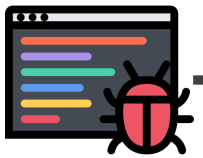
VIPER: Spotting Syscall-Guard Variables for Data-Only Attacks

Hengkai Ye Song Liu Zhechang Zhang Hong Hu

PennState

Current Exploit Method: Control-Flow Hijacking



Memory-Access Primitives

Arbitrary Read
Arbitrary Write

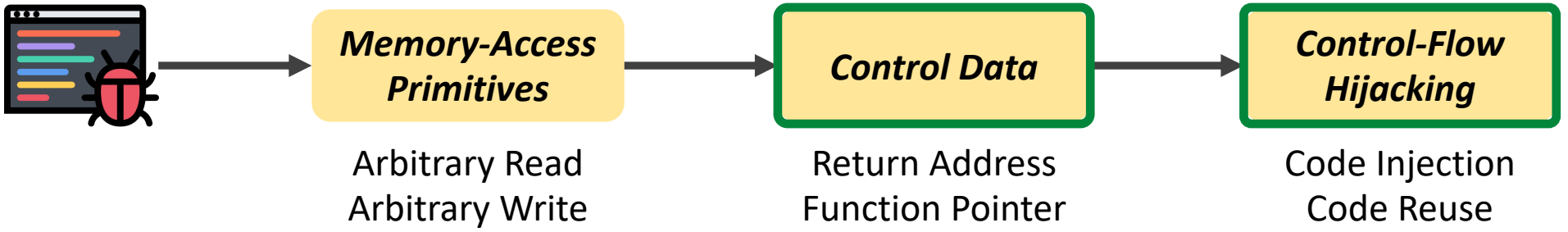
Control Data

Return Address
Function Pointer

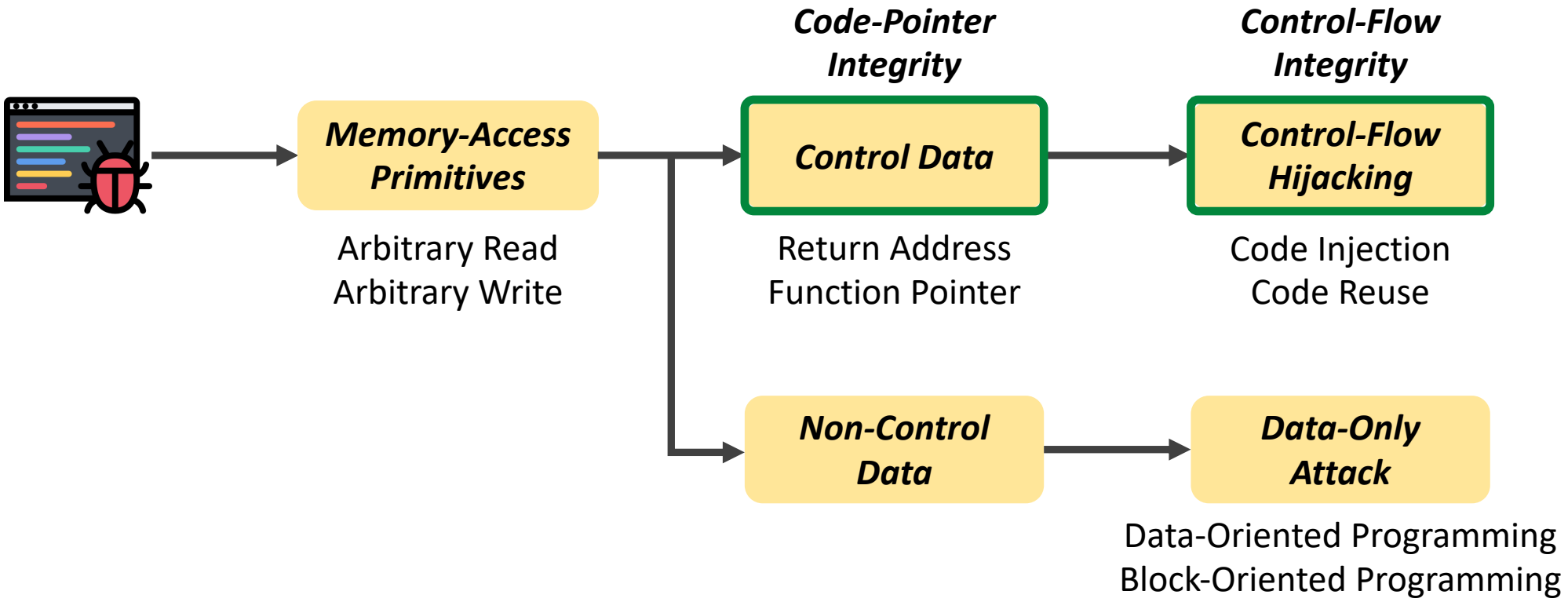
