Assessing the Impact of Interface Vulnerabilities in Compartmentalized Software

*Hugo Lefeuvre*¹, Vlad-Andrei Badoiu², Yi Chien³, Felipe Huici⁴, Nathan Dautenhahn³, Pierre Olivier¹ ¹The University of Manchester, ²University Politehnica of Bucharest, ³Rice University, ⁴Unikraft.io

NDSS'23, 28th February 2023, San Diego, CA



Software compartmentalization =

Software compartmentalization =

Decompose software into lesser-privileged components

Software compartmentalization =

Decompose software into lesser-privileged components Components only have access to *what they need to do their job*

One process	
Component	
Rest of the app	

Any application

Software compartmentalization =

Decompose software into lesser-privileged components Components only have access to *what they need to do their job*



Any application

Software compartmentalization =



Software compartmentalization =



Software compartmentalization =



Software compartmentalization =



Software compartmentalization =

Decompose software into lesser-privileged components



Software compartmentalization =

Decompose software into lesser-privileged components



Software compartmentalization =

Decompose software into lesser-privileged components



Software compartmentalization =

Decompose software into lesser-privileged components



Software compartmentalization =

Decompose software into lesser-privileged components



Software compartmentalization =

Decompose software into lesser-privileged components

Components only have access to what they need to do their job



Goals of these works: compartmentalization of **legacy** software ... with a **low engineering effort** ... at a **low performance cost**

• Things are not as easy as they seem:

• Things are not as easy as they seem:



Compartment 2 (trusted)		
api(void* ptr)		
<pre>resize(int size)</pre>		
open(<i>FILE* f</i>)		

• Things are not as easy as they seem:



• Things are not as easy as they seem:



• Things are not as easy as they seem:



• Things are not as easy as they seem:

Once separation is enforced, cross-component interfaces become the attack surface



• Dereference of corrupted pointer

 Usage of corrupted indexing information

```
void resize(int size) {
    // ...
    var = matrix[size];
    // ...
}
```

• Things are not as easy as they seem:

Once separation is enforced, cross-component interfaces become the attack surface



• Dereference of corrupted pointer

- Usage of corrupted indexing information
- Usage of corrupted object

void	<pre>open(FILE* f) {</pre>
//	\dots
//	
}	

• Things are not as easy as they seem:

Once separation is enforced, cross-component interfaces become the attack surface



• Dereference of corrupted pointer

- Usage of corrupted indexing information
- Usage of corrupted object
- ... and many others!

• Things are not as easy as they seem:

Once separation is enforced, cross-component interfaces become the attack surface

We unify these vulnerabilities as:



• Things are not as easy as they seem:

Once separation is enforced, cross-component interfaces become the attack surface

We unify these vulnerabilities as:



• Things are not as easy as they seem:

Once separation is enforced, cross-component interfaces become the attack surface

We unify these vulnerabilities as:



LOTS of them in unmodified components.

• Things are not as easy as they seem:

Once separation is enforced, cross-component interfaces become the attack surface

We unify these vulnerabilities as: Compartment Interface Vulnerabilities (CIVs)



LOTS of them in unmodified components. Affect all compartmentalization frameworks to various degrees. 27

CIVs = Vulnerabilities arising due to *lack of* or *improper* Control and Data flow validation at compartment boundaries

CIVs = Vulnerabilities arising due to *lack of* or *improper* Control and Data flow validation at compartment boundaries

Classes of CIVs...

Data Leakages	Data Corruption	Temporal Violations

CIVs = Vulnerabilities arising due to *lack of* or *improper* Control and Data flow validation at compartment boundaries

Classes of CIVs...

Data Leakages	Data Corruption	Temporal Violations
Exposure of addresses		
 Exposure of compartment- confidential data 		

CIVs = Vulnerabilities arising due to *lack of* or *improper* Control and Data flow validation at compartment boundaries

Classes of CIVs...

Data Leakages	Data Corruption	Temporal Violations
 Exposure of addresses Exposure of compartment- confidential data 		

Data over-sharing, sharing of uninitialized memory (incl. compiler-added padding)

CIVs = Vulnerabilities arising due to *lack of* or *improper* Control and Data flow validation at compartment boundaries

Classes of CIVs...

Data Leakages	Data Corruption	Temporal Violations
 Exposure of addresses Exposure of compartment- confidential data 	 Dereference of corrupted pointer Usage of corrupted indexing information Usage of corrupted object 	

Data over-sharing, sharing of uninitialized memory (incl. compiler-added padding)

CIVs = Vulnerabilities arising due to *lack of* or *improper* Control and Data flow validation at compartment boundaries

Classes of CIVs...

Data Leakages	Data Corruption	Temporal Violations
 Exposure of addresses Exposure of compartment- confidential data 	 Dereference of corrupted pointer Usage of corrupted indexing information Usage of corrupted object 	
Data averable size a basis a of substitutions	the second test of the second second second be sufficient.	

Data over-sharing, sharing of uninitialized memory (incl. compiler-added padding)

Usage of interface-crossing data without appropriate sanitization

CIVs = Vulnerabilities arising due to *lack of* or *improper* Control and Data flow validation at compartment boundaries

Classes of CIVs...

memory (incl. compiler-added padding)

Data Leakages	Data Corruption	Temporal Violations
 Exposure of addresses Exposure of compartment- confidential data 	 Dereference of corrupted pointer Usage of corrupted indexing information Usage of corrupted object 	 Expectation of API usage ordering Usage of corrupted synchronization primitive Shared memory TOCTOU
Data over-sharing, sharing of uninitialized	Usage of interface-crossing data without	

appropriate sanitization

CIVs = Vulnerabilities arising due to *lack of* or *improper* Control and Data flow validation at compartment boundaries

Classes of CIVs...

Data Leakages	Data Corruption	Temporal Violations
 Exposure of addresses Exposure of compartment- confidential data 	 Dereference of corrupted pointer Usage of corrupted indexing information Usage of corrupted object 	 Expectation of API usage ordering Usage of corrupted synchronization primitive Shared memory TOCTOU
Data over-sharing, sharing of uninitialized memory (incl. compiler-added padding)	Usage of interface-crossing data without appropriate sanitization	Many causes, missing copies, double fetches, lack of enforcement of API semantics,

CIVs = Vulnerabilities arising due to *lack of* or *improper* Control and Data flow validation at compartment boundaries

Classes of CIVs...

Data Leakages	Data Corruption	Temporal Violations
 Exposure of addresses Exposure of compartment- confidential data 	 Dereference of corrupted pointer Usage of corrupted indexing information Usage of corrupted object 	 Expectation of API usage ordering Usage of corrupted synchronization primitive Shared memory TOCTOU
Data over-sharing, sharing of uninitialized memory (incl. compiler-added padding)	Usage of interface-crossing data without appropriate sanitization	Many causes, missing copies, double fetches lack of enforcement of API semantics,

Full taxonomy of CIVs in our paper!
Research questions:

• How many CIVs are there at legacy, unported APIs?

- How many CIVs are there at legacy, unported APIs?
- Are all APIs similarly affected by CIVs? (e.g., library v.s. module APIs)

- How many CIVs are there at legacy, unported APIs?
- Are all APIs similarly affected by CIVs? (e.g., library v.s. module APIs)
- How hard are these CIVs to address when compartmentalizing?

- How many CIVs are there at legacy, unported APIs?
- Are all APIs similarly affected by CIVs? (e.g., library v.s. module APIs)
- How hard are these CIVs to address when compartmentalizing?
- How bad are they? i.e., if you don't fix them, what can attackers do?

