FINITE 🚯 S

Understanding the Finite State Risk Profiles

Until now, IoT devices have been like black boxes, meaning that users have no control or visibility into what is running inside them. Finite State's firmware analysis, which illuminates the software and components buried deep inside IoT devices, helps security teams to properly assess the associated risk to their network.

TRANSPARENCY INTO DEVICE RISK

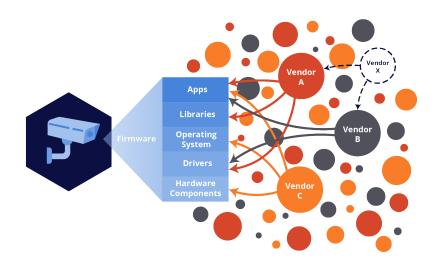
Finite State's risk model is the industry's most comprehensive. Our platform fuses passive network monitoring, firmware analysis, vulnerability data sources, exploit data sources, manufacturer disclosure statements, your own inventory management systems, threat feeds, and more to feed our data models. Further, we continuously update these risk scores and provide historical views, enabling you to understand your current, at the moment posture and how this has changed over time.

Finite State's risk model considers the most dimensions and factors, giving deep insight into the real risk of your deployed device. We can consider the risk based on placement within the network, configuration of the services on the device, the code contained within the firmware, and the type of product to understand all the possible dimensions of risk. We store all of this data and can visualize risk over time as well for each device, allowing you to better understand how your security posture is improving as you respond to our insights.

We use reverse engineering to unpack the firmware image in a device to understand key components, including overall firmware subcomponents and whether the original equipment manufacturer (OEM) follows a secure software development lifecycle.

UNDERSTANDING FIRMWARE CONTENTS

Security teams usually have no idea what is running inside their IoT devices, making it nearly impossible to properly assess risk. To make matters worse, global supply chains and lack of transparency into IoT and other connected devices leaves nearly every organization exposed to potential vulnerabilities buried inside.

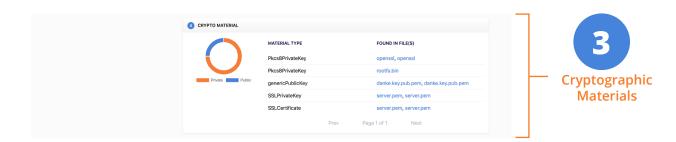


Finite State has built the world's largest firmware reverse engineering system, which has analyzed hundreds of thousands of firmware images (resulting in hundreds of millions of unique files). This approach makes it possible to understand the true risk of devices on any network based on the software that is installed, its configuration, and how the operating system as a whole is configured. To understand the risk any device may pose to the network, we look at key pieces of content in the firmware image, including the software bill of materials and hard-coded credentials and other crypto material that may be present.

FINITE 🛞 STATE

UNDERSTANDING FIRMWARE CONTENTS

	S NETWORK FIRMWARE					ALERTS user@email.com	Logout	
	Alaris PC 8015 Infusion	n Pump / 9.33	3_rootf	s_extracted				
Г	RISK OVERVIEW	2 SOFTWARE COM	PONENTS					
	Software components Potential memory consultions Credentials	Search software						
	Potencial memory corruptions	SOFTWARE		VERSIONS		UNIQUE VERSION COUNT		
	Unsafe function calls	OpenSSL		0.9.7c, (+2 more)		3		
	Code complexity Safety features	wpa_supplicant		3.4.8		1	_	
	Crypto material	ProFTPD		1.3.3		1		_
Risk	8 Software components	LinuxKernel		3.2.0		1		Software
verview	0 Credentials	iptables		1.4.17		1		of Mate
	🔞 CVEs	udhcp		1.21.0		1		
	67) Safety features	BusyBox		1.21.0		1		
	-			Prev Page	e 1 of 1 Next			
	Crypto material							
	2 Code complexity	CREDENTIALS						
	70 Unsafe function calls	PRIVILEGED USERS		DETECTED USERS	HARDCODED PASSW	ORDS DETECTED		
	Potential memory corruptions	2		44	8		44	
		Search credentials						
	FIRMWARE BREAKDOWN	USER	GID	PASSWORD HASH	PASSWORD	FOUND IN FILE(S)		
	34% Binary 23% Plain text 43% Other	adm				shadow, shadow		(2)
	SHA256 hash	backup	34	x		passwd, passwd		
	4771dc8238 🔊 Size	bin	2	×		passwd, passwd		
	22.19 MB	daemon	1	x		passwd, passwd		Hard-co
	VIEW ALL FILES IN FIRMWARE	dbus	81	×		passwd, passwd		Credent
		default	1000			shadow, shadow		creacin
		ftp	83			shadow, shadow		
		haldaemon	68	x		passwd, passwd		
		lp				shadow, shadow		
		чі				shauow, shadow		