Control-Flow Hijacking

Code Injection
Code Reuse

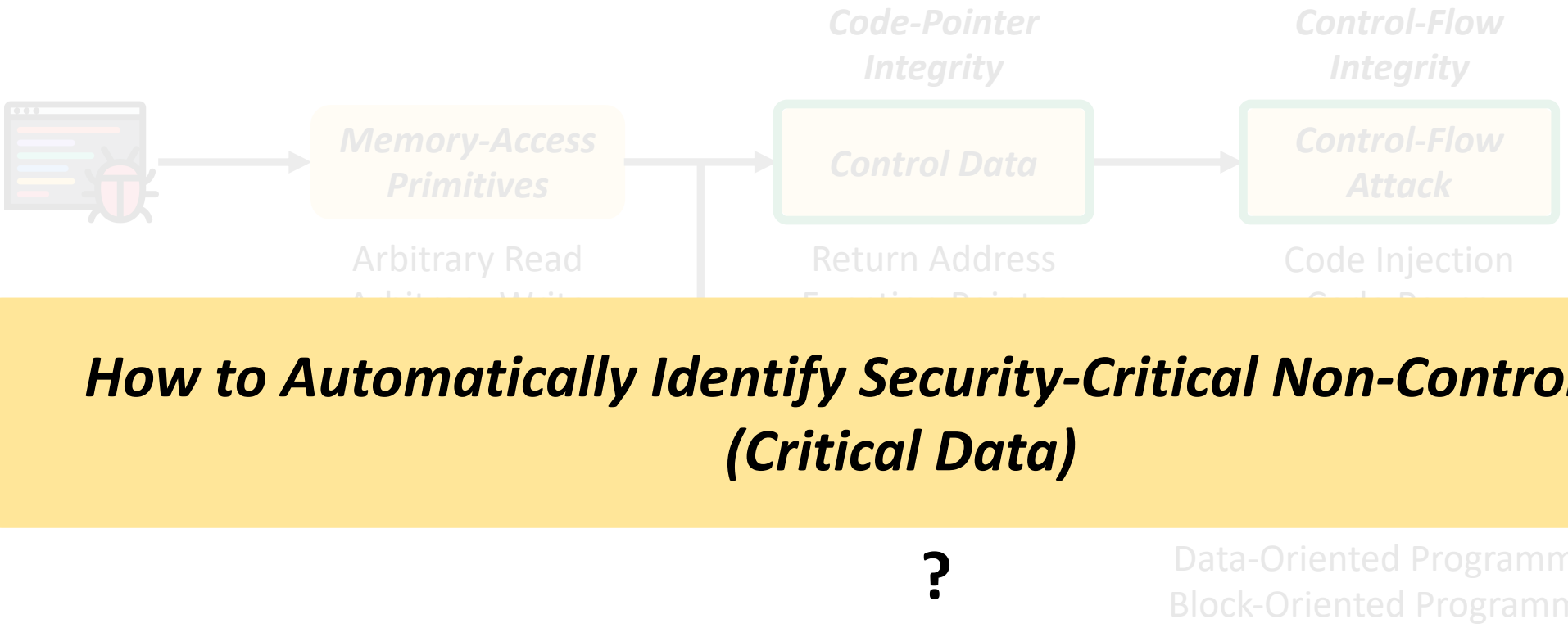
Current Exploit Method: Control-Flow Hijacking



Next Gen Exploit Method: Data-Only Attack



Next Gen Exploit Method: Data-Only Attack



Spotting Critical Data is Challenging

Critical data

- No common low-level properties (e.g., data type, memory location)
- Difficult to infer high-level semantics