Research questions: Need to understand these to achieve adequate counter-measures

- How many CIVs are there at legacy, unported APIs?
- Are all APIs similarly affected by CIVs? (e.g., library v.s. module APIs)
- How hard are these CIVs to address when compartmentalizing?
- How bad are they? i.e., if you don't fix them, what can attackers do?

Research questions: Need to understand these to achieve adequate counter-measures

- How many CIVs are there at legacy, unported APIs?
- Are all APIs similarly affected by CIVs? (e.g., library v.s. module APIs)
- How hard are these CIVs to address when compartmentalizing?
- How bad are they? i.e., if you don't fix them, what can attackers do?

This work's approach:

Research questions: Need to understand these to achieve adequate counter-measures

- How many CIVs are there at legacy, unported APIs?
- Are all APIs similarly affected by CIVs? (e.g., library v.s. module APIs)
- How hard are these CIVs to address when compartmentalizing?
- How bad are they? i.e., if you don't fix them, what can attackers do?

This work's approach:

• Design a **fuzzer / tool specialized to find CIVs**: *ConfFuzz*

Research questions: Need to understand these to achieve adequate counter-measures

- How many CIVs are there at legacy, unported APIs?
- Are all APIs similarly affected by CIVs? (e.g., library v.s. module APIs)
- How hard are these CIVs to address when compartmentalizing?
- How bad are they? i.e., if you don't fix them, what can attackers do?

This work's approach:

- Design a **fuzzer / tool specialized to find CIVs**: *ConfFuzz*
- Apply it at scale to many applications and library interfaces to gather a data set of real-world CIVs

Research questions: Need to understand these to achieve adequate counter-measures

- How many CIVs are there at legacy, unported APIs?
- Are all APIs similarly affected by CIVs? (e.g., library v.s. module APIs)
- How hard are these CIVs to address when compartmentalizing?
- How bad are they? i.e., if you don't fix them, what can attackers do?

This work's approach:

- Design a **fuzzer / tool specialized to find CIVs**: *ConfFuzz*
- Apply it at scale to many applications and library interfaces to gather a data set of real-world CIVs
- Study, systematize, patternize the resulting data set

High-Level Overview:

 Instrument application to intercept cross compartment function calls o



- Instrument application to intercept cross compartment function calls o
 - Based on Intel Pin (DBI)



- Instrument application to intercept cross compartment function calls o
 - Based on Intel Pin (DBI)
- Interface (boundaries, types) is automatically detected using binary debug (DWARF) information •



- Instrument application to intercept cross compartment function calls o
 - Based on Intel Pin (DBI)
- Interface (boundaries, types) is automatically detected using binary debug (DWARF) information •
- A fuzzing monitor drives the exploration with custom CIV mutations ^o



- Instrument application to intercept cross compartment function calls o
 - Based on Intel Pin (DBI)
- Interface (boundaries, types) is automatically detected using binary debug (DWARF) information •
- A fuzzing monitor drives the exploration with custom CIV mutations ^o
- The workload is application-specific (benchmark, test suite, etc.)



- Instrument application to intercept cross compartment function calls o
 - Based on Intel Pin (DBI)
- Interface (boundaries, types) is automatically detected using binary debug (DWARF) information •
- A fuzzing monitor drives the exploration with custom CIV mutations ^o
- The workload is application-specific (benchmark, test suite, etc.)
- The fuzzing monitor automatically triages and stores crash reports ^o



Study results: Overview

Using ConfFuzz we gathered a substantial data set

тм	Application	Compartment API	References	Cr	ashes	Victims	API	Coverage	I	mpact (of w	hich arb		
1 1 1	Application	Compartment AF1	Kelefences	Raw	Dedup.	vicunis	Callers	Coverage	Read	Write	Exec	Alloc	Null
	иттра	libmarkdown	[42]	192	13	3	1	100% (4/4)	10 (8)	7 (7)	0 (0)	1	4
	ппги	mod_markdown		381	71	5	1	100% (1/1)	62 (52)	17 (14)	2 (1)	0	30
	aspell	libaspell		278	8	1	1	34% (48/141)	7 (7)	7 (7)	2 (1)	0	3
	bind9	libxml2 (write API)		0	0	0	1	86% (13/15)	0 (0)	0 (0)	0 (0)	0	0
	bzip2	libbz2	[67], [5]	16	5	1	1	62% (5/8)	5 (2)	1 (0)	0 (0)	0	0
	cURL	libnghttp2		61	7	2	1	50% (18/36)	3 (3)	5 (5)	0 (0)	1	3
	exif	libexif		400	7	1	1	10% (13/129)	3 (3)	0 (0)	0 (0)	0	5
		libavcodec		316	20	3	4	31% (19/60)	13 (12)	12 (12)	0 (0)	3	7
	FFmpeg	libavfilter		51	1	1	2	12% (2/16)	1 (1)	0 (0)	0 (0)	0	1
		libavformat		217	9	2	3	52% (10/19)	8 (7)	1 (1)	0 (0)	0	7
	file	libmagic		150	5	1	1	63% (7/11)	5 (2)	1 (1)	0 (0)	0	4
	ait	libcurl	[22]	13	4	2	1	90% (18/20)	2 (2)	2 (2)	0 (0)	1	1
	gn	libpcre		81	2	1	1	44% (8/18)	2 (2)	0 (0)	0 (0)	2	0
	Inkecane	libpng	[67]	66	3	1	1	46% (14/30)	2 (1)	2 (2)	0 (0)	0	1
roq	шкэсарс	libpoppler	[16]	81	4	2	1	100% (9/9)	4 (3)	4 (4)	0 (0)	0	2
lpu	libxml2-tests	libxml2 (write API)		0	0	0	1	100% (47/47)	0 (0)	0 (0)	0 (0)	0	0
Sa	lighttpd	mod_deflate		117	26	2	1	100% (6/6)	16 (11)	5 (0)	1 (1)	2	9
	Image	libghostscript	[5]	67	14	2	1	100% (11/11)	4 (2)	1 (1)	0 (0)	3	9
	Magick	libpng	[67]	778	44	1	2	22% (17/77)	2 (2)	9 (9)	2 (0)	2	39
	Wagiek	libtiff	[67]	197	14	2	1	30% (13/43)	3 (3)	6 (6)	0 (0)	0	13
	Nginy	libpcre		144	10	1	1	93% (14/15)	8 (7)	3 (3)	0 (0)	6	2
	Ngilix	mod_geoip	[52]	276	25	2	1	35% (5/14)	21 (17)	4 (1)	1 (1)	1	10
	Okular	libmarkdown	[42]	64	5	3	1	100% (4/4)	3 (1)	0 (0)	0 (0)	1	2
	Okulai	libpoppler	[16]	195	9	1	1	6% (24/379)	8 (6)	7 (7)	0 (0)	1	4
	Redis	mod_redisbloom		389	23	1	1	42% (8/19)	18 (13)	6 (4)	0 (0)	0	13
	Redis	mod_redisearch		381	21	1	1	54% (18/33)	15 (14)	14 (11)	0 (0)	0	12
	rsync	libpopt		167	8	1	1	90% (9/10)	4 (3)	2 (0)	0 (0)	0	5
	squid	libxml2		226	12	1	1	70% (7/10)	9 (5)	3 (3)	4 (1)	0	4
	su	libaudit		0	0	0	1	66% (2/3)	0 (0)	0 (0)	0 (0)	0	0
	Wiresbark	libpcap		162	8	2	1	50% (20/40)	8 (3)	5 (5)	0 (0)	0	4
	WITCHIAIK	libzlib		42	1	1	1	85% (6/7)	0 (0)	0 (0)	0 (0)	0	1
	Total:			5508	379	47	38	N/A	246 (192)	124 (105)	12 (5)	24	195
	cURL	libssl	[5]	198	27	1	1	25% (14/56)	18 (10)	5 (4)	1 (1)	0	17
	GPA	libgpgme		174	9	1	1	4% (3/72)	7 (2)	0 (0)	0 (0)	0	6
2	GPG	libgcrypt	[5]	4221	105	1	1	15% (15/95)	64 (60)	4 (0)	0 (0)	77	20
po.	Memcached	internal_hashtable	[45]	4037	16	1	1	50% (6/12)	10 (5)	2 (0)	0 (0)	1	6
afe	Mature	internal_libssl-keys	[45], [60], [15], [34]	599	46	1	1	50% (2/4)	32 (1)	28 (0)	0 (0)	0	22
S	nginx	libssl	[5], [1], [22], [51]	346	39	2	1	11% (11/96)	16 (13)	19 (13)	2 (1)	0	26
	anda	internal_auth-api		191	5	1	1	100% (5/5)	5 (4)	0 (0)	0 (0)	0	4
	sudo	libapparmor		97	3	1	1	100% (2/2)	2 (2)	2 (0)	0 (0)	0	2
	Total:			9863	250	9	8	N/A	154 (97)	60 (17)	3 (2)	78	103

