key* setup inside this firmware, exposing potential man-in-the-middle attacks. This example highlights that recently released firmware contains multiple risky cryptographic misconfigurations.

SSH Host Key Generation Logic Error

Several Huawei CloudEngine Series Switches and E9000 modules have a logic error in the `/etc/rcS.d/S21dropbear` initialization script that keeps the dropbear SSH RSA and DSS host keys from being dynamically generated. For the SSH host keys to be dynamically generated, the files `/etc/dropbear/dropbear_rsa_host_key` and `/etc/dropbear/dropbear_dss_host_key` must not exist. However, both files already exist in firmware images for each device. This example was observed for the CE6851HI switch in firmware version V100R005C10SPC100.



```
#!/bin/sh
if [ ! -d /etc/dropbear ] ; then
mkdir -p /etc/dropbear
fi
if [ ! -e /etc/dropbear/dropbear_rsa_host_key ] ; then
    echo Generating dropbear rsa key
    dropbearkey -t rsa -f /etc/dropbear/dropbear_rsa_host_key
fi
if [ ! -e /etc/dropbear/dropbear_dss_host_key ] ; then
    echo Generating dropbear dss key
    dropbearkey -t dss -f /etc/dropbear/dropbear_dss_host_key
fi
dropbear 0>>/dev/null 2>>/dev/null
```

TABLE 1: Results Across Select Firmware

Key Type	Firmware Images	Products
SSH RSA Private Keys	279	424
SSH RSA Public Keys	147	301
Authorized Key Files	8	47
Host Key Files	62	145

SECURE CODING PRACTICES

In the modern world, we expect all networked devices to come under attack through known and unknown vulnerabilities. Secure coding practices reduce the risk that these vulnerabilities can be successfully exploited by implementing controls designed to mitigate common software weaknesses.

The Software Engineering Institute has developed secure coding standards for commonly used platforms. Our automated analysis looks inside compiled firmware for code that executes in an insecure manner. Building on this analysis, we can review compliance with secure coding principles on individual device and firmware revisions, as well as trends over time.

Code written by software engineers is built by a compiler into binary that is machine-executable. Many modern code compilers enable protections by default for common vulnerabilities such as buffer overflow attacks and memory corruption exploits. It is our expectation that all code should be compiled with these settings enabled, as it takes a conscious effort to disable these security settings when compiling. Additionally, basic and simple security best practices dictate that engineers must keep their development tools up-to-date in order to get these inherent security advantages.

Iotasphere measures how many devices maintain a percentage of their files protected by these safeguards. No devices analyzed maintained this expectation. Further, when relaxing standards, we discovered that less than half of the Huawei devices we analyzed maintain at least 50% of their codebases compiled with these vulnerability protections.

Results Summary

Address space layout randomization (ASLR) is a memory protection solution for operating systems that was first developed in 2001. It is available on all modern operating systems and compilers as of 2011. Enabling ASLR ensures that executable code is loaded into an unpredictable location, so that when a malicious actor attempts to exploit an incorrect address location, the application can stop the attack and alert an operator. In analyzing the bulk of Huawei's firmware, we found that only 34.97% of binaries had ASLR enabled. From a security perspective, this means that the majority of firmware is likely vulnerable to relatively simplistic memory address lookup methods used during exploitation.

In many operating systems, software engineers will write code that is dependent on other code. For code that is published in the executable and linkable format (ELF), the relocation read only (RELRO) technique was developed to prevent the manipulation of these files in memory by a malicious actor. Operating systems vendors such as Red Hat commit to providing all binaries with RELRO enabled. Dismayingly, only 12.23% of the Huawei binaries analyzed are protected by RELRO, and the rest are vulnerable to memory corruption attacks RELRO should mitigate.

Data execution prevention (DEP) solutions mark sections of memory as non-executable. This means that any code introduced to these spaces by a malicious actor cannot be run. Modern operating systems used by network equipment began introducing solutions for these types of attacks in 2001 and have been widely available since 2005. Based on our analysis, only 73.96% of Huawei binaries Iotasphere analyzed have DEP enabled. This means a substantial number of devices are vulnerable to the classes of memory corruption attacks DEP is designed to prevent.

Buffer overflow attacks exploit code that allows writing to adjacent memory locations. Common programming languages used for systems and networks do not protect against this by default. However, solutions such as StackGuard can monitor for overflows and invalidating writes to adjacent memory. Based on our analysis, 26.69% of the Huawei binary files Iotasphere analyzed are protected by StackGuard. This means a substantial number of devices could be exploited via stack-buffer overflows.

Based on the pervasiveness of these secure coding vulnerabilities across enterprise and consumer grade equipment produced by Huawei, we can definitively conclude that Huawei, as an organization, does not practice secure coding principles. Given the level of documentation that exists and that these protections have been available in many compilers since the 1990s, we can conclude that Huawei's security training program has not been effective.

Detailed Example

The system analyzed the prevalence of DEP, ASLR, RELRO, and StackGuard across all ELF binary files extracted from version the latest version (V200R003C10SPC600) of firmware for Huawei SVN5600 and SVN5800 secure access gateway products. The results for this product were astonishingly poor with 0% of binaries having RELRO and DEP enabled, 25% having ASLR enabled, and 0.3% having StackGuard.

Results Across Firmware Dataset

Code Security Feature	% of Binaries with Feature
ASLR	34.97%
RELRO	12.23%
StackGuard	26.69%
DEP	73.96%

VULNERABLE CODE CONDITIONS

The most dangerous vulnerabilities in software are related to memory corruption. In most problematic cases, a segmentation fault is produced that protects the program. However, malicious actors can craft specific input values that can lead to arbitrary code execution, giving them control over the target system. Organizations ranging from Microsoft to the European Organization for Nuclear Research (CERN) maintain banned lists of functions that can trigger these conditions, and they provide tools to prevent their usage. Responsible software engineering organizations use tools and practices to prevent unsafe function usage and choose the safe alternative when it is available. Most of these alternative functions have existed for more than a decade, and experienced developers know how to use them. It is expected that telecommunications network equipment manufacturers would maintain these standards and use modern software toolchains.

In the course of our analysis, we measured how often these unsafe functions were called in Huawei's software binaries as well as third-party binaries included in firmware images. It is important to consider how security-conscious Huawei's developers are when writing code and how well they evaluate the safety of their dependencies.

Of all safe and unsafe functions calls, safe functions are used in fewer than 17% of function invocations.

Our system, Iotasphere, examines each binary executable file for known unsafe functions and determines where and how often those functions are called. In addition to the published unsafe functions, our analysis identified unsafe function wrappers in Huawei code. Identification of these function wrappers enhanced our automated analysis resulting in more comprehensive detection of unsafe function use.

Finally, our system attempts to statically determine if any unsafe function call could result in memory corruption. This approach leverages cutting edge techniques in binary analysis to uncover the most dangerous conditions within the binaries.

Results Summary

Across the entire Huawei enterprise firmware data set, we found that Huawei devices have a staggering number of unsafe and potentially exploitable code conditions. Overall, Huawei engineers and their suppliers chose to use the safe version of a function less than 17% of the time in the firmware we analyzed.

When looking deeper, the research showed an average of 21 potential memory corruptions per binary file, leading to an average of 180 potentially exploitable conditions in each firmware. This metric is a good proxy for estimating the potential for new 0-day vulnerabilities to be discovered, and due to the large numbers, these devices should be considered high risk.