Previous work

- Manual inspection: tedious human efforts, not scalable
- FlowStitch [Security'15]: rely on explicit sources/sinks
  - e.g., argument of ***setuid***
- KENALI [NDSS'16]: rely on error codes in Linux Kernel

Our Contribution

- Automatic identification of syscall-guard variables
  - Branch force
  - Corruptibility assessment
- A framework - *VIPER*
  - 34 unknown syscall-guard variables from 13 programs
  - 4 new data-only attacks on SQLite and V8
- https://github.com/psu-security-universe/viper

Motivating Example

```
1  void do_authentication(char *user, ...) {
2    int authenticated = 0;
3    ...
4    while (!authenticated) {
5     /* Get a packet from the client */
6     type = packet_read();
7     ...
8     if (auth_password(user, password))
9       authenticated = 1;
10    ...
11    if (authenticated) break;
12   }
13   /* Perform session preparation. */
14   do_authenticated(pw);      // open access
15 }
```

How to identify "authenticated"?

Chen, Shuo, et al. "Non-control-data attacks are realistic threats." USENIX security symposium. Vol. 5. 2005.

Motivating Example

```
1  void do_authentication(char *user, ...) {
2    int authenticated = 0;      // non-control data
3    ...
4    while (!authenticated) {
5     /* Get a packet from the client */
6     type = packet_read();      // bug -> write primitive
7     ...
8     if (auth_password(user, password))
9       authenticated = 1;
10    ...
11    if (authenticated) break;
12   }
13   /* Perform session preparation. */
14   do_authenticated(pw);       // open access
15 } +-> do_authenticated
16       +-> do_exec_pty
17           +-> do_child
18               +-> execve(shell, argv, env); // guarded syscall
```

How to identify "authenticated"?

Most data-only attacks rely on ***security-related syscalls***

Security-related syscalls are often guarded by security checks

Syscall-Guard Branch: security checks as conditional branches

Syscall-Guard Variable: variables in syscall-guard branches

VIPER: identify syscall-guard variables

Chen, Shuo, et al. "Non-control-data attacks are realistic threats." USENIX security symposium. Vol. 5. 2005.

Does Syscall-Guard Variable Matter?

Program	Critical Data	Security Impact
nginx	`clcf->root.data`	access any server file
	`ctx`	execute arbitrary program
openssh	`authenticated`	login w/ wrong password
	`original_uid`	obtain root-user privilege
sudo	`user_details.uid`	obtain root-user privilege
null httpd	`config.server_cgi_dir`	execute arbitrary program
	`config.server_htdocs_dir`	access any server file
ghttpd	`ptr`	execute arbitrary program
orzhttpd	`conn->basedir.path`	access any server file
wu-ftpd	`pw->pw_uid`	obtain root-user privilege
telnet	`loginprg`	execute arbitrary program
chromium	`m_universalAccess`	disable same-origin check
httpdx	`ftps.i["admin"].pass`	admin login w/o password
	`ftps.i["anon."].flags`	can detele file or directory
	`ftps.i["anon."].root`	access any file on the server
	`handlers[cgi].cmd`	execute arbitrary program
IE Browser	`safemode`	execute arbitrary code