	A	Commentaria A DI	Deferre	Cr	ashes	TV: Alerson	API	Coverage	II II	mpact (of w	hich arbi	trary)	
IM	Application	Compartment API	Kelerences	Raw	Dedup.	victims	Callers	Coverage	Read	Write	Exec	Alloc	Null
	UTTDA	libmarkdown	[42]	192	13	3	1	100% (4/4)	10 (8)	7 (7)	0 (0)	1	4
	нига	mod_markdown		381	71	5	1	100% (1/1)	62 (52)	17 (14)	2 (1)	0	30
	aspell	libaspell		278	8	1	1	34% (48/141)	7 (7)	7 (7)	2 (1)	0	3
	bind9	libxml2 (write API)		0	0	0	1	86% (13/15)	0 (0)	0 (0)	0 (0)	0	0
	bzip2	libbz2	[67], [5]	16	5	1	1	62% (5/8)	5 (2)	1 (0)	0 (0)	0	0
	cURL	libnghttp2		61	7	2	1	50% (18/36)	3 (3)	5 (5)	0 (0)	1	3
	exif	libexif		400	7	1	1	10% (13/129)	3 (3)	0 (0)	0 (0)	0	5
		libavcodec		316	20	3	4	31% (19/60)	13 (12)	12 (12)	0 (0)	3	7
	FFmpeg	libavfilter		51	1	1	2	12% (2/16)	1 (1)	0 (0)	0 (0)	0	1
		libavformat		217	9	2	3	52% (10/19)	8 (7)	1 (1)	0 (0)	0	7
	file	libmagic		150	5	1	1	63% (7/11)	5 (2)	1 (1)	0 (0)	0	4
	ait	libcurl	[22]	13	4	2	1	90% (18/20)	2 (2)	2 (2)	0 (0)	1	1
	gn	libpcre		81	2	1	1	44% (8/18)	2 (2)	0 (0)	0 (0)	2	0
	Inkscape	libpng	[67]	66	3	1	1	46% (14/30)	2 (1)	2 (2)	0 (0)	0	1
roq	Inkscape	libpoppler	[16]	81	4	2	1	100% (9/9)	4 (3)	4 (4)	0 (0)	0	2
Ipu	libxml2-tests	libxml2 (write API)		0	0	0	1	100% (47/47)	0 (0)	0 (0)	0 (0)	0	0
Sa	lighttpd	mod_deflate		117	26	2	1	100% (6/6)	16 (11)	5 (0)	1 (1)	2	9
	Imaga	libghostscript	[5]	67	14	2	1	100% (11/11)	4 (2)	1 (1)	0 (0)	3	9
	Magiak	libpng	[67]	778	44	1	2	22% (17/77)	2 (2)	9 (9)	2 (0)	2	39
	Wagick	libtiff	[67]	197	14	2	1	30% (13/43)	3 (3)	6 (6)	0 (0)	0	13
	Nginx	libpcre		144	10	1	1	93% (14/15)	8 (7)	3 (3)	0 (0)	6	2
		mod_geoip	[52]	276	25	2	1	35% (5/14)	21 (17)	4 (1)	1 (1)	1	10
	Olarlar	libmarkdown	[42]	64	5	3	1	100% (4/4)	3 (1)	0 (0)	0 (0)	1	2
	Okular	libpoppler	[16]	195	9	1	1	6% (24/379)	8 (6)	7 (7)	0 (0)	1	4
	Padia	mod_redisbloom		389	23	1	1	42% (8/19)	18 (13)	6 (4)	0 (0)	0	13
	Reuis	mod_redisearch		381	21	1	1	54% (18/33)	15 (14)	14 (11)	0 (0)	0	12
	rsync	libpopt		167	8	1	1	90% (9/10)	4 (3)	2 (0)	0 (0)	0	5
	squid	libxml2		226	12	1	1	70% (7/10)	9 (5)	3 (3)	4 (1)	0	4
	su	libaudit		0	0	0	1	66% (2/3)	0 (0)	0 (0)	0 (0)	0	0
	Wirschark	libpcap		162	8	2	1	50% (20/40)	8 (3)	5 (5)	0 (0)	0	4
	WIICSHAIK	libzlib		42	1	1	1	85% (6/7)	0 (0)	0 (0)	0 (0)	0	1
	Total:	-	-	5508	379	47	38	N/A	246 (192)	124 (105)	12 (5)	24	195
	cURL	libssl	[5]	198	27	1	1	25% (14/56)	18 (10)	5 (4)	1 (1)	0	17
	GPA	libgpgme		174	9	1	1	4% (3/72)	7 (2)	0 (0)	0 (0)	0	6
8	GPG	libgcrypt	[5]	4221	105	1	1	15% (15/95)	64 (60)	4 (0)	0 (0)	77	20
l q	Memcached	internal_hashtable	[45]	4037	16	1	1	50% (6/12)	10 (5)	2 (0)	0 (0)	1	6
afe	Nginy	internal_libssl-keys	[45], [60], [15], [34]	599	46	1	1	50% (2/4)	32 (1)	28 (0)	0 (0)	0	22
S	inginx	libssl	[5], [1], [22], [51]	346	39	2	1	11% (11/96)	16 (13)	19 (13)	2 (1)	0	26
	ando	internal_auth-api		191	5	1	1	100% (5/5)	5 (4)	0 (0)	0 (0)	0	4
	sudo	libapparmor		97	3	1	1	100% (2/2)	2 (2)	2 (0)	0 (0)	0	2
	Total:	-		9863	250	9	8	N/A	154 (97)	60 (17)	3 (2)	78	103