STATE



Search software		
SOFTWARE	VERSIONS	UNIQUE VERSION COUNT
OpenSSL	0.9.7c, (+2.more)	3
pa_supplicant	3.4.8	1
oFTPD	1.3.3	1
nuxKernel	3.2.0	1
ables	1.4.17	1
hcp	1.21.0	1
usyBox	1.21.0	1

FINITE

SOFTWARE BILL OF MATERIALS

Finite State unpacks the software bill of materials, providing visibility into what's running on an IoT device, including binaries like Bash, BusyBox, Curl, dropbear, and even OpenSSL. Not only does this help inform the risk profile, but understanding the software bill of materials allows us to more positively identify products and software running on the network based on firmware-version unique characteristics.

2

CREDENTIALS				
PRIVILEGED USERS		DETECTED USER		RDS DETECTED USERS
Search credentials				
USER	GID	PASSWORD HASH	PASSWORD	FOUND IN FILE(S)
adm				shadow, shadow
backup	34	×		passwd, passwd
bin	2	x		passwd, passwd
daemon	1	x		passwd, passwd
dbus	81	x		passwd, passwd
default	1000			shadow, shadow
ftp	83			shadow, shadow
haldaemon	68	x		passwd, passwd
halt				shadow, shadow
lp				shadow, shadow
		Prev P	age 1 of 3 Next	

HARD-CODED CREDENTIALS

Automated analysis capabilities locate, extract, and attempt to recover plaintext credentials for all accounts on the system. Having a full accounting of the credentials in a firmware often leads to the discovery of potential backdoors that increase the risk to the network.



CRYPTOGRAPHIC MATERIALS

Similar to hard-coded credentials, cryptographic material contained in a firmware image is highly problematic. The presence of materials such as private keys and authorized key files can produce backdoors allowing unintended access to the device. The presence of poorly configured cryptographic settings like the presence of standardized host key files may weaken the security envelope of devices, as these should be unique per device, not common across firmware.

Worse yet, because of the supply chain for these devices, numerous devices are marketed and sold from completely different companies and contain these same cryptographic materials inside.

Note that seeing this in the firmware does not necessarily mean there was malicious intent. In most cases crypto materials are included as part of the debugging process. Unfortunately, these materials can be forgotten.



GAUGING QUALITY AND SAFETY

Firmware analysis is also critical to understanding a product manufacturer's secure software development lifecycle. Finite State analyzes key factors that indicate the relative security of any software development lifecycle by observing typical indications of secure development, including how the presence of known vulnerabilities, whether third-party software is used, the use of binary safety features, code complexity, the number of safe and unsafe function calls, and memory corruptions.