11 syscall arguments

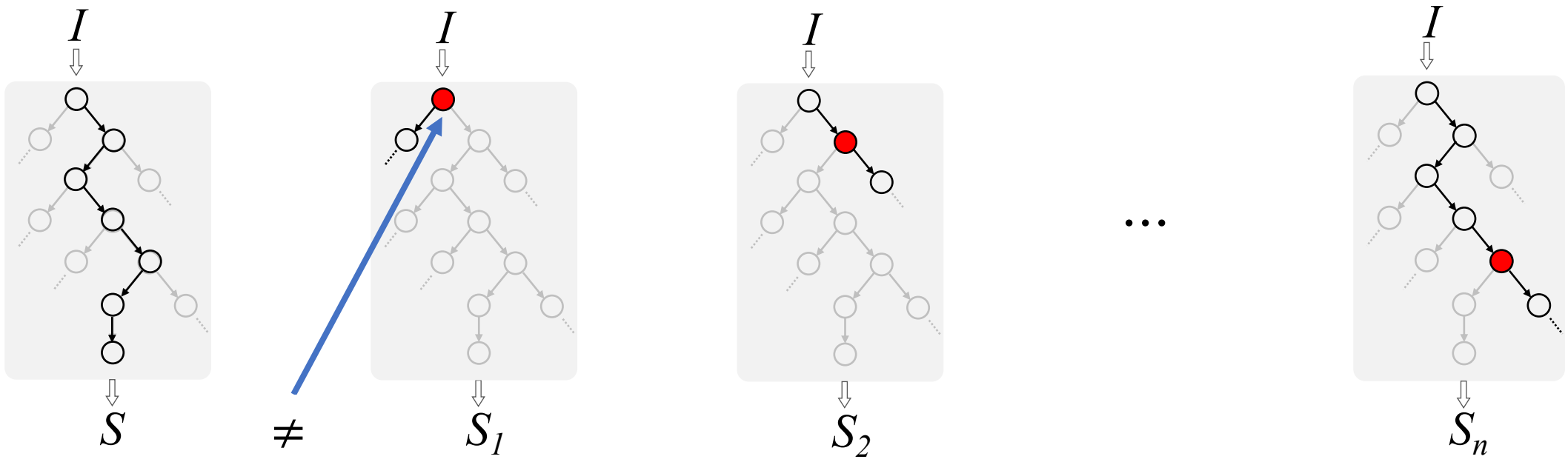
6 syscall-guard variables

Challenges

- Identify ***sole*** contribution of each variable
  - Symbolic execution can identify a complete path
    - Limitation: cannot tell which variables are more critical
- Efficient and scalable analysis
  - Static analysis
    - Limitations: indirect calls, inter-procedural analysis, etc

Branch Force: Identify Syscall-Guard Branches

- Flip every branch during execution
- Hook syscalls to find newly invoked ones
- If yes, the flipped is a syscall-guard branch

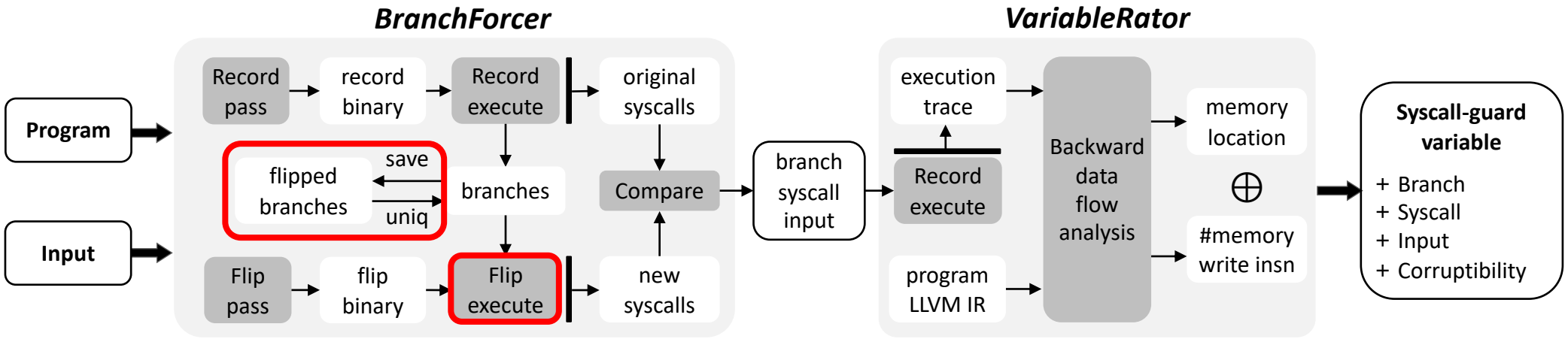


Corruptibility Assessment

- Backward Data-Flow Analysis
  - Generate data flow of syscall-guard variables
- Assessment (for each memory node in the data flow)
  - Metric 1: memory location
    - ***Global > Heap > Stack***
  - Metric 2: number of memory-write instructions
    - Assumption: every memory-write could be abused

More details can be found in the paper

Workflow of VIPER



- Unique Branch Flipping
- Forkserver

- Record execution trace on LLVM IR level
- Simulate execution based on recorded trace