	тм	Application	Compartment API	References	Cr	ashes	Victime	API	Coverage	Impact (of w		nich arbi	itrary)	
		Аррисацон	Compartment Al I	Kelerences	Raw	Dedup.	vicuins	Callers	Coverage	Read	Write	Exec	Alloc	Null
		UTTDA	libmarkdown	[42]	192	13	3	1	100% (4/4)	10 (8)	7 (7)	0 (0)	1	4
25 applications		нира	mod_markdown		381	71	5	1	100% (1/1)	62 (52)	17 (14)	2 (1)	0	30
25 applications —		aspell	libaspell		278	8	1	1	34% (48/141)	7 (7)	7 (7)	2 (1)	0	3
		bind9	libxml2 (write API)		0	0	0	1	86% (13/15)	0 (0)	0 (0)	0 (0)	0	0
		bzip2	libbz2	[67], [5]	16	5	1	1	62% (5/8)	5 (2)	1 (0)	0 (0)	0	0
26 ADIs in total		cURL	libnghttp2		61	7	2	1	50% (18/36)	3 (3)	5 (5)	0 (0)	1	3
SO APIS III LOLAI		exif	libexif		400	7	1	1	10% (13/129)	3 (3)	0 (0)	0 (0)	0	5
			libavcodec		316	20	3	4	31% (19/60)	13 (12)	12 (12)	0 (0)	3	7
		FFmpeg	libavfilter		51	1	1	2	12% (2/16)	1 (1)	0 (0)	0 (0)	0	1
			libavformat		217	9	2	3	52% (10/19)	8 (7)	1 (1)	0 (0)	0	7
		file	libmagic		150	5	1	1	63% (7/11)	5 (2)	1 (1)	0 (0)	0	4
		oit	libcurl	[22]	13	4	2	1	90% (18/20)	2 (2)	2 (2)	0 (0)	1	1
		gr	libpcre		81	2	1	1	44% (8/18)	2 (2)	0 (0)	0 (0)	2	0
	2	Inkscape	libpng	[67]	66	3	1	1	46% (14/30)	2 (1)	2 (2)	0 (0)	0	1
	pog	museupe	libpoppler	[16]	81	4	2	1	100% (9/9)	4 (3)	4 (4)	0 (0)	0	2
	pm	libxml2-tests	libxml2 (write API)		0	0	0	1	100% (47/47)	0 (0)	0 (0)	0 (0)	0	0
	Sc	lighttpd	mod_deflate		117	26	2	1	100% (6/6)	16 (11)	5 (0)	1 (1)	2	9
		Image	libghostscript	[5]	67	14	2	1	100% (11/11)	4 (2)	1 (1)	0 (0)	3	9
		Magick	libpng	[67]	778	44	1	2	22% (17/77)	2 (2)	9 (9)	2 (0)	2	39
			libtiff	[67]	197	14	2	1	30% (13/43)	3 (3)	6 (6)	0 (0)	0	13
		Nginx	libpcre		144	10	1	1	93% (14/15)	8 (7)	3 (3)	0 (0)	6	2
		8	mod_geoip	[52]	276	25	2	1	35% (5/14)	21 (17)	4 (1)	1 (1)	1	10
		Okular	libmarkdown	[42]	64	5	3	1	100% (4/4)	3 (1)	0 (0)	0 (0)	1	2
			libpoppler	[16]	195	9	1	1	6% (24/379)	8 (6)	7 (7)	0 (0)	1	4
		Redis	mod_redisbloom		389	23	1	1	42% (8/19)	18 (13)	6 (4)	0 (0)	0	13
			mod_redisearch		381	21	1	1	54% (18/33)	15 (14)	14 (11)	0 (0)	0	12
		rsync	libpopt		167	8	1	1	90% (9/10)	4 (3)	2 (0)	0 (0)	0	5
		squid	libxml2		226	12	1	1	70% (7/10)	9 (5)	3 (3)	4(1)	0	4
		su	libaudit		0	0	0	1	66% (2/3)	0 (0)	0(0)	0 (0)	0	0
		Wireshark	libpcap		162	8	2	1	50% (20/40)	8 (3)	5 (5)	0 (0)	0	4
			IIDZIID		42	1	1	1	85% (0/7)		0(0)	0 (0)	0	
		Total:			5508	379	47	38	N/A	246 (192)	124 (105)	12 (5)	24	195
		cURL	libssl	[5]	198	27	1	1	25% (14/56)	18 (10)	5 (4)	1 (1)	0	17
		GPA	libgpgme		174	9	1	1	4% (3/72)	7 (2)	0 (0)	0 (0)	0	6
	8	GPG	libgcrypt	[5]	4221	105	1	1	15% (15/95)	64 (60)	4 (0)	0 (0)	77	20
	oq	Memcached	internal_hashtable	[45]	4037	16	1	1	50% (6/12)	10 (5)	2 (0)	0 (0)	1	6
	afe	Nginy	internal_libssl-keys	[45], [60], [15], [34]	599	46	1	1	50% (2/4)	32 (1)	28 (0)	0 (0)	0	22
	~	right	libssl	[5], [1], [22], [51]	346	39	2	1	11% (11/96)	16 (13)	19 (13)	2 (1)	0	26
		sudo	internal_auth-api		191	5	1	1	100% (5/5)	5 (4)	0 (0)	0 (0)	0	4
		Juuo	libapparmor		97	3	1	1	100% (2/2)	2 (2)	2 (0)	0 (0)	0	2
		Total:			9863	250	9	8	N/A	154 (97)	60 (17)	3 (2)	78	103

	тм	Application	Compartment API	References	Cr	ashes	Victims	API	Coverage	I	mpact (of wl	which arbitrary)		
	1.11	Аррисацон	Compartment Al I	Keterences	Raw	Dedup.	vicunis	Callers	Coverage	Read	Write	Exec	Alloc	Null
		UTTDA	libmarkdown	[42]	192	13	3	1	100% (4/4)	10 (8)	7 (7)	0 (0)	1	4
25 applications		нира	mod_markdown		381	71	5	1	100% (1/1)	62 (52)	17 (14)	2 (1)	0	30
25 applications —		aspell	libaspell		278	8	1	1	34% (48/141)	7 (7)	7 (7)	2 (1)	0	3
		bind9	libxml2 (write API)		0	0	0	1	86% (13/15)	0 (0)	0 (0)	0 (0)	0	0
		bzip2	libbz2	[67], [5]	16	5	1	1	62% (5/8)	5 (2)	1 (0)	0 (0)	0	0
26 ADIs in total		cURL	libnghttp2		61	7	2	1	50% (18/36)	3 (3)	5 (5)	0 (0)	1	3
SO APIS III LOLAI		exif	libexif		400	7	1	1	10% (13/129)	3 (3)	0 (0)	0 (0)	0	5
			libavcodec		316	20	3	4	31% (19/60)	13 (12)	12 (12)	0 (0)	3	7
		FFmpeg	libavfilter		51	1	1	2	12% (2/16)	1 (1)	0 (0)	0 (0)	0	1
			libavformat		217	9	2	3	52% (10/19)	8 (7)	1 (1)	0 (0)	0	7
		file	libmagic		150	5	1	1	63% (7/11)	5 (2)	1 (1)	0 (0)	0	4
		oit	libcurl	[22]	13	4	2	1	90% (18/20)	2 (2)	2 (2)	0 (0)	1	1
		gn	libpcre		81	2	1	1	44% (8/18)	2 (2)	0 (0)	0 (0)	2	0
	2	Inkscape	libpng	[67]	66	3	1	1	46% (14/30)	2 (1)	2 (2)	0 (0)	0	1
	pog		libpoppler	[16]	81	4	2	1	100% (9/9)	4 (3)	4 (4)	0 (0)	0	2
	pu	libxml2-tests	libxml2 (write API)		0	0	0	1	100% (47/47)	0 (0)	0 (0)	0 (0)	0	0
	Se	lighttpd	mod_deflate		117	26	2	1	100% (6/6)	16 (11)	5 (0)	1 (1)	2	9
		Image	libghostscript	[5]	67	14	2	1	100% (11/11)	4 (2)	1 (1)	0 (0)	3	9
		Magick	libpng	[67]	778	44	1	2	22% (17/77)	2 (2)	9 (9)	2 (0)	2	39
			libtiff	[67]	197	14	2	1	30% (13/43)	3 (3)	6 (6)	0 (0)	0	13
		Nginx	libpcre	(40)	144	10	1	1	93% (14/15)	8 (7)	3 (3)	0 (0)	6	2
			mod_geoip	[52]	276	25	2	1	35% (5/14)	21 (17)	4(1)	1 (1)	1	10
		Okular	libmarkdown	[42]	64	5	3	1	100% (4/4)	3(1)	0 (0)	0 (0)	1	2
			libpoppler	[16]	195	9	1	1	6% (24/3/9)	8 (6)	7(1)	0 (0)	1	4
		Redis	mod_redisbloom		389	23	1	1	42% (8/19)	18 (13)	6 (4)	0 (0)	0	13
			mod_redisearch		381	21	1	1	54% (18/33)	15 (14)	14 (11)	0 (0)	0	12
		rsync	libropt		107	8	1	1	90% (9/10)	4 (5)	2 (0)	0(0)	0	5
		squid	libxmi2 liboudit		226	12	1	1	/0% (//10)	9 (5)	3 (3)	4(1)	0	4
		su	libraan		162	0	0	1	50% (2/3)		5 (5)	0(0)	0	
		Wireshark	libzlib		102	0	2	1	30% (20/40) 85% (6/7)	0(0)	3 (3)	0(0)	0	4
	<u> </u>	Tetel	10210		42 5509	270	1	1	85% (0/7)		0(0)	12 (5)	0	1
		Total:			5508	379	47	38	N/A	246 (192)	124 (105)	12 (5)	24	195
		CURL	libssl	[5]	198	27	1	1	25% (14/56)	18 (10)	5 (4)	1(1)	0	17
		GPA	libgpgme	[7]	1/4	9	1	1	4% (3/72)	7 (2)	0(0)	0 (0)	0	0
	xo	GPG	libgcrypt	[5]	4221	105	1	1	15% (15/95)	64 (60)	4 (0)	0(0)	11	20
	feb	Memcached	internal_hashtable	[45]	4037	10	1	1	50% (6/12)	10 (5)	2 (0)	0 (0)	1	0
	Sa	Nginx	libeel	[45], [00], [15], [34]	399	40	1	1	50% (2/4)	52(1)	28 (0)	$\frac{0}{2}(0)$	0	22
			internal outh oni	[5], [1], [22], [51]	340	59	2	1	100% (5/5)	10 (15)	19 (15)	2 (1)	0	20
		sudo	libapparmor		07	3	1	1	100% (3/3) 100% (2/2)	3 (4)	2 (0)		0	4
	<u> </u>		поарранног		21	3	1	1	100% (212)	2 (2)	2 (0)		0	4
		Total:			9863	250	9	8	N/A	154 (97)	60 (17)	3 (2)	78	103