Most Vulnerable Binaries in Huawei Enterprise Firmware

Filename	# Possible Memory Corruptions
brcmhbacmd	64
lanconf64e	60
hbacmd	42
autop.exe	42
libomu_mta.so	42
libinic.so	41
mapsdb	40
ConfigManApp.com	39
bootloader	37
mapsconfig	36
psql	34
cald	34
libodbchive.so.1.0.0	31

Unsafe Function Usage

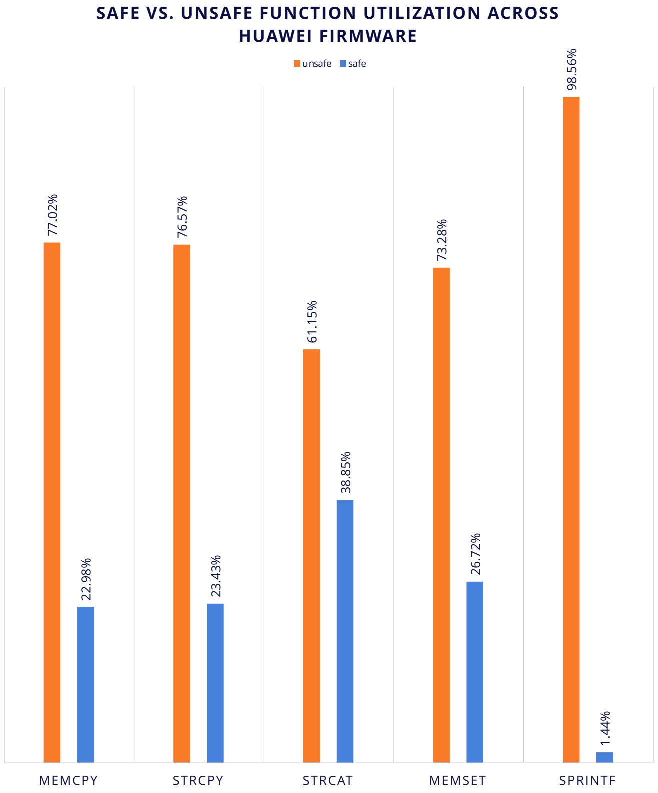
The main classes of unsafe functions are classified as memory operations, string operations, print functions, program execution, and file operations. A majority of the top 20 most commonly used unsafe functions in Huawei firmware are memory- and string-related. The use of these functions, especially when safer alternatives have been available for well over a decade, may indicate a disregard for secure development practices. It is important to note that the custom implementations of "safe" wrapper functions `safe_strncpy` and `VOS_memcopy_s` are actually *unsafe* and appear in the top 20 list. Further discussion of these unsafe wrappers follows.

Top 20 Most Commonly Used Unsafe Functions in Huawei Firmware

Function	# of Calls
memset	622,912
memcpy	232,695
puts	190,284
strncpy	162,271
fopen	85,511
strcpy	66,079
memcmp	52,146
sscanf	27,663
memmove	19,543
access	14,959
system	14,257
strcat	12,693
strncat	10,133
remove	8.485
asprintf	4,951
stpcpy	4,086
VOS_memcpy_s	3,675
longjmp	1,922
execl	1,234

Comparing Safe and Unsafe Function Usage

Our system calculated the total number of unsafe and safe function call invocations across the Huawei firmware dataset. These results were combined into collections of function types. Of all safe and unsafe functions calls, safe functions are used in just 17% of function calls. The percent usage per category are presented in the following graph. Details about the specific safe and unsafe functions analyzed are available in the table at the end of this section.

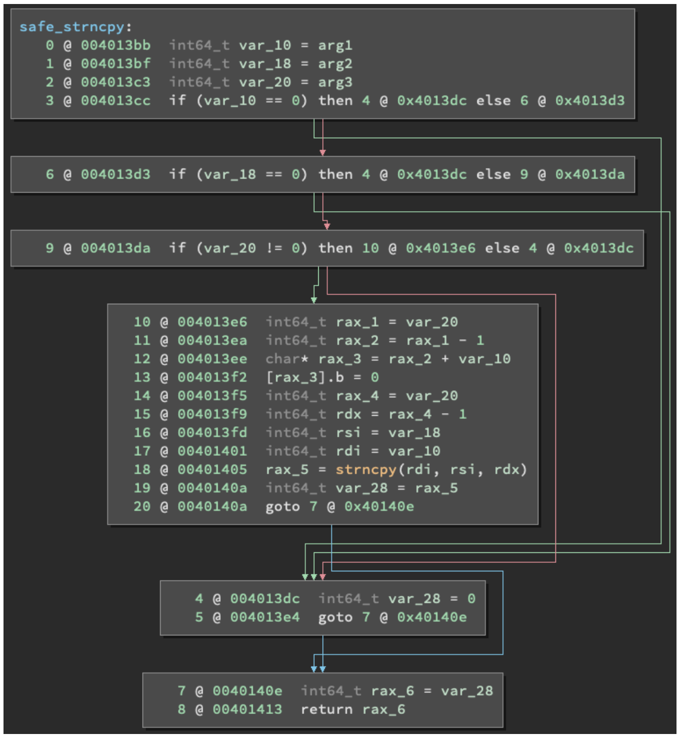


Unsafe Function Wrappers

Our analysis also revealed several unsafe function wrappers. These wrappers effectively redefine an *unsafe* function as a *safe* one, but in reality, they do not provide the requisite verification logic. Two functions that stand out are `safe_strncpy` and `VOS_memcpy_s`. These functions are referenced 53,519 and 4,814 times respectively across our dataset. Both functions are amongst the top 20 most commonly referenced unsafe functions found in the analyzed Huawei firmware.

By name, the `safe_strncpy` function appears to be a custom implementation of a safe `strncpy`. However safe_strncpy calls strncpy without properly validating all of its parameters.

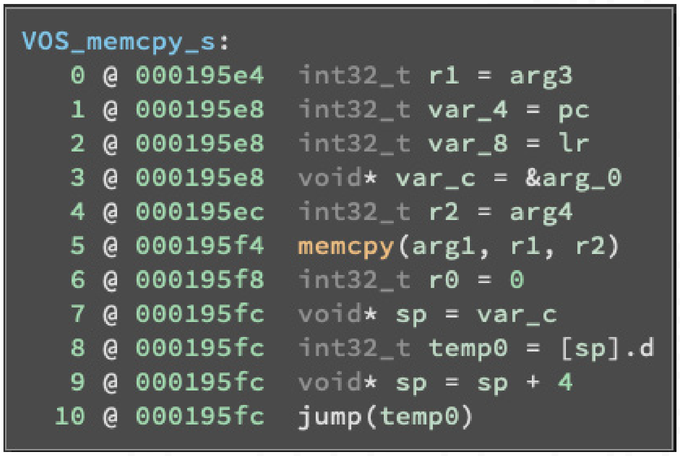
safe_strncpy takes three parameters: the destination buffer (arg1), the source buffer (arg2), , and the number of bytes to copy (arg3). The following code snippet shows that neither the size of the source buffer nor the number of bytes to copy are validated against the size of the destination buffer before use.



```
safe_strncpy:
   0 @ 004013bb  int64_t var_10 = arg1
   1 @ 004013bf  int64_t var_18 = arg2
   2 @ 004013c3  int64_t var_20 = arg3
   3 @ 004013cc  if (var_10 == 0) then 4 @ 0x4013dc else 6 @ 0x4013d3

   6 @ 004013d3  if (var_18 == 0) then 4 @ 0x4013dc else 9 @ 0x4013da

   9 @ 004013da  if (var_20 != 0) then 10 @ 0x4013e6 else 4 @ 0x4013dc

  10 @ 004013e6  int64_t rax_1 = var_20
  11 @ 004013ea  int64_t rax_2 = rax_1 - 1
  12 @ 004013ee  char* rax_3 = rax_2 + var_10
  13 @ 004013f2  [rax_3].b = 0
  14 @ 004013f5  int64_t rax_4 = var_20
  15 @ 004013f9  int64_t rdx = rax_4 - 1
  16 @ 004013fd  int64_t rsi = var_18
  17 @ 00401401  int64_t rdi = var_10
  18 @ 00401405  rax_5 = strncpy(rdi, rsi, rdx)
  19 @ 0040140a  int64_t var_28 = rax_5
  20 @ 0040140a  goto 7 @ 0x40140e

   4 @ 004013dc  int64_t var_28 = 0
   5 @ 004013e4  goto 7 @ 0x40140e

   7 @ 0040140e  int64_t rax_6 = var_28
   8 @ 00401413  return rax_6
```

Similarly VOS_memcpy_s appears to be a custom implementation for memcpy_s. However it calls memcpy without any parameter validation.