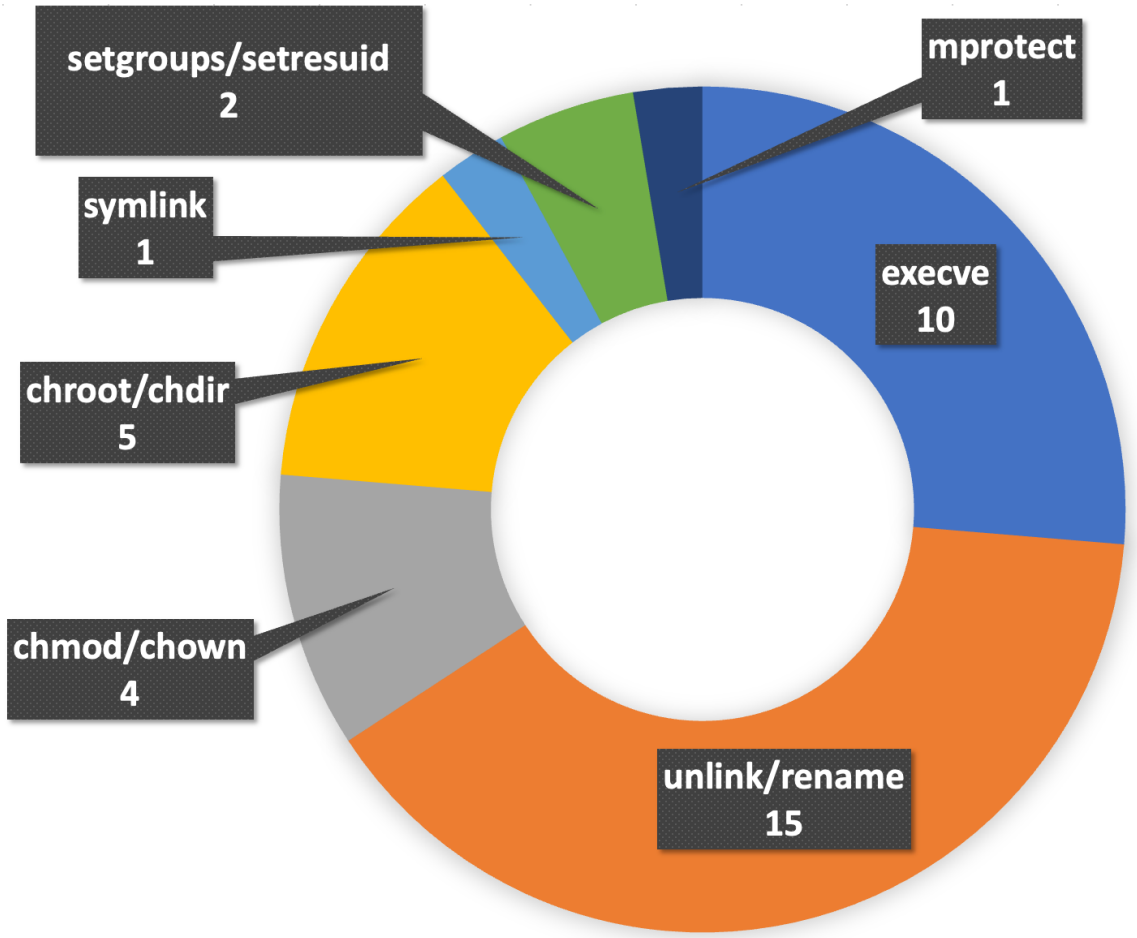
Evaluation (setting)

- 20 programs for evaluation
  - 9 programs with known data-only attacks (e.g., OpenSSH)
  - 7 programs from FuzzBench (e.g., SQLite)
  - 4 other well-tested programs (e.g., V8)
- Corpus
  - Testcases in source code repository
  - Online corpus (e.g., FuzzBench Dataset)
  - Fuzz with AFL++

Evaluation (identified syscall-guard variables)