Image: construction	(C	S NETWORK 🔟 FIRMWARE				ALERTS	stephanie@finitestate.io	Logout
k i i i i i i i i i i i i i iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii		RISK OVERVIEW	72 CVES					
kinew			By severity By publish yea	ır				
View								
Verware Operation Operation <td></td> <td>Cardie comministry Safety features</td> <td>Search CVEs</td> <td></td> <td></td> <td></td> <td></td> <td></td>		Cardie comministry Safety features	Search CVEs					
Image: Second secon			CVE	SEVERITY	PUBLISH DATE	EXPLOITS	SUMMARY	
Image: Constant of the constant		8 Software components	CVE-	10	Mar 3, 2016	0	The doapr_outch function	
Image: Select features Image: Select featu		00 Credentials	CVE-	10	May 4, 2016	0	The ASN.1 implementation	
CVE C <t< td=""><td>view</td><td>-</td><td>CVE-</td><td>10</td><td>Mar 3, 2016</td><td>0</td><td>The fmtstr function in cry</td><td></td></t<>	view	-	CVE-	10	Mar 3, 2016	0	The fmtstr function in cry	
 CVP Code complexity CVF <l< td=""><td></td><td>-</td><td></td><td></td><td></td><td></td><td></td><td></td></l<>		-						
CVE <td></td> <td>-</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>		-						
CVE <td></td> <td>O Crypto material</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>		O Crypto material						
Image: series of unsafe function calls Prev P		2 Code complexity						
VEW ALL FILES IN FIRMWARE VEW ALL FILES IN FIRMWARE CVE- CVE-		70 Unsafe function calls						
34% Binary 23% Plain text 43% Other 34% Binary 23% Plain text 43% Other SHA256 hash 4771d6238 ° BINARES WITH A FEATURE ENABLED Size 22.19 MB SAFETY FEATURE BINARES WITH A FEATURE ENABLED VIEW ALL FILES IN FIRMWARE SAFETY FEATURE SAFET Y EATURE DEP 59% RELRO 0% StackGuard 0%		Potential memory corruptions						
SHA256 hash 4771d62238 (S) BINARIES WITH A FEATURE ENABLED BINARIES IN FIRMWARE Size 22.19 MB VIEW ALL FILES IN FIRMWARE BINARIES IN FIRMWARE VIEW ALL FILES IN FIRMWARE ASLR 100% DEP 59% 100% RELRO 0% 100%	L .	FIRMWARE BREAKDOWN		Pro	ev Page 1 of 8	Next		
4771d08238 ° BRARES WITH A FEATURE ENABLED BRARES NITH A FEATURE ENABLED 603 Size Safet y FEATURE PERCENT BINARIES ENABLED 603 VIEW ALL FILES IN FIRMWARE Safet y FEATURE PERCENT BINARIES ENABLED RELRO 0% StackQuard 0%		34% Binary 23% Plain text 43% Other	5 SAFETY FEATURES					
22.19 MB SAFETY FEATURE PERCENT BINARIES ENABLED VIEW ALL FILES IN FIRMWARE ASLR 100% DEP 59% RELRO 0% StackOuard 0%		4771dc8238 🗅		BLED				
DEP 59% RELRO 0% StackQuard 0%			SAFETY FEATURE	PERCENT	BINARIES ENABLED			
RELRO 0% StackOuard 0%		VIEW ALL FILES IN FIRMWARE	ASLR	100%				
StackGuard 0%			DEP	59%				
StackGuard 0%			RELRO	0%				
					ev Page 1 of 1	Next		

62 CODE COMPLEXITY			
Search files			
BINARY FILE NAME	TOTAL COMPLEXITY	MOST COMPLEX FUNCTION(S)	
libc-2.13.so	29654	_IO_vfscanf	
libc-2.13.so	29654	_IO_vfscanf	-
libcrypto.so.1.0.0	20686	bn_sqr_comba8	
libcrypto.so.1.0.0	20686	bn_sqr_comba8	
busybox	11957	FUN_0006133c	
busybox	11957	FUN_0006133c	C Com
sdcsupp	7148	FUN_0002f9a4	Com
sdcsupp	7148	FUN_0002f9a4	Com
dhd.ko	6902	FUN_000277cc	
dhd.ko	6902	FUN_000277cc	