VOS_memcpy_s takes four parameters. arg1 and arg3 are pointers to destination and source buffers and arg2 and arg4 are the sizes of the of those buffers respectively. In the following code snippet arg1, arg3, and arg4 are passed directly to memcpy. arg2, the size of the buffer pointed to by arg1 is completely ignored.



```
VOS_memcpy_s:
   0 @ 000195e4  int32_t r1 = arg3
   1 @ 000195e8  int32_t var_4 = pc
   2 @ 000195e8  int32_t var_8 = lr
   3 @ 000195e8  void* var_c = &arg_0
   4 @ 000195ec  int32_t r2 = arg4
   5 @ 000195f4  memcpy(arg1, r1, r2)
   6 @ 000195f8  int32_t r0 = 0
   7 @ 000195fc  void* sp = var_c
   8 @ 000195fc  int32_t temp0 = [sp].d
   9 @ 000195fc  void* sp = sp + 4
  10 @ 000195fc  jump(temp0)
```

To use either of these functions in a safe manner, the onus is on the programmer to validate the parameters to safe_strncpy and VOS_memcpy_s to ensure the calls to strncpy and memcpy do not overflow the destination buffer. Given that the function names imply that safe_strncpy and VOS_memcpy_s are safe functions and a part of larger Huawei APIs, it is unlikely a programmer would take the extra effort to validate the parameters.

The result of this practice: there may be hundreds of potential buffer overflows due to unsafe function wrappers.

Huawei Enterprise Firmware with the Most Potential Memory Corruptions

Products	Firmware Version	# Potential Memory Corruptions
E9000, M910	V100R001C00SPC297	673
AR3600	V200R007C00SPCB00	608
E9000, MZ510	V100R001C00SPC295	476
CE12800	V200R002C50SPC800	448
AR1200	V200R007C00SPCC00	311

Safe and Unsafe Function Collections

Function Type	Unsafe Functions	Safe Functions
memcpy	memcpy, wmemcpy_sOptAsm, memcpy_sOptTc, osal_memcpy, tpms_memcpy, drv_cvb_memcpy_s_impl, vos_memcpy_s	memcpy_s, wmemcpy_s, log_memcpy_sc, drv_cvb_memcpy_s_impl, call_memcpy, tsocket_memcpy_s
memset	memset, tpmsi_memset, wmemset	memset_s, cvb_memset_s_impl, tsocket_memset_s, osal_sdk_memset_s, call_memset, log_memset_sc
strcpy	strcpy, __strcpy__null, better_strncpy,drv_cvb_strcpy_s_impl, ipsi_strcpy_s, osip_strncpy, safe_strncpy, stpcpy, stpncpy, strncpy, tpmsi_strncpy, wcpcpy, wcscpy, wps_strncpy	strncpy_s, strlcpy, strndup, strdup, log_strncpy_sc, wcscpy_
strcat	strcat, strncat, drv_cvb_strcat_s_impl	strcat_s, strlcat, wcscat_s, tsocket_strcat_s
sprintf	sprintf, vsprintf, swprintf	asprintf, vsprintf, vasprintf
scanf	scanf, sscanf, swscanf, vscanf, wscanf	scanf_s, sscanf_s, swscanf_s, vscanf_s, swprintf_s
memmove	memmove, wmemmove	memmove_s, wmemmove_
exec	execl, system, wsystem	execlp, execv, execve, execvp, execle, execvpe, fexecve

QUALITY CHANGE OVER TIME

Huawei products are complex systems built by large teams of engineers. Engineers make mistakes which can lead to product vulnerabilities. Huawei has publicly committed to improving the security of their products many times. Recently, Huawei pledged to invest 2 billion dollars to develop a comprehensive solution to improving the cybersecurity of their products.[47] With commitments like that, it is reasonable to expect to see the cybersecurity risk of their products decreasing over time.

Huawei has a poor track record of security improvement over time. The HCSEC oversight board report published in July of 2018 had this to say about Huawei's quality process: "Huawei's processes continue to fall short of industry good practice and make it difficult to provide long term assurance. The lack of progress in remediating these is disappointing."[48] The following year the report said "No material progress has been made by Huawei in the remediation of the issues reported last year."[49]

Our system assesses security across the Finite State Device Risk Matrix, which includes nine different risk categories. We can compare these factors across products or firmware versions. Using our dataset and analysis techniques, we are able to quantitatively assess a trend that shows improvement, deterioration, or stagnation over time for firmware versions that apply to the same product as well as across the brand as a whole.

Results Summary

The analysis presented in this report focused on the latest version of firmware for each device analyzed. However, we acquired an earlier version of firmware for a CE6851 network switch and compared the analysis results of the V100 firmware against the current V200 firmware. These firmware versions were released two years apart from each other, so they provide a substantial separation in time to assess the rate of change from a security perspective within this product line.

Security became quantifiably worse for users that patched their devices to the updated version of firmware.

As shown on the next page, when analyzing these firmware images across the Finite State 9-dimensional Risk Matrix, it is clear that the newer, V200, version of the firmware has worse security than the older, V100 version in most categories. The only exceptions are that the component age became lower, as expected in a new version, and code complexity and unsafe function calls decreased slightly, which are generally worthwhile investments.