57

	тм	Application	Compartment API	References	Cr	Crashes Vic		API Coverage		Impact (of v		vhich arbitrary)		
	1.01	Аррисацон	Compartment Al I	Kererences	Raw	Dedup.	vicuins	Callers	Coverage	Read	Write	Exec	Alloc	Null
		UTTDA	libmarkdown	[42]	192	13	3	1	100% (4/4)	10 (8)	7 (7)	0 (0)	1	4
25 applications		niiru	mod_markdown		381	71	5	1	100% (1/1)	62 (52)	17 (14)	2 (1)	0	30
	\rightarrow	aspell	libaspell		278	8	1	1	34% (48/141)	7 (7)	7 (7)	2 (1)	0	3
		bind9	libxml2 (write API)		0	0	0	1	86% (13/15)	0 (0)	0 (0)	0 (0)	0	0
		bzip2	libbz2	[67], [5]	16	5	1	1	62% (5/8)	5 (2)	1 (0)	0 (0)	0	0
26 A Dis in total		cURL	libnghttp2		61	7	2	1	50% (18/36)	3 (3)	5 (5)	0 (0)	1	3
SU APIS III LULAI		exif	libexif		400	7	1	1	10% (13/129)	3 (3)	0 (0)	0 (0)	0	5
			libavcodec		316	20	3	4	31% (19/60)	13 (12)	12 (12)	0 (0)	3	7
		FFmpeg	libavfilter		51	1	1	2	12% (2/16)	1 (1)	0 (0)	0 (0)	0	1
			libavformat		217	9	2	3	52% (10/19)	8 (7)	1 (1)	0 (0)	0	7
		file	libmagic		150	5	1	1	63% (7/11)	5 (2)	1 (1)	0 (0)	0	4
		oit	libcurl	[22]	13	4	2	1	90% (18/20)	2 (2)	2 (2)	0 (0)	1	1
16 of which taken		511	libpcre		81	2	1	1	44% (8/18)	2 (2)	0 (0)	0 (0)	2	0
16 OF WHICH Laken		Inkoone	libpng	[67]	66	3	1	1	46% (14/30)	2 (1)	2 (2)	0 (0)	0	1
from the literature	po.		libpoppler	[16]	81	4	2	1	100% (9/9)	4 (3)	4 (4)	0 (0)	0	2
nom the interature	pu	libxml2-tests	libxml2 (write API)		0	0	0	1	100% (47/47)	0 (0)	0 (0)	0 (0)	0	0
	S	lighttpd	mod_deflate	(117	26	2	1	100% (6/6)	16 (11)	5 (0)	1 (1)	2	9
		Image	libghostscript	[5]	67	14	2	1	100% (11/11)	4 (2)	1 (1)	0 (0)	3	9
		Magick	libpng	[6/]	778	44	1	2	22% (17/77)	2 (2)	9 (9)	2 (0)	2	39
			libtiff	[67]	197	14	2		30% (13/43)	3 (3)	6 (6)	0 (0)		13
		Nginx	libpcre	(50)	144	10	1	1	93% (14/15)	8 (7)	3 (3)		0	2
		<u> </u>	mod_geo1p	[52]	276	25	2	1	35% (5/14)	21 (17)	4(1)	1(1)		10
		Okular	libmarkdown	[42]	04	5	3	1	100% (4/4)	3(1)				2
			nopoppier	[10]	195	9	1	1	0% (24/3/9)	8 (0)	(1)			4
		Redis	mod_redisoron		281	25	1	1	42% (8/19) 54% (18/22)	18 (15)	0 (4)			13
			librort		381	21	1	1	54% (18/55)	15 (14)	14(11)			12
		aguid	libym12		226	0	1	1	90% (9/10)	4 (5)	$\frac{2}{3}$ (0)			3
		squiu	liboudit		220	12	1	1	66% (2/2)	9(3)	3 (3)	4(1)		4
		su	libroon		162	8	2	1	50% (20/40)	8 (3)	5 (5)			
		Wireshark	libzlib		42	0	1	1	85% (6/7)					1
	<u> </u>	Total:	HOLHO		5508	270	1	28	N/A	246 (102)	124 (105)	12 (5)	24	105
	<u> </u>	allPI	libert	[5]	108	27	47	1	25% (14/56)	19 (10)	5 (4)	12(3)	24	195
		GPA	libaname	[3]	196	0	1	1	$\frac{25\%}{14/50}$	$\frac{10(10)}{7(2)}$	3 (4)	1(1)		6
		GPG	libgerupt	[5]	4221	9	1	1	4% (3/72)	64 (60)			77	20
	xoc	Memcached	internal hashtable	[45]	4037	16	1	1	50% (6/12)	10 (5)	2 (0)		1	6
	fet	Wiemeacheu	internal libed-keye	[45] [60] [15] [34]	500	46	1	1	50% (2/4)	32 (1)	28 (0)	0 (0)	0	22
	Sa	Nginx	libss]	[5] [1] [22] [51]	346	39	2	1	11% (11/96)	16 (13)	19 (13)	$\frac{2}{2}$		26
			internal auth-ani	[0], [1], [22], [01]	191	5	1	1	100% (5/5)	5 (4)	0.00	0.00	10	4
		sudo	libapparmor		97	3	1	1	100% (2/2)	2 (2)	2 (0)	0.00	0	2
	<u> </u>	Total:	noupputtion		9863	250	0	8	N/A	154 (97)	60 (17)	3 (2)	78	103
		Total.			2005	250	,	0	19/14	154 (97)	00(17)	5 (2)	10	105