• CRITICAL 6	•	нісн 13	• MEDIUM 54	• LOW 6
earch CVEs	SEVERITY	PUBLISH DATE	EXPLOITS	SUMMARY
CVE-	10	Mar 3, 2016	0	The doapr_outch functio
CVE-	10	May 4, 2016	0	The ASN.1 implementation
CVE-	10	Mar 3, 2016	0	The fmtstr function in cr
CVE-	10	Mar 3, 2016	0	Double free vulnerability
CVE-	10	Nov 9, 2010	0	Multiple stack-based but
CVE-	9	Dec 6, 2011	0	Use-after-free vulnerabil
CVE-	7.8	Sep 26, 2016	0	Multiple memory leaks in
CVE-	7.8	May 4, 2016	0	The asn1_d2i_read_bio fu

PRESENCE OF KNOWN VULNERABILITIES

Based on the Software Bill of Materials that includes all known third-party binaries, the operating system, and awareness of the product itself, Finite State identifies all known vulnerabilities in this software automatically. Data from these vulnerability data sources is automatically deduplicated and presented to users. Finite State also correlates information from the vulnerability database about the risk of the vulnerability with known exploit data, allowing users to understand how these vulnerabilities are being used by real-world malicious actors.

This equips you with the same level of visibility used by an attacker who is aware of all software on the device and capable of using these known vulnerabilities to link together an attack chain. One vulnerability can be used for access to the device, one can be used for privilege escalation, and so forth until the device is fully compromised.

This level of visibility into known vulnerabilities baked into a device is only possible through firmware analysis. If you look up this specific device within the National Vulnerability Database, you'll notice that is has no CVEs associated with it. But if you look INSIDE the device using the Finite State Platform, you can see that there are more than a thousand CVEs associated with the packages that are present.

SAFETY FEATURES

Another component that we look at in our analysis are the binary safety features. A binary is generally compiled from source code into machine executable code. Most modern compilers will come with safety features to prevent address lookups, buffer overflows, and things of that nature. These features in modern compilers are turned on by default—so when we see that these are not enabled on binaries, we can assume that someone has actively turned these features off. That may have been done maliciously, or it may have been done to make the existing code work. We cannot determine intent; however, we can see if these compiler level protections are turned on consistently to protect against malicious attacks.

CODE COMPLEXITY

Code complexity can help analysts understand the risk profile and stability estimations of any unit of code. This particular metric effectively looks at the number of different decisions that can be made in a unit of code. When this score is higher, there are more logical paths to follow, which means there is a higher level of difficulty to adequately test the software. Software that is more difficult to test has been shown in many studies to have a higher risk of defects, which correlates with security vulnerabilities. Simply put, simpler code is more secure.



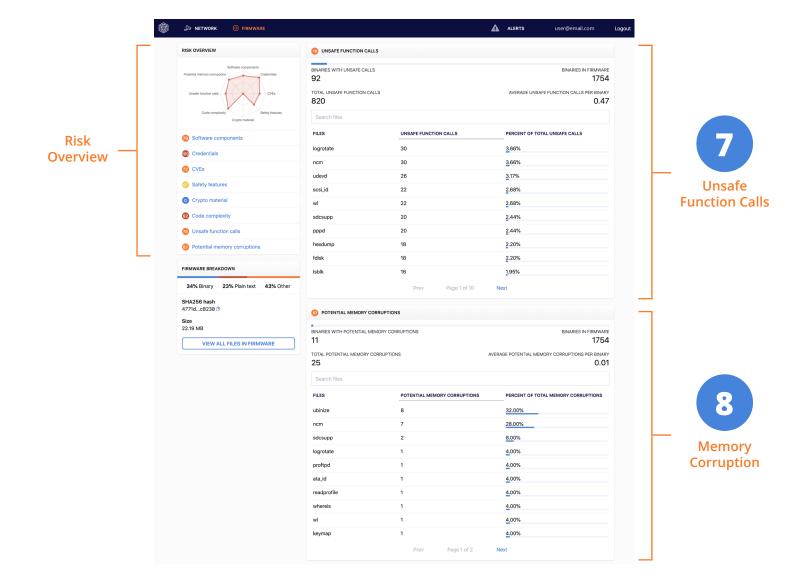
BINARIES WITH A FEATURE ENA	BLED	BINARIES IN FIRMWARE 603
SAFETY FEATURE	PERCENT BINARIES ENABLED	
ASLR	100%	
DEP	59%	
RELRO	0%	
StackGuard	0%	