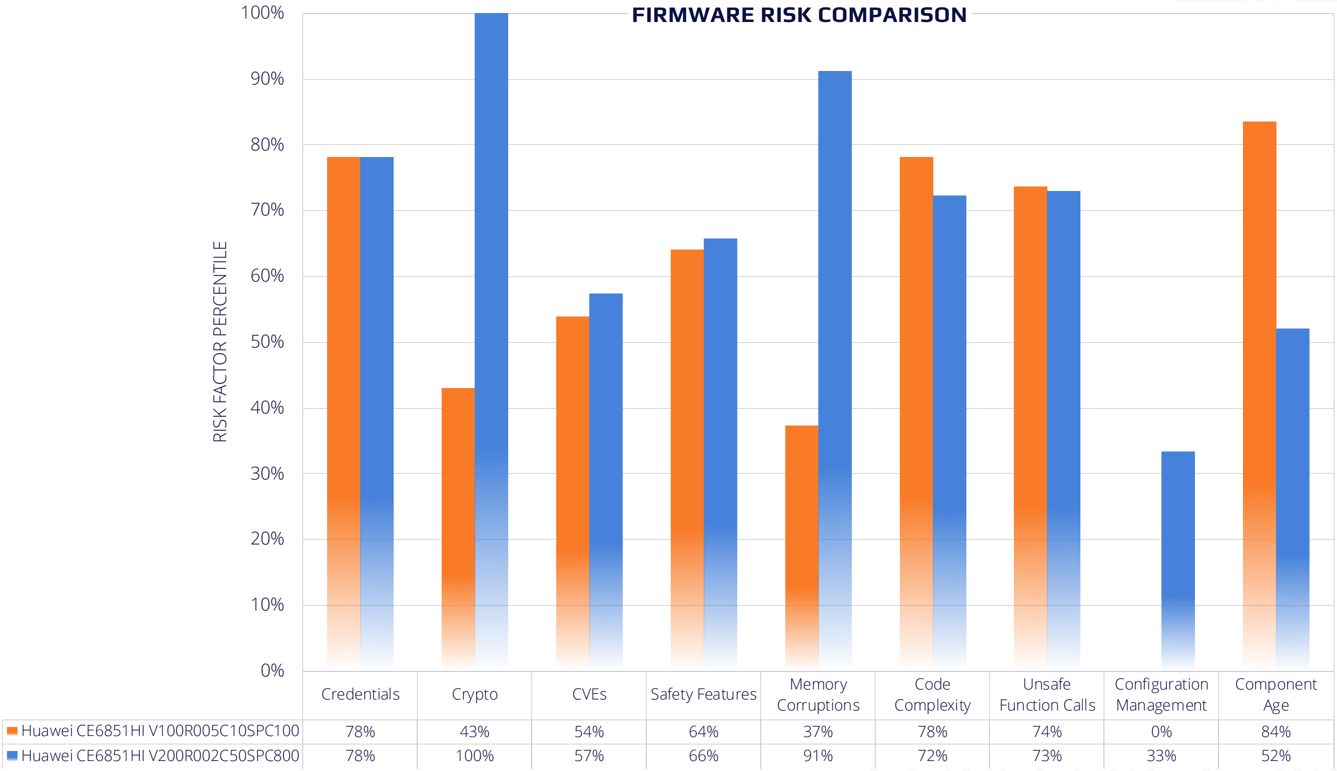
Notably, the number of known vulnerabilities (CVEs) *increased* from the older version of the firmware to the latest version. Typically, CVEs are directly correlated with the age of components used, so the increase is the result of poor technology choices within the new version. The number of possible memory corruptions skyrocketed from V100 to V200, which reinforces the increased number of known vulnerabilities.

The results of this analysis do not indicate the type of improvement we expect to see from a company who has stated they have focused efforts and investments targeted at improving their security program.

[47] *Huawei $2 billion security pledge followed walkout by British official*. Reuters. Dec. 13, 2018. https://www.reuters.com/article/us-huawei-europe-britain/huawei-2-billion-security-pledge-followed-walkout-by-british-official-sources-idUSKBN1OC23W

[48] Huawei Cyber Security Evaluation Centre (HCSEC) oversight board annual report. July 2018.
https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/727415/20180717_HCSEC_Oversight_Board_Report_2018_-_FINAL.pdf

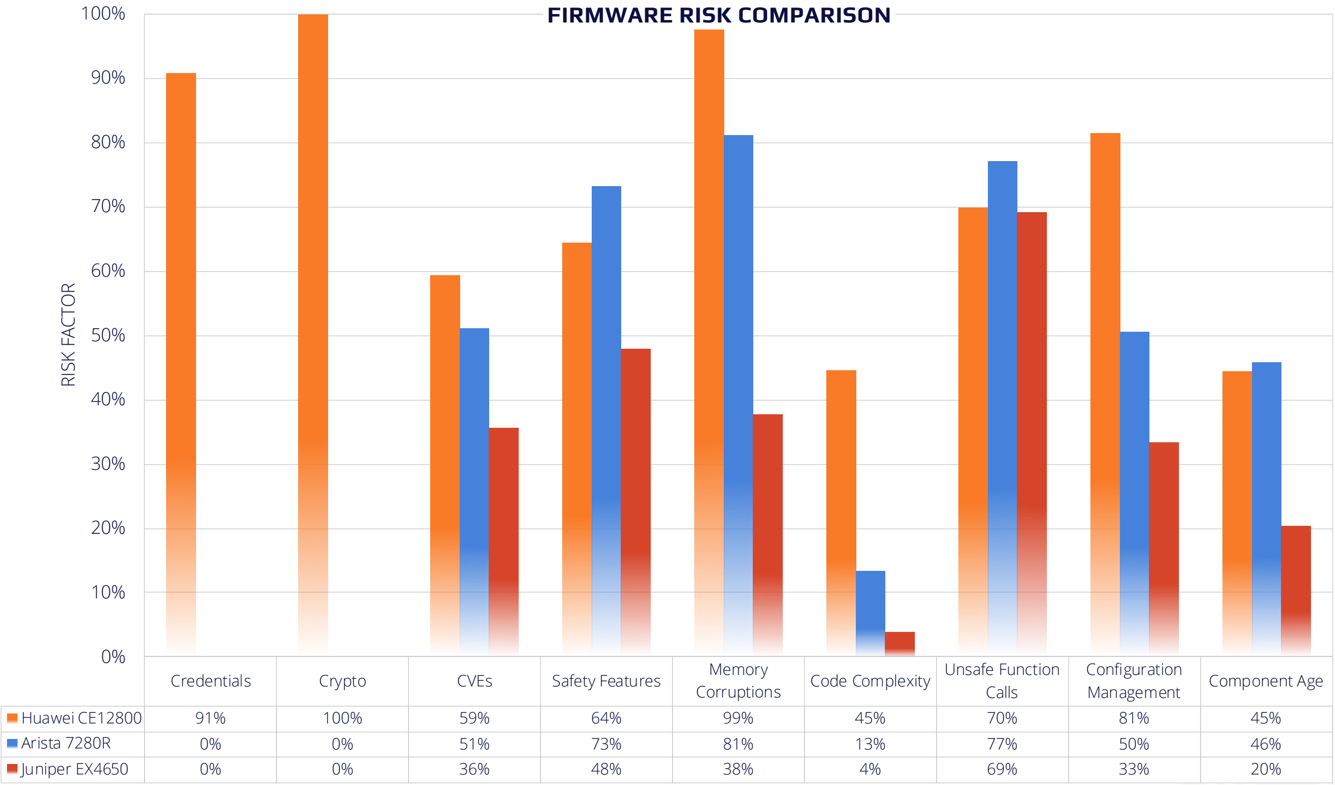
[49] Huawei Cyber Security Evaluation Centre (HCSEC) oversight board annual report. March 2019.



	Credentials	Crypto	CVEs	Safety Features	Memory Corruptions	Code Complexity	Unsafe Function Calls	Configuration Management	Component Age
🟧 Huawei CE6851HI V100R005C10SPC100	78%	43%	54%	64%	37%	78%	74%	0%	84%
🟦 Huawei CE6851HI V200R002C50SPC800	78%	100%	57%	66%	91%	72%	73%	33%	52%

Category	Description	V100R005C10SPC100	V200R002C50SPC800
Credentials	The total number of credentials with hard-coded default passwords discovered in the firmware.	78%	78%
Crypto	The total number of unique, hard-coded default SSH private keys, SSH public keys, authorized_keys files, and host keys.	43%	100%
CVEs	A sum of the total number of CVEs with each CVE weighted by its mean category CVSS score (i.e. High = 0.8).	54%	57%
Safety Features	The average of the percent of binaries without each of RELRO, ASLR, DEP, and StackGuard enabled.	64%	66%
Memory Corruptions	The total number of potential memory corruptions automatically identified in the firmware's binaries.	37%	91%
Code Complexity	The percentage of all of the functions in the firmware's binaries that have a cyclomatic complexity score above 10.	78%	72%
Unsafe Function Calls	The percentage of all function calls within the firmware where the unsafe option was used rather than the safe one.	74%	73%
Configuration Management	The number of unique occurrences of different versions of the same library within a single firmware image.	0%	33%
Component Age	The average age of third-party components based upon the release date of the detected version.	84%	52%

CASE STUDY: HUAWEI CE12800 vs. JUNIPER EX4650 vs. ARISTA 7280R



	Credentials	Crypto	CVEs	Safety Features	Memory Corruptions	Code Complexity	Unsafe Function Calls	Configuration Management	Component Age
🟧 Huawei CE12800	91%	100%	59%	64%	99%	45%	70%	81%	45%
🟦 Arista 7280R	0%	0%	51%	73%	81%	13%	77%	50%	46%
🟥 Juniper EX4650	0%	0%	36%	48%	38%	4%	69%	33%	20%

Evaluating vulnerabilities in an absolute sense is important to understand your exposure. Hackers do not generally care about how many more vulnerabilities one device has compared to another. They just look for exposures and exploit them.