58

	тм	Application	Compartment API	References	Cr	ashes	Victime	API	Coverage	Iı	npact (of w	hich arb	itrary)	
		Аррисацон	Compartment Arr	Kererences	Raw	Dedup.	vicuins	Callers	Coverage	Read	Write	Exec	Alloc	Null
		LITTO	libmarkdown	[42]	192	13	3	1	100% (4/4)	10 (8)	7 (7)	0 (0)	1	4
25 applications		нира	mod_markdown		381	71	5	1	100% (1/1)	62 (52)	17 (14)	2 (1)	0	30
zo applications		aspell	libaspell		278	8	1	1	34% (48/141)	7 (7)	7 (7)	2 (1)	0	3
		bind9	libxml2 (write API)		0	0	0	1	86% (13/15)	0 (0)	0 (0)	0 (0)	0	0
		bzip2	libbz2	[67], [5]	16	5	1	1	62% (5/8)	5 (2)	1 (0)	0 (0)	0	0
26 A Dis in total		cURL	libnghttp2		61	7	2	1	50% (18/36)	3 (3)	5 (5)	0 (0)	1	3
SO APIS III LULAI		exif	libexif		400	7	1	1	10% (13/129)	3 (3)	0 (0)	0 (0)	0	5
			libavcodec		316	20	3	4	31% (19/60)	13 (12)	12 (12)	0 (0)	3	7
		FFmpeg	libavfilter		51	1	1	2	12% (2/16)	1 (1)	0 (0)	0 (0)	0	1
			libavformat		217	9	2	3	52% (10/19)	8 (7)	1 (1)	0 (0)	0	7
		file	libmagic		150	5	1	1	63% (7/11)	5 (2)	1 (1)	0 (0)	0	4
		ait	libcurl	[22]	13	4	2	1	90% (18/20)	2 (2)	2 (2)	0 (0)	1	1
1C of which taken		gn	libpcre		81	2	1	1	44% (8/18)	2 (2)	0 (0)	0 (0)	2	0
16 of which taken		Inkaana	libpng	[67]	66	3	1	1	46% (14/30)	2 (1)	2 (2)	0 (0)	0	1
from the literature	coq	miscupe	libpoppler	[16]	81	4	2	1	100% (9/9)	4 (3)	4 (4)	0 (0)	0	2
nom the interature	pu	libxml2-tests	libxml2 (write API)		0	0	0	1	100% (47/47)	0 (0)	0 (0)	0 (0)	0	0
	Sa	lighttpd	mod_deflate		117	26	2	1	100% (6/6)	16 (11)	5 (0)	1 (1)	2	9
		Image	libghostscript	[5]	67	14	2	1	100% (11/11)	4 (2)	1 (1)	0 (0)	3	9
		Magick	libpng	[67]	778	44	1	2	22% (17/77)	2 (2)	9 (9)	2 (0)	2	39
		inigion	libtiff	[67]	197	14	2	1	30% (13/43)	3 (3)	6 (6)	0 (0)	0	13
		Nginx	libpcre		144	10	1	1	93% (14/15)	8 (7)	3 (3)	0 (0)	6	2
		- ·B····	mod_geoip	[52]	276	25	2	1	35% (5/14)	21 (17)	4 (1)	1 (1)	1	10
		Okular	libmarkdown	[42]	64	5	3	1	100% (4/4)	3 (1)	0 (0)	0 (0)	1	2
			libpoppler	[16]	195	9	1	1	6% (24/379)	8 (6)	7 (7)	0 (0)	1	4
		Redis	mod_redisbloom		389	23	1	1	42% (8/19)	18 (13)	6 (4)	0 (0)	0	13
			mod_redisearch		381	21	1	1	54% (18/33)	15 (14)	14 (11)	0 (0)	0	12
		rsync	libpopt		167	8	1	1	90% (9/10)	4 (3)	2 (0)	0 (0)	0	5
Found 620		squid	libxml2		226	12	1	1	70% (7/10)	9 (5)	3 (3)	4(1)	0	4
Found 629		su	libaudit		0	0	0	1	66% (2/3)	0 (0)	0 (0)	0 (0)	0	0
unique CIVs 🥆		Wireshark	Hapcan		162	8	2	1	50% (20/40)	8 (3)	5 (5)	0 (0)	0	4
			lidzlid		42	1	1	1	85% (6/7)	0 (0)	0 (0)	0 (0)	0	
		Total:			5508	379	47	38	N/A	246 (192)	124 (105)	12 (5)	24	195
		cURL	libssl	[5]	198	27	1	1	25% (14/56)	18 (10)	5 (4)	1 (1)	0	17
		GPA	libgpgme		174	9	1	1	4% (3/72)	7 (2)	0 (0)	0 (0)	0	6
	×	GPG	libgerypt	[5]	4221	105	1	1	15% (15/95)	64 (60)	4 (0)	0 (0)	77	20
	oqa	Memcached	internal_hashtable	[45]	4037	16	1	1	50% (6/12)	10 (5)	2 (0)	0 (0)	1	6
	Safe	Noiny	internal_libssl-keys	[45], [60], [15], [34]	599	46	1	1	50% (2/4)	32 (1)	28 (0)	0 (0)	0	22
	_	rightx	libssl	[5], [1], [22], [51]	346	39	2	1	11% (11/96)	16 (13)	19 (13)	2 (1)	0	26
		sudo	internal_auth-api		191	5	1	1	100% (5/5)	5 (4)	0 (0)	0 (0)	0	4
		5440	libapparmor		97	3	1	1	100% (2/2)	2 (2)	2 (0)	0 (0)	0	2
		Total:			9863	250	9	8	N/A	154 (97)	60 (17)	3 (2)	78	103