6

Search files		
INARY FILE NAME	TOTAL COMPLEXITY	MOST COMPLEX FUNCTION(S)
bc-2.13.so	29654	_IO_vfscanf
ibc-2.13.so	29654	_IO_vfscanf
libcrypto.so.1.0.0	20686	bn_sqr_comba8
libcrypto.so.1.0.0	20686	bn_sqr_comba8
busybox	11957	FUN_0006133c
busybox	11957	FUN_0006133c
sdcsupp	7148	FUN_0002f9a4
sdcsupp	7148	FUN_0002f9a4
dhd.ko	6902	FUN_000277cc
dhd.ko	6902	FUN_000277cc



GAUGING QUALITY AND SAFETY



STATE



BINARIES WITH UNSAFE CALLS		BINARIES IN FIRMWAR 1754
TOTAL UNSAFE FUNCTION CALLS		AVERAGE UNSAFE FUNCTION CALLS PER BINAR 0.47
Search files		
FILES	UNSAFE FUNCTION CALLS	PERCENT OF TOTAL UNSAFE CALLS
logrotate	30	3.66%
ncm	30	3.66%
udevd	26	<u>3</u> .17%
scsi_id	22	2.68%
W	22	2.68%
sdcsupp	20	2.44%
pppd	20	2.44%
hexdump	18	<u>2</u> .20%
fdisk	18	2.20%
Isblk	16	1.95%

FINITE

UNSAFE FUNCTION CALLS

In programming languages like C, there are a series of legacy functions like strcpy that are considered unsafe and have modern analogs like strncpy. These legacy functions have long been known to be insecure for many years. The secure functions, in many cases, have been available for more than a decade. Finite State identifies the first- and third-party binaries being used and whether they use these functions. If manufacturers include numerous unsafe function calls in their own code, that suggests they are struggling to build secure software. The presence of unsafe function calls in third-party code is also likely a string indicator of how thoroughly they vet and maintain third-party tools. Together, this metric provides a better understanding of the priority level given to security throughout the software development lifecycle.

MEMORY CORRUPTION

A memory corruption is a type of vulnerability that may occur when memory is altered without an explicit assignment, meaning that the items stored at that memory location can be modified. Finite State has developed analysis tools to automatically find previously unknown 0-day memory corruption vulnerabilities, allowing us to understand how well the software development team implemented memory management practices, as well as what unknown memory corruption vulnerabilities live within the software.

TRANSPARENCY IMPROVES SECURITY

With years of offensive cyber operations experience, our team understands that attackers know more about your devices than you do. They gain this knowledge by looking inside the firmware, and they find trivially exploitable vulnerabilities. The Finite State Platform has been designed from the ground up to enable you to look deep inside the devices on your network, gain an in-depth understanding of the risks buried inside the firmware, and establish a new era of transparency. Transparency undeniably improves security, and by leveraging our unique capabilities in firmware reverse engineering, comprehensive risk modeling, and advanced detection models, you can stay ahead of attackers for the first time.

8

BINARIES WITH POTENTIAL M 11	EMORY CORRUPTIONS		BINARIES IN FIRMWAF
TOTAL POTENTIAL MEMORY (CORRUPTIONS		AVERAGE POTENTIAL MEMORY CORRUPTIONS PER BINAR 0.0
Search files			
FILES	POTENTIAL MEMORY	CORRUPTIONS	PERCENT OF TOTAL MEMORY CORRUPTIONS
ubinize	8		32.00%
ncm	7		28.00%
sdcsupp	2		8.00%
logrotate	1		4.00%
proftpd	1		4.00%
ata_id	1		4.00%
readprofile	1		4.00%
whereis	1		4.00%
wl	1		4.00%
keymap	1		4.00%