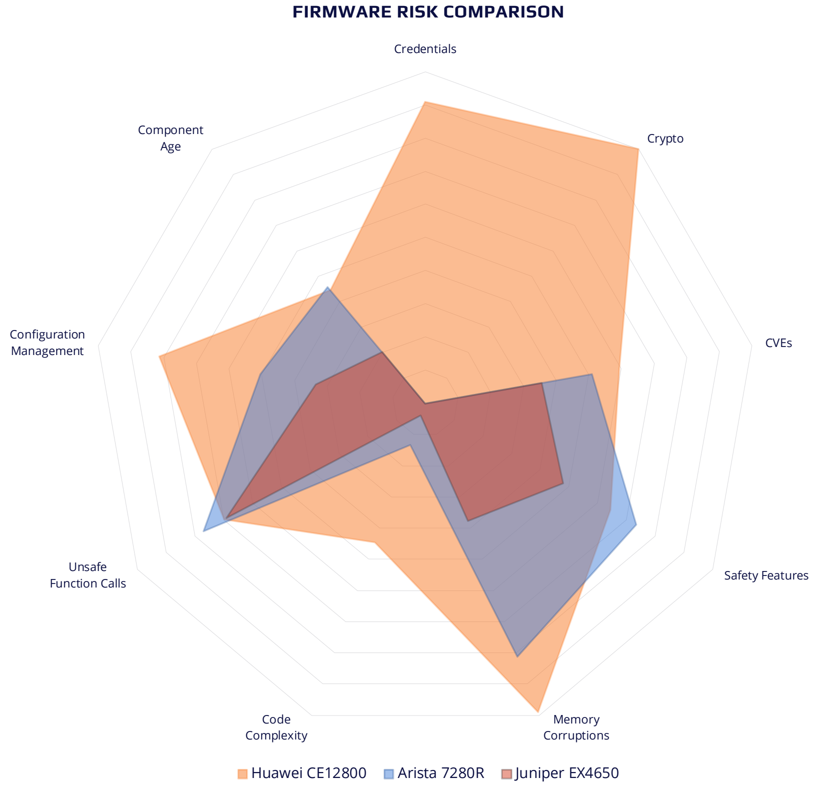
However, when analyzing cybersecurity overall, it must be viewed through the lens of risk management. Security practitioners know that you can never eliminate risk; you can only minimize it. When it comes to supply chain risk, buyers of products have to weigh their options among those that are available to them in the market. Every product has vulnerabilities. Every product has unique supply chain risks. The best supply chain security programs enable stakeholders to use *data* to make informed risk-based decisions.

To that end, this section analyzes high-end network switches from three different vendors: the Huawei CE12800, the Arista 7280R, and the Juniper EX4650.

All of these switches are equipment that might be used in a data center or in a telecommunications network (for 5G, in the Core or Multi-access Edge Computing infrastructure).

Methodology

The firmware for each of these devices was processed through Iotasphere as described in the rest of this report. Once the vulnerability and risk data was computed, it was characterized against the Finite State Device Risk Matrix, which includes nine different risk categories. For each category, the raw results were summarized and mapped against our entire firmware data set using a probability distribution. Through that distribution, the percentile of risk was computed, and higher values map to higher risks. Risks that are at or close to the 100th percentile are among the highest values we've ever seen for that category. The results are as follows.



Category	Description	Huawei CE12800	Arista 7280R	Juniper EX4650
Credentials	The total number of credentials with hard-coded default passwords discovered in the firmware.	91%	0%	0%
Crypto	The total number of unique, hard-coded default SSH private keys, SSH public keys, authorized_keys files, and host keys.	100%	0%	0%
CVEs	A sum of the total number of CVEs with each CVE weighted by its mean category CVSS score (i.e. High = 0.8).	59%	51%	36%
Safety Features	The average of the percent of binaries without each of RELRO, ASLR, DEP, and StackGuard enabled.	73%	66%	82%
Memory Corruptions	The total number of potential memory corruptions automatically identified in the firmware's binaries.	99%	81%	38%
Code Complexity	The percentage of all of the functions in the firmware's binaries that have a cyclomatic complexity score above 10.	45%	13%	4%
Unsafe Function Calls	The percentage of all function calls within the firmware where the unsafe option was used rather than the safe one.	70%	77%	69%
Configuration Management	The number of unique occurrences of different versions of the same library within a single firmware image.	81%	50%	33%
Component Age	The average age of third-party components based upon the release date of the detected version.	45%	46%	20%

Results Summary

Overall, the Huawei device incurred the highest risk rating. In all but three categories, the Huawei device had the highest risk factor, generally by a substantial margin.

In the credentials category, the analysis found three different hard-coded default credentials in the firmware, whereas the Arista and Juniper devices had none. Similarly, for cryptographic material, the Huawei device had numerous cryptographic keys embedded, including host keys.

In the CVEs category, the Huawei device had a total of 152 known CVEs (with a CVSS-weighted score of 87.9). The Arista device was second with a total of 109 CVEs (with a CVSS-weighted score of 61.7), and the Juniper device fared best with 23 CVEs (with a CVSS-weighted score of 12.6).

The Huawei CE12800 also had more than twice as many likely memory corruptions when compared to the Arista device, which itself was four times worse than the Juniper device. The code complexity metric also tells a consistent story. Complex code leads to more vulnerabilities, and unsurprisingly, the Huawei switch had substantially more complex code.

The code safety features and unsafe function calls categories were both relatively close. Juniper was the highest risk from a safety features standpoint due to having no binaries using either StackGuard or RELRO. The unsafe function calls metric was bad across the board, but the Arista device fared the worst. Unfortunately for everyone, network equipment software developers still vastly prefer to use insecure functions rather than the secure options that are available to them.

CONCLUSIONS

Huawei Devices Come with Serious Technical Security Risks

Whether end users are concerned about Huawei and the Chinese government accessing their networks or other malicious hackers, the high number of vulnerabilities in Huawei devices should be a primary driver in decision making. While we cannot prove malicious intent through a technical analysis, we can concretely state that *55% of tested devices had at least one potential backdoor*.

On average, there are 102 known vulnerabilities in a Huawei firmware image. A significant percentage of these are rated as critical or high severity. Looking deeper than just the known vulnerabilities, there is substantial evidence that 0-day vulnerabilities based upon memory corruptions are abundant in Huawei firmware. In summary, if you include known, remote-access vulnerabilities along with possible backdoors, Huawei devices appear to be at high risk of potential compromise.

It should be no surprise then, that Huawei devices fared worse than comparable devices from other vendors when compared across the Finite State Device Risk Matrix. Huawei claims to be prioritizing security investments to address some of these known issues. However, through analysis of firmware changes over time, this study suggests that the security posture of these devices is, in some respects, actually *decreasing* over time. This overall weak security posture, coupled with a lack of improvement, obviously increases security risks associated with use of Huawei devices.

From a technical supply chain security standpoint, Huawei devices are some of the worst we've ever analyzed.

Supply Chain Risk Management is More Than Just Technical Risks

The technical risks related to security vulnerabilities and possible backdoors are only part of the risk assessment process. It is also important to recognize geopolitical and legal environment related to your industry and suppliers. What are the possible consequences of a cyber attack, and what is the likelihood that a vendor in you supply chain could collaborate with an adversary?