5 security impact types

		A	Contract ADI	D.C	Cr	ashes	N /2 /2	API	Coverage	I	mpact (of wl	nich arbi	trary)	
	TM	Application	Compartment API	References	Raw	Dedup.	Victims	Callers	Coverage	Read	Write	Exec	Alloc	Null
		LITTO	libmarkdown	[42]	192	13	3	1	100% (4/4)	10 (8)	7 (7)	0 (0)	1	4
25 applications		нигра	mod_markdown		381	71	5	1	100% (1/1)	62 (52)	17 (14)	2 (1)	0	30
25 applications	\rightarrow	aspell	libaspell		278	8	1	1	34% (48/141)	7 (7)	7 (7)	2 (1)	0	3
		bind9	libxml2 (write API)		0	0	0	1	86% (13/15)	0 (0)	0 (0)	0 (0)	0	0
		bzip2	libbz2	[67], [5]	16	5	1	1	62% (5/8)	5 (2)	1 (0)	0 (0)	0	0
26 ABIs in total		cURL	libnghttp2		61	7	2	1	50% (18/36)	3 (3)	5 (5)	0 (0)	1	3
SU AFIS III LULAI		exif	libexif		400	7	1	1	10% (13/129)	3 (3)	0 (0)	0 (0)	0	5
			libavcodec		316	20	3	4	31% (19/60)	13 (12)	12 (12)	0 (0)	3	7
		FFmpeg	libavfilter		51	1	1	2	12% (2/16)	1 (1)	0 (0)	0 (0)	0	1
			libavformat		217	9	2	3	52% (10/19)	8 (7)	1 (1)	0 (0)	0	7
		file	libmagic		150	5	1	1	63% (7/11)	5 (2)	1 (1)	0 (0)	0	4
		git	libcurl	[22]	13	4	2	1	90% (18/20)	2 (2)	2 (2)	0 (0)	1	1
16 of which takon		5	libpcre		81	2	1	1	44% (8/18)	2 (2)	0 (0)	0 (0)	2	0
		Inkseape	libpng	[67]	66	3	1	1	46% (14/30)	2(1)	2 (2)	0 (0)	0	1
from the literature	lpo	1	libpoppler	[16]	81	4	2	1	100% (9/9)	4 (3)	4 (4)	0 (0)	0	2
for the interature	ma	libxml2-tests	libxml2 (write API)		0	0	0	1	100% (47/47)	0 (0)	0 (0)	0 (0)	0	0
	S	lighttpd	mod_deflate	[5]	117	26	2	1	100% (6/6)	16 (11)	5 (0)	1 (1)	2	9
		Image	libghostscript	[5]	0/	14	2	1	100% (11/11)	4 (2)	1(1)	0 (0)	3	9
		Magick	hopng	[07]	1/8	44	1	2	22% (1////)	2 (2)	9 (9)	2 (0)	2	39
			libum	[0/]	197	14	2	1	$\frac{30\% (13/43)}{020\% (14/15)}$	3 (3)	0 (0)	0 (0)	0	15
		Nginx	mod gooin	[52]	276	25	2	1	95% (14/15) 25% (5/14)	8 (7) 21 (17)	3 (3) 4 (1)	$\frac{0}{1}$	0	10
		-	libmarkdown	[52]	270	25	2	1	$\frac{55\%}{100\%}$ (5/14)	$\frac{21(17)}{2(1)}$	4(1)	$\frac{1}{0}$ (1)	1	10
		Okular	libnonnlor	[42]	105	5	3	1	60% (24/270)	3 (1) 8 (6)		0 (0)	1	
			mod redisbloom	[10]	380	23	1	1	<u>12% (8/19)</u>	8 (0) 18 (13)	$\frac{7(1)}{6(4)}$	$\frac{0}{0}$	1	13
		Redis	mod_redisearch		381	23	1	1	54% (18/33)	15(13)	14(11)	$\frac{0}{0}$	0	12
		rsync	libnont		167	8	1	1	90% (9/10)	4(3)	$\frac{14(11)}{2(0)}$	0 (0)	0	5
		squid	libyml2		226	12	1	1	70% (7/10)	9 (5)	$\frac{2}{3}(3)$	$\frac{0}{4}(1)$	0	
Found 629		squid	libaudit		0	0	0	1	66% (2/3)	0(0)		$\frac{4(1)}{0(0)}$	0	0
		54	libncan		162	8	2	1	50% (20/40)	8 (3)	5 (5)	0 (0)	0	4
unique CIVs 📉		Wireshark	libzlib		42	1	1	1	85% (6/7)	0 (0)	0 (0)	0 (0)	0	1
		Total:			5508	379	47	38	N/A	246 (192)	124 (105)	12 (5)	24	195
		cURE	libssl	[5]	198	27	1	1	25% (14/56)	18 (10)	5 (4)	1(1)	0	17
		GPA	libgpgme		174	9	1	1	4% (3/72)	7 (2)	0 (0)	0 (0)	0	6
	2	GPG	libgerypt	[5]	4221	105	1	1	15% (15/95)	64 (60)	4 (0)	0 (0)	77	20
	po.	Memcached	internal_hashtable	[45]	4037	16	1	1	50% (6/12)	10 (5)	2 (0)	0 (0)	1	6
	afe	Nginy	internal_libssl-keys	[45], [60], [15], [34]	599	46	1	1	50% (2/4)	32 (1)	28 (0)	0 (0)	0	22
	S	reginx	libssl	[5], [1], [22], [51]	346	39	2	1	11% (11/96)	16 (13)	19 (13)	2 (1)	0	26
		sudo	internal_auth-api		191	5	1	1	100% (5/5)	5 (4)	0 (0)	0 (0)	0	4
		suuo	libapparmor		97	3	1	1	100% (2/2)	2 (2)	2 (0)	0 (0)	0	2
		Total:			9863	250	9	8	N/A	154 (97)	60 (17)	3 (2)	78	103

"How many CIVs are there at legacy APIs? Are all APIs similarly affected?"

• Confirms: CIVs are widespread among unmodified APIs/code

- Confirms: CIVs are widespread among unmodified APIs/code
- But: there are **clear disparities among APIs**

- Confirms: CIVs are widespread among unmodified APIs/code
- But: there are clear disparities among APIs
 - CIV counts vary 0 105 CIVs for a single API

- Confirms: CIVs are widespread among unmodified APIs/code
- But: there are clear disparities among APIs
 - CIV counts vary 0 105 CIVs for a single API



"How many CIVs are there at legacy APIs? Are all APIs similarly affected?"

- Confirms: CIVs are widespread among unmodified APIs/code
- But: there are clear disparities among APIs
 - CIV counts vary 0 105 CIVs for a single API



For each scenario

"How many CIVs are there at legacy APIs? Are all APIs similarly affected?"

- Confirms: CIVs are widespread among unmodified APIs/code
- But: there are clear disparities among APIs
 - CIV counts vary 0 105 CIVs for a single API



Number of vulnerable API endpoints (= has CIVs) versus non-vulnerable

- Confirms: CIVs are widespread among unmodified APIs/code
- But: there are clear disparities among APIs
 - CIV counts vary 0 105 CIVs for a single API



"How many CIVs are there at legacy APIs? Are all APIs similarly affected?"

- Confirms: CIVs are widespread among unmodified APIs/code
- But: there are clear disparities among APIs
 - CIV counts vary 0 105 CIVs for a single API



• There are large and (almost) totally CIV-free APIs

- Confirms: CIVs are widespread among unmodified APIs/code
- But: there are clear disparities among APIs
 - CIV counts vary 0 105 CIVs for a single API



- There are large and (almost) totally CIV-free APIs
- There are small and fully vulnerable APIs

"How many CIVs are there at legacy APIs? Are all APIs similarly affected?"

- Confirms: CIVs are widespread among unmodified APIs/code
- But: there are clear disparities among APIs
 - CIV counts vary 0 105 CIVs for a single API



- There are large and (almost) totally CIV-free APIs
- There are small and fully vulnerable APIs

No correlation between API size and CIV count!

Study results: CIV Patterns

- "Are all APIs similarly affected? How hard are they to fix?"
- We observe recurring API design patterns that lead to CIVs
Study results: CIV Patterns

"Are all APIs similarly affected? How hard are they to fix?"

- We observe recurring API design patterns that lead to CIVs
- These reinforce the idea that the presence of CIVs is influenced by structural properties rather than API size or quantity of shared data

Study results: CIV Patterns

"Are all APIs similarly affected? How hard are they to fix?"

- We observe recurring API design patterns that lead to CIVs
- These reinforce the idea that the presence of CIVs is influenced by structural properties rather than API size or quantity of shared data
- Highlight one of these patterns here
- Many more in the paper!

• Module APIs are the most CIV-vulnerable APIs in the study (HTTPd, Nginx, Redis, lighttpd, etc.)