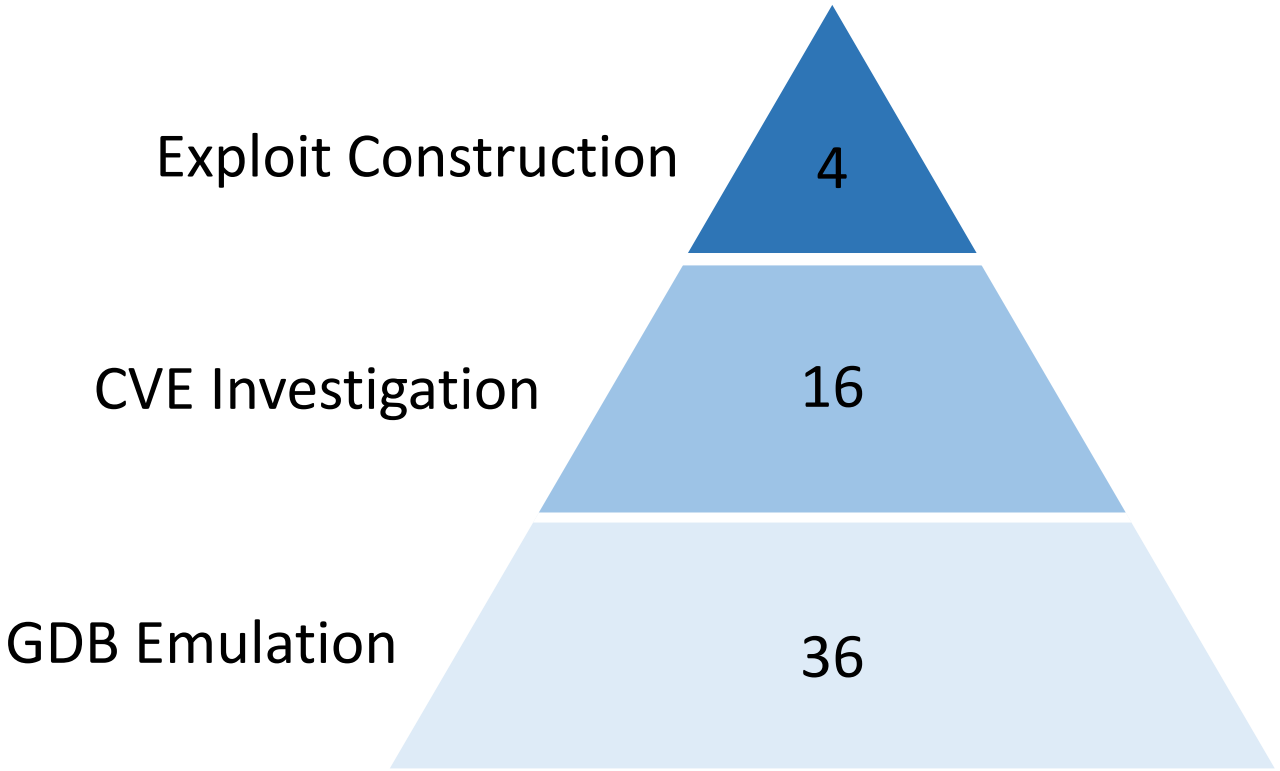
Program	Guard Variable	Branch Location	Syscall	Malicious Goal
sqlite	mode	shell.c:5002	symlink	create symlinks to any file
		shell.c:5038	chmod	change any file to any mode
	p->doXdgOpen	shell.c:20270	execve	execute arbitrary program
	p->zTempFile	shell.c:20560	unlink	delete any file
	isDelete	sqlite3.c:42939	unlink	delete any file
	zPath	sqlite3.c:43094	unlink	delete any file
	exists	sqlite3.c:60294	unlink	delete any file
	isWal	sqlite3.c:58492	unlink	delete any file
curl	tempstore	cookie.c:1732	rename	overwrite any file
	tempstore	hsts.c:386	rename	overwrite any file
	tempstore	altsvc.c:359	rename	overwrite any file
harfbuzz	blob->mode	hb-blob.cc:453	mprotect	make RO memory writable
nginx	sa_family	$_connection.c:631	chmod	change file mode
	ngx_terminate	$_process_cycle.c:305	unlink	delete any file
	ngx_quit	$_process_cycle.c:305	unlink	delete any file
	ft.st_uid	($: ngx) $_file.c:631	chown	change owner of any file
	ft.st_mode	$_file.c:640	chmod	change file mode
openssh	result*	auth-passwd.c:128	execve	login without password
	received_sigterm	sshd.c:1163	unlink	delete any file
	received_sighup	sshd.c:1177	execve	execute arbitrary program
sudo	details->chroot	exec.c:173	chroot	change root path
	info	sudo.c:697	chdir	change directory path
null httpd	in_RequestURI	main.c:39	execve	enable CGI to run programs
ghttpd	filename*	protocol.c:127	execve	enable CGI to run programs
wu-ftpd	RootDirectory	ftpd.c:1029	chroot	change root path of current user
	anonymous	ftpd.c:2527	setgroups	obtain root privilege
		ftpd.c:2893	chroot	change root path of anonymous
	guest	ftpd.c:2893	chroot	change root path of guest
	rval	ftpd.c:2708	setresuid	login without password
jhead	RegenThumbnail	jhead.c:978	execve	execute arbitrary program
	EditComment	jhead.c:1003	execve	edit any file using vi
	CommentInsertfileName	jhead.c:1003	execve	edit any file using vi
	CommentInsertLiteral	jhead.c:1003	execve	edit any file using vi
jasper	fileobj->flags	jas_stream.c:1392	unlink	delete any file
pdfalto	first	XRef.cc:240	unlink	delete files in specific folders
	offsets[0]	XRef.cc:240	unlink	delete files in specific folders
gzip	fd	gzip.c:2111	unlink	delete any file
v8	enable_os_system	d8-posix.cc:762	execve	execute any program