Even if it were possible to mitigate all of the technical risks for a device upon receipt, there are still ongoing risks that firmware updates or Huawei engineers being used as part of the enterprise service agreement could make a change to devices to facilitate access or monitoring. This is a risk with *all* vendors with which you do business, but in Huawei's case, the laws in China need to be considered to understand the likelihood component of any risk calculus.

The Chinese National Intelligence Law of 2016 requires all companies "to support, provide assistance, and cooperate in national intelligence work."[50] Even if Huawei may be technically correct in saying that Chinese law doesn't explicitly "compel" the installation of backdoors, China's intelligence and counter-espionage activities tend to be so expansive that these provisions could be used to justify activities extending well beyond China's borders.

Managing the Risk

These conclusions beg the natural question: "Is it possible to manage the risk?" The answer is that it is *always* possible to manage risk if you apply enough resources to it, but the amount of resources you need to apply to adding additional security controls around these devices should factor into your buying and deployment decisions.

The next section discusses some steps that can be taken to manage these risks.

Firmware Security Verification is Possible at Scale

One of the most critical steps to managing risk, regardless of the vendor of your products, is verifying the security posture of the devices you own. Despite assertions that devices and firmware updates could not be scalably tested for security properties, we demonstrate that verification can be conducted at scale, enabling increased transparency and security.

- In a matter of hours, the Finite State Platform was able to process and analyze more than 9,936 firmware images comprised of more than 1.5 million unique files.

[50] https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/790270/HCSEC_OversightBoard-Report-2019.pdf

- Through firmware analysis, the platform was able to uncover deeper vulnerabilities than comparable vulnerability scanning tools.
- By using automated analytical tools, the end users of these devices have a mechanism to enforce security requirements upon their vendors, ultimately making networks safer for everyone.

Transparency Leads to Better Security

In just about every example in history, increased transparency directly leads to better security. The more eyes that are able to look at a device or its source code, the more likely someone will spot a defect. At Finite State, transparency is core to our mission. Increased transparency, enabled by our firmware analysis technology, can provide clarity around the true risks of devices rather than relying upon potentially politically charged accusations.

5G Can Be Deployed Securely

Huawei's Global Cyber Security and Privacy Officer John Suffolk referred to cyberspace as having become the "nervous system" of society itself. The advent of 5G will compound on this reality.

5G is not merely an upgrade on 4G – some have referred to 5G as altering the very DNA of our digital experience.[51] While the technological empowerment that 5G brings should lead to dramatic improvements for our lives and society as a whole, all the enormous potential brings with it equally enormous potential catastrophic consequences if 5G networks are insecure.

By quantitatively analyzing the risks in the equipment that makes up these networks, the owners of these networks can push their vendors to build more secure devices. It is impossible to completely eliminate the risk of a cyber attack, but with comprehensive supply chain security, continuous firmware and software verification, thoughtful risk-mitigating network design, and proper ongoing monitoring, those risks can be substantially minimized.

[51] https://www.aspistrategist.org.au/huawei-and-the-ambiguity-of-chinas-intelligence-and-counter-espionage-laws/

STEPS TO MITIGATE YOUR RISK

Even the most sophisticated cybersecurity teams will struggle to manage risk in a 5G world. As the world becomes reliant on 5G and on the IoT devices that help make up the network, organizations using this promising infrastructure will be largely dependent on the manufacturers of the infrastructure to provide security. There are practical steps you can start taking now.

1. Implement a True Supply Chain Security Program

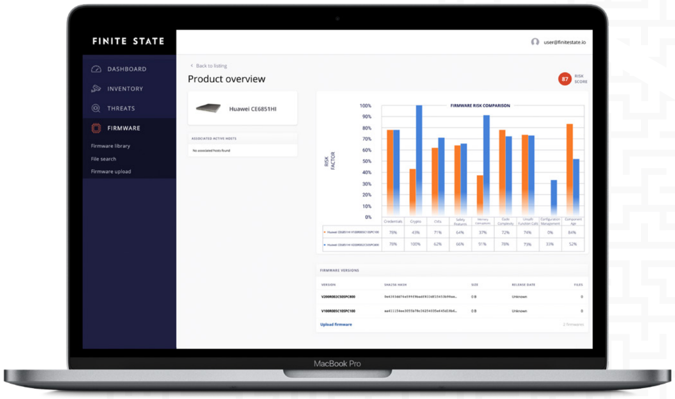
Know Your Vendors

The first step in supply chain security is simply understanding your organization's supply chain. Generate an inventory of all the devices you have, and work with procurement to understand more about each device and its own unique supply chain.

> **"[Telecom providers are] going to prefer cheaper kit if it helps them provide the service they need (absent of any other considerations). No-one currently buys telecoms' services based on how secure they are, so a company wouldn't get rewarded if they invested more than their competitors in making a more secure service."[52]**

Leverage Your Buying Power

Insist on adding language to contracts that allows you to conduct independent security testing of every device and corresponding security updates. In addition, establish channels with vendors to report your findings.



Product overview

Huawei CE6851HI

ASSOCIATED ACTIVE HOSTS

No associated hosts found

	Credentials	Crypto	CVEs	Safety Features	Memory Corruptions	Code Complexity	Unsafe Function Calls	Configuration Management	Component Age
▪ Huawei CE6851HI V100R005C10SPC100	78%	43%	71%	64%	37%	72%	74%	0%	84%
▪ Huawei CE6851HI V200R002C50SPC800	78%	100%	62%	66%	91%	78%	73%	33%	52%

FIRMWARE VERSIONS

VERSION	SHA256 HASH	SIZE	RELEASE DATE	FILES
V200R002C50SPC800	0e4303dd74e599f9badf833df15453b98ae...	0 B	Unknown	0
V100R005C10SPC100	aa411154ee3055b78e36254035e645d18b6...	0 B	Unknown	0

Upload firmware

2 firmwares

Verify Everything

Especially in critical infrastructure environments, every device should be thoroughly tested before deployment, and more importantly, the firmware should be analyzed using automated analysis tools. Vulnerability testing of most devices will report back a list of possible defects. Firmware testing will go far beyond that and provide a deep understanding of how secure the software and firmware is. While comprehensive firmware analysis was infeasible a few years ago, the technology now exists.

2. Identify

Because of the unique nature of IoT, OT, and other embedded devices, an inordinate amount of the work required to provide security is focused on visibility – that is, understanding exactly what devices are on your network and how they are configured.

With traditional IT devices, this visibility task is accomplished primarily by deploying agents inside all of your IT assets. With this inside view of the endpoints, the agents can accurately report back about OS information, installed software, patch levels, running services, etc.

Due to the black-box nature of embedded devices, this approach simply doesn't translate. While users can monitor the behaviors of devices through their network traffic, they don't have the luxury of looking inside the devices. This can be overcome by combining firmware verification with network monitoring, giving you a predictable set of behaviors and endpoint-like security models for these embedded devices.

[52] https://www.washingtonpost.com/brand-studio/wp/2018/12/14/the-dawn-of-the-5g-world/?utm_term=.e4a375528c25

Practice Continuous, Passive Scanning
Find a way to passively monitor your network in real-time rather than running periodic scans. Endeavor to see every device that joins your network and know exactly where they are without the risk of dangerous disruptions that can be caused by active network scans.