- Module APIs are the most CIV-vulnerable APIs in the study (HTTPd, Nginx, Redis, lighttpd, etc.)
- More CIVs and worse CIVs on average



- Module APIs are the most CIV-vulnerable APIs in the study (HTTPd, Nginx, Redis, lighttpd, etc.)
- More CIVs and worse CIVs on average
- Unlike library APIs, module APIs must be very generic and yield high performance
 - Consequence: the application's core internal state is exposed to the module

od_geoip mod_proxy Apache Module API
Apache Core
APACHE

HTTP SERVER PROJECT

- Module APIs are the most CIV-vulnerable APIs in the study (HTTPd, Nginx, Redis, lighttpd, etc.)
- More CIVs and worse CIVs on average
- Unlike library APIs, module APIs must be very generic and yield high performance
 - Consequence: the application's **core internal state** is **exposed to the module**

request structure crossing the interface



- Module APIs are the most CIV-vulnerable APIs in the study (HTTPd, Nginx, Redis, lighttpd, etc.)
- More CIVs and worse CIVs on average
- Unlike library APIs, module APIs must be **very** generic and yield high performance
 - Consequence: the application's core internal state is exposed to the module



>75 fields, 60% of which pointers

request structure

- Module APIs are the most CIV-vulnerable APIs in the study (HTTPd, Nginx, Redis, lighttpd, etc.)
- More CIVs and worse CIVs on average
- Unlike library APIs, module APIs must be **very** generic and yield high performance
 - Consequence: the application's core internal state is exposed to the module

Somewhat counter-intuitive: modularity does not imply low compartmentalization complexity



>75 fields, 60% of which pointers

request structure

• "How bad are they?"

- "How bad are they?"
- Confirms: CIVs are high-impact
 - >75% of scenarios present at least a write vulnerability

- "How bad are they?"
- Confirms: CIVs are high-impact
 - >75% of scenarios present at least a write vulnerability
 - ~70% of write and read, and ~50% of execute vulnerabilities are arbitrary

- "How bad are they?"
- Confirms: CIVs are high-impact
 - >75% of scenarios present at least a write vulnerability
 - ~70% of write and read, and ~50% of execute vulnerabilities are arbitrary
 - Only 8/39 scenarios have execution impact CIVs
 - but read and/or write primitives can likely be combined to achieve code execution

- "How bad are they?"
- Confirms: CIVs are high-impact
 - >75% of scenarios present at least a write vulnerability
 - ~70% of write and read, and ~50% of execute vulnerabilities are arbitrary
 - Only 8/39 scenarios have execution impact CIVs
 - but read and/or write primitives can likely be combined to achieve code execution
- Here: illustrate security impact with a concrete scenario
 - Key extraction in OpenSSL safebox
- More in the paper

- Assume we isolate OpenSSL to protect SSL keys
 - for example, from a compromised Nginx
- The key compartment interface is the **OpenSSL public API**



- Assume we isolate OpenSSL to protect SSL keys
 - for example, from a compromised Nginx
- The key compartment interface is the **OpenSSL public API**
- We find several CIVs that enable for read, write, and execution impact OpenSSL option setting primitive – part of the public API OpenSSL option setting API Leads to arbitrary R/W ulong SSL_CTX_set_options(SSL_CTX *ctx, ulong op) { return ctx->options |= op;

- Assume we isolate OpenSSL to protect SSL keys
 - for example, from a compromised Nginx
- The key compartment interface is the **OpenSSL public API**
- We find several CIVs that enable for read, write, and execution impact
 OpenSSL option setting primitive part of the public API
 Dereference arbitrary pointer, set it, and return it: arbitrary read and arbitrary write oracle → Key extracted

- Assume we isolate OpenSSL to protect SSL keys
 - for example, from a compromised Nginx
- The key compartment interface is the **OpenSSL public API**
- We find several CIVs that enable for read, write, and execution impact
- Because of CIVs, isolating at the OpenSSL boundary is weak



OpenSSL

OpenSSL Library AP

- Same scenario, but let's plug at a different interface
- This time, the **OpenSSL internal key API**
 - This is a very popular use-case / application in the literature



- Same scenario, but let's plug at a different interface
- This time, the OpenSSL internal key API
 - This is a very popular use-case / application in the literature
- Here too, we find key-extraction CIVs



- Same scenario, but let's plug at a different interface
- This time, the OpenSSL internal key API
 - This is a very popular use-case / application in the literature
- Here too, we find key-extraction CIVs



93

- Same scenario, but let's plug at a different interface
- This time, the **OpenSSL internal key API**
 - This is a very popular use-case / application in the literature
- Here too, we find key-extraction CIVs

Point *in to the key, encrypt with known *key, then decrypt

• Fixing them requires to make the component stateful

Prototype of one of the key API endpoints

 \rightarrow Key extracted



94

- Same scenario, but let's plug at a different interface
- This time, the **OpenSSL internal key API**
 - This is a very popular use-case / application in the literature
- Here too, we find key-extraction CIVs

Point *in to the key, encrypt with known *key, then decrypt

• Fixing them requires to make the component stateful

Because of CIVs, all OpenSSL isolation use-cases from the literature are pointless & fixes are non-trivial



Prototype of one of the key API endpoints

Ways forward to tackle CIVs:

Ways forward to tackle CIVs:

- 1. Progress towards more systematic, automatic CIV defenses
 - we highlight limitations of existing defenses in the paper

Ways forward to tackle CIVs:

- 1. Progress towards more systematic, automatic CIV defenses
 - we highlight limitations of existing defenses in the paper
- 2. (Re-)Design interfaces to be CIV-resilient by design
 - we provide a set of guidelines to achieve this

Ways forward to tackle CIVs:

1. Progress towards more systematic, automatic CIV defenses

- we highlight limitations of existing defenses in the paper
- 2. (Re-)Design interfaces to be **CIV-resilient by design**
 - we provide a set of guidelines to achieve this

Fundamentally hard: need to understand API semantics

Both approaches are complimentary

 \rightarrow (Re-)Design interfaces to be CIV-resilient by design

 \rightarrow (Re-)Design interfaces to be CIV-resilient by design

- Not every interface is a good compartmentalization boundary
 - choose your interface wisely

 \rightarrow (Re-)Design interfaces to be CIV-resilient by design

- Not every interface is a good compartmentalization boundary
 - choose your interface wisely
- Simplify as much as possible interface-crossing objects
 - no system resource handles, not complex structs, synchronization primitives
 - if not possible, you should probably plug at a different interface

 \rightarrow (Re-)Design interfaces to be CIV-resilient by design

- Not every interface is a good compartmentalization boundary
 - choose your interface wisely
- Simplify as much as possible interface-crossing objects
 - no system resource handles, not complex structs, synchronization primitives
 - if not possible, you should probably **plug at a different interface**
- Enforce API semantics (ordering, concurrency support)

• CIVs should be at the center of every compartmentalization approach

- CIVs should be at the center of every compartmentalization approach
- API design patterns influence CIV prevalence and severity
 - it's not so much about the size of the API
 - ... it's about the complexity of API-crossing objects

- CIVs should be at the center of every compartmentalization approach
- API design patterns influence CIV prevalence and severity
 - it's not so much about the size of the API
 - ... it's about the complexity of API-crossing objects
- Addressing CIVs requires more than a few checks
 - ... strong solutions often require refactoring the API
 - automatic compartmentalization is harder that setting & enforcing bounds

- CIVs should be at the center of every compartmentalization approach
- API design patterns influence CIV prevalence and severity
 - it's not so much about the size of the API
 - ... it's about the complexity of API-crossing objects
- Addressing CIVs requires more than a few checks
 - ... strong solutions often require refactoring the API
 - automatic compartmentalization is harder that setting & enforcing bounds
- We need more research to address the problem of CIVs

- CIVs should be at the center of every compartmentalization approach
- API design patterns influence CIV prevalence and severity
 - it's not so much about the size of the API
 - ... it's about the complexity of API-crossing objects
- Addressing CIVs requires more than a few checks
 - ... strong solutions often require refactoring the API
 - automatic compartmentalization is harder that setting & enforcing bounds
- We need more research to address the problem of CIVs

NDSS'23 Paper: <u>https://arxiv.org/abs/2212.12904</u> Project website: <u>https://conffuzz.github.io</u> Code & Dataset under BSD-3 & CC-BY