3. Mitigate

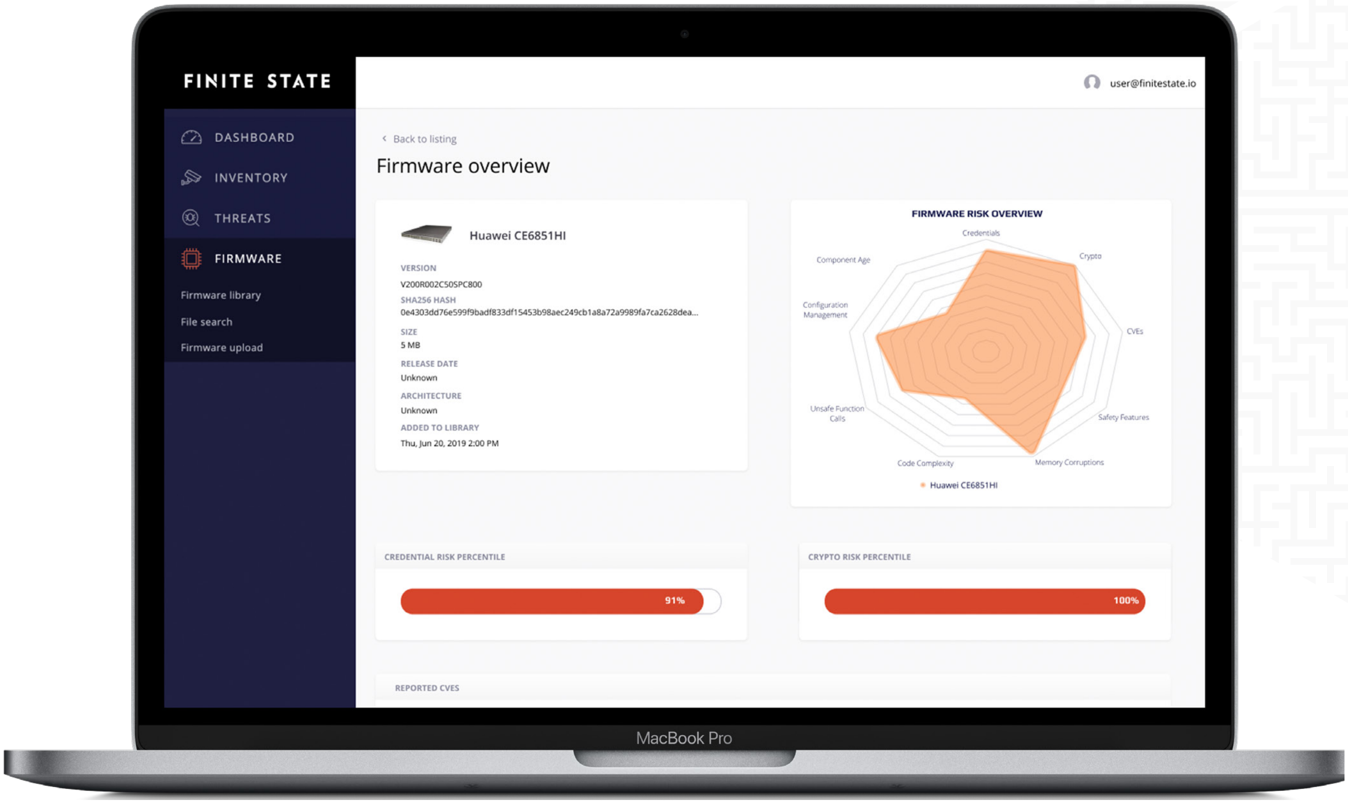
IoT and embedded devices complicate IT security and risk management. Organizations often lack a clear understanding of the actual numbers – and types – of devices within their enterprise, and their vulnerabilities as a vector into the network for cyber attacks. Even once the devices are accounted for, they may have already been compromised by an attacker.

As with other areas of cybersecurity, organizations should be using a risk management approach toward device security, and apply layers of controls that includes proper cybersecurity protocols. But, if you don't know what is on your network, or the vulnerabilities inside of those network nodes, securing your enterprise becomes a significant challenge.

Proper device security needs to incorporate both digital and physical characteristics of each device, including:

- Data collected by each device;
- Network interfaces (Ethernet, WiFi, Bluetooth, Zigbee, Z-Wave, etc.);
- Exposure to the internet;
- Physical location in your facility (i.e. in the boardroom vs. in a closet);
- Physical interfaces and actuators;
- Software vulnerabilities;
- Library vulnerabilities;
- Configuration vulnerabilities; and
- Default credentials.

Ascertaining all this information is daunting and resource draining without the proper solution. Look for a partner with a robust model for risk that leverages firmware analytics to map device details into your risk models.



Firmware overview

Huawei CE6851HI

VERSION
V200R002C50SPC800

SHA256 HASH
0e4303dd76e599f9badf833df15453b98aec249cb1a8a72a9989fa7ca2628dea...

SIZE
5 MB

RELEASE DATE
Unknown

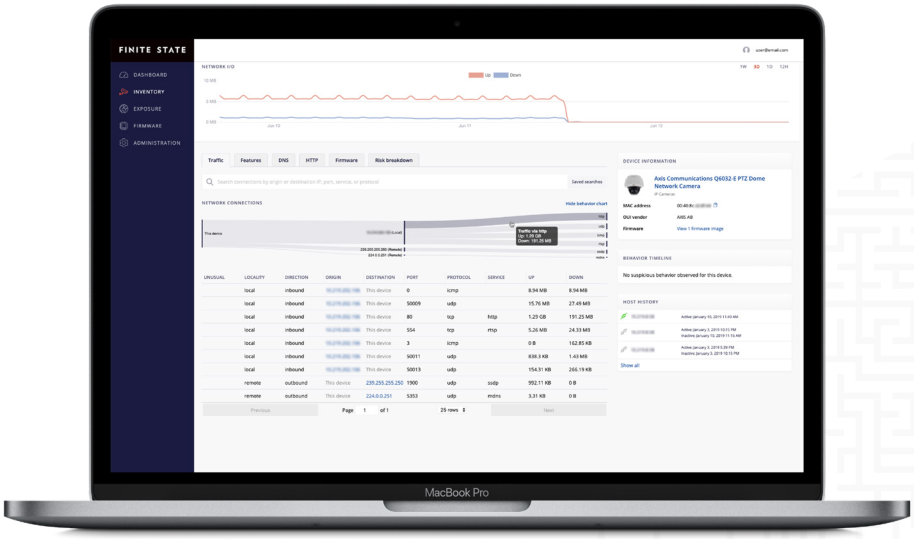
ARCHITECTURE
Unknown

ADDED TO LIBRARY
Thu, Jun 20, 2019 2:00 PM

CREDENTIAL RISK PERCENTILE
91%

CRYPTO RISK PERCENTILE
100%

REPORTED CVES



UNUSUAL	LOCALITY	DIRECTION	ORIGIN	DESTINATION	PORT	PROTOCOL	SERVICE	UP	DOWN
	local	inbound	[redacted]	This device	0	icmp		8.94 MB	8.94 MB
	local	inbound	[redacted]	This device	50009	udp		15.76 MB	27.49 MB
	local	inbound	[redacted]	This device	80	tcp	http	1.29 GB	191.25 MB
	local	inbound	[redacted]	This device	554	tcp	rtsp	5.26 MB	24.33 MB
	local	inbound	[redacted]	This device	3	icmp		0 B	162.85 KB
	local	inbound	[redacted]	This device	50011	udp		838.3 KB	1.43 MB
	local	inbound	[redacted]	This device	50013	udp		154.31 KB	266.19 KB
	remote	outbound	This device	239.255.255.250	1900	udp	ssdp	992.11 KB	0 B
	remote	outbound	This device	224.0.0.251	5353	udp	mdns	3.31 KB	0 B

Previous Page 1 of 1 25 rows Next

DEVICE INFORMATION

Axis Communications Q6032-E PTZ Dome Network Camera
IP Cameras

MAC address	00:40:8c:[redacted]
OUI vendor	AXIS AB
Firmware	View 1 firmware image

BEHAVIOR TIMELINE

No suspicious behavior observed for this device.

HOST HISTORY

[redacted]	Active: January 10, 2019 11:49 AM
[redacted]	Active: January 3, 2019 10:15 PM Inactive: January 10, 2019 11:16 AM
[redacted]	Active: January 3, 2019 5:39 PM Inactive: January 3, 2019 10:15 PM

Show all

4. Detect

One effective approach to detecting attacks is to conduct behavioral analysis of as many of the devices on your network as possible. For example, our approach is to continuously monitor the devices on our customer networks and use advanced machine learning algorithms to compare them to baseline models for that device, its firmware, and its category. Because of the unprecedented visibility into networks and the accurate inventory (something we call "device intelligence"), we can then quickly detect behaviors that are indicative of an attack, and we can do it without overwhelming security teams with false positives.

The following example illustrates the utility of this device intelligence approach. SSH traffic is prevalent on most enterprise networks. It is used to manage servers and enable remote login capabilities for numerous products. There is nothing inherently malicious about SSH. However, if we know that SSH traffic is originating from an IoT device (such as a security camera) and terminating at another device on your network, there is a major problem. IoT devices should never be 'logging in' to other endpoints on your network. It's crucial to be able to immediately identify these behaviors and be alerted in time to respond to the ongoing attack.

5. Respond

One of the biggest advantages attackers have when it comes to IoT is that even in the rare cases that they or their IoT malware is detected, there is no way to conduct a forensic analysis of the device.

Since organizations can't install forensics software on an embedded IoT device – and are lacking the tools to collect and analyze IoT files, look at running processes, or capture memory – there is no understanding that can be gained on the threat.

You should consider performing threat hunting operations within your networks that include the ability to:

- Leverage some form of device intelligence to understand what the devices on your network should be doing;
- Monitor for indicators of compromise and store historical data using traffic analysis;
- Find a solution that allows you to look inside IoT firmware the same way you would look at other endpoints on your network;
- Leverage a firmware database to detect deviations from baseline firmware images – allowing identification of installed malware; and
- Make sure your solution integrates with a NAC that enables per-device control and isolation while investigating compromise.

ABOUT FINITE STATE

Finite State provides comprehensive IoT cybersecurity for enterprise networks. With extensive backgrounds in advanced cybersecurity research and development, our team understands the intricacies of hidden risk in today's enterprise networks better than anyone. As cyber threats mount, and their impact on global security grows exponentially, we are obligated to use our skills and talents to shape a safer, smarter future.

Our mission at Finite State is to protect the next generation of networks by providing impactful security and intelligence for all of the connected devices on those networks. Transparency unambiguously improves security, and we believe that providing deep visibility to end users will incentivize manufacturers to build more secure products. The security industry itself has largely ignored this calling, shipping point-products for profit without bothering to address the systemic nature of modern vulnerabilities.

Finite State's approach is more comprehensive, not just because it's a market opportunity, but because we have a responsibility, a duty, to do this the right way, once and for all. While we focus on IoT devices, the connected devices that make up 5G networks share these same characteristics and hidden risks. IoT has become the entry point of choice for cyber attacks, and attackers have the edge in their ability to target and exploit trivial vulnerabilities in IoT firmware.

Finite State gives defenders a tactical advantage by providing deep visibility and proactive protection of every device on their network, detering even the most sophisticated actors.

Learn more about Finite State at www.finitestate.io.

APPENDIX A: DEVICES TESTED

- AR100
- AR120
- AR1200
- AR160
- AR2200
- AR3200
- AR3600
- ATN 910C
- BTS3900 GSM
- CE12800
- CE12804
- CE12804S
- CE12808
- CE12808S
- CE12812
- CE12816
- CE5810-24T4S-EI
- CE5810-48T4S-EI
- CE5850-48T4S2Q-EI
- CE5850-48T4S2Q-HI
- CE5855-24T4S2Q-EI
- CE5855-48T4S2Q-EI
- CE5880-48T6Q-EI
- CE6810-48S4Q-EI
- CE6850-48S4Q-EI
- CE6850-48S6Q-HI
- CE6850-48T4Q-EI
- CE6850-48T6Q-HI
- CE6850U-24S2Q-HI
- CE6850U-48S6Q-HI
- CE6851-48S6Q-HI
- CE6855-48S6Q-HI
- CE6855-48T6Q-HI
- CE6856-48S6Q-HI
- CE6856-48T6Q-HI
- CE6857-48S6CQ-EI
- CE6860-48S18CQ-EI
- CE6860-48S8CQ-EI
- CE6862-48S8CQ-EI
- CE6865-48S8CQ-EI
- CE6870-24S6CQ-EI
- CE6870-48S6CQ-EI
- CE6870-48T6CQ-EI
- CE6875-48S4CQ-EI
- CE6880-24S4Q2CQ-EI
- CE6880-48S4Q2CQ-EI
- CE6880-48T4Q2CQ-EI
- CE7850-32Q-EI
- CE7855-32Q-EI
- CE8850-32CQ-EI
- CE8850-64CQ-EI
- CE8861-4C-EI
- CE8868-4C-EI
- CH220 V3
- CH221
- DBS3900 GSM
- E6000 Chassis
- E9000 Chassis
- eSpace ECS
- eSpace U1911
- eSpace U1960
- eSpace U1980
- eSpace U1981
- eSpace USM
- FusionAccess
- FusionCompute
- FusionInsight
- GTSOFTX3000
- Huawei solutions for SAP HANA
- MicroDC
- NE20E-S2
- NE5000E
- NetEngine NE40E-M2
- OptiX PTN 905A
- OptiX PTN 905B
- OptiX PTN 906A

- OptiX PTN 910
- OptiX PTN 910-F
- OptiX PTN 950
- OptiX PTN 960
- RSE6500
- S1700-16G
- S1700-24-AC
- S1700-24GR
- S1700-28GR-4X
- S1700-52GR-4X
- S1700-52R-2T2P-AC
- S1700-8-AC
- S1720-10GF-2P
- S1720-10GF-PWR-2P
- S1720-10GW-2P
- S1720-10GW-PWR-2P
- S1720-20GFR-4TP
- S1720-28GFR-4P
- S1720-28GFR-4TP
- S1720-28GFR-PWR-4P
- S1720-28GWR-4P
- S1720-28GWR-4X
- S1720-28GWR-PWR-4P
- S1720-28GWR-PWR-4TP
- S1720-28GWR-PWR-4X
- S1720-52GFR-4P
- S1720-52GFR-PWR-4P
- S1720-52GWR-4P
- S1720-52GWR-4X
- S1720-52GWR-PWR-4P
- S1720-52GWR-PWR-4X
- S1720X-16XWR
- S1720X-32XWR
- S628-E
- S628-PWR-E
- S628X-E
- S628X-PWR-E
- S652-E
- S652-PWR-E
- S652X-E
- S652X-PWR-E
- SCC800
- SD100
- Secospace USG6305
- Secospace USG6305-W
- Secospace USG6310S
- Secospace USG6310S-W
- Secospace USG6310S-WL
- Secospace USG6310S-WL-OVS
- Secospace USG6510
- Secospace USG6510-WL
- SmartAX EA5801
- SmartAX MA5600T
- SmartAX MA5633
- SmartAX MA5670
- SmartAX MA5800
- SmartAX MA5871
- SMU
- SVN5600
- SVN5630
- SVN5660
- SVN5800
- SVN5830
- SVN5850
- SVN5860
- SVN5880
- U-SYS SoftX3000
- UAC3000
- UPS5000
- VP9660