36 syscall-guard variables from 14 programs



Evaluation (exploitability investigation)

Program	Guard Variable	Branch Location	Rate (S, H, G)	CK	CVE	Type	Cap
sqlite	`mode`	shell.c:5002	(55, 0, 0)	◔			
		shell.c:5038	(75, 0, 0)	◔			
	`p->doXdgOpen`	shell.c:20270	(181770, 0, 0)	●	2017-6983	TC	AW
	`p->zTempFile`	shell.c:20560	(86907, 0, 0)	●	2017-6983	TC	AW
	`isDelete`	sqlite3.c:42939	(8353, 29276, 0)	●	2017-6983	TC	AW
	`zPath`	sqlite3.c:43094	(57, 15036, 0)	◔			
	`exists`	sqlite3.c:60294	(58, 15036, 0)	◔			
	`isWal`	sqlite3.c:58492	(61, 15046, 0)	◔			
curl	`tempstore`	cookie.c:1732	(15, 0, 0)	◕	2019-3822	H/SBoF	AW
	`tempstore`	hsts.c:386	(15, 0, 0)	◕	2019-3822	H/SBoF	AW
	`tempstore`	altsvc.c:359	(15, 0, 0)	◕	2019-3822	H/SBoF	AW
harfbuzz	`blob->mode`	hb-blob.cc:453	(31, 352, 0)	◕	2015-8947	HBoF	AW
nginx	`sa_family`	$_connection.c:631	(0, 84831, 0)	◔			
	`ngx_terminate`	$_process_cycle.c:305	(0, 0, 208640)	◕	2013-2028	SBoF	AW
	`ngx_quit`	$_process_cycle.c:305	(0, 0, 208640)	◕	2013-2028	SBoF	AW
	`ft.st_uid`	($: ngx) $_file.c:631	(350832, 0, 0)	◔			
	`ft.st_mode`	$_file.c:640	(175218, 0, 0)	◔			
openssh	`result`*	auth-passwd.c:128	(5, 48153980, 0)	◔			
	`received_sigterm`	sshd.c:1163	(0, 0, 1463147)	◔			
	`received_sighup`	sshd.c:1177	(0, 0, 1470603)	◔			
sudo	`details->chroot`	exec.c:173	(0, 0, 2039)	◕	2012-0809	FS	AW
	`info`	sudo.c:697	(1702, 253382, 1982)	◕	2012-0809	FS	AW
null httpd	`in_RequestURI`	main.c:39	(0, 525, 0)	◕	2002-1496	HBoF	AW
ghttpd	`filename`*	protocol.c:127	(9, 0, 5912)	◕	2002-1904	SBoF	AW
wu-ftpd	`RootDirectory`	ftpd.c:1029	(0, 0, 7322)	◔			
	`anonymous`	ftpd.c:2527	(0, 0, 7432)	◔			
		ftpd.c:2893	(0, 0, 8341)	◔			
	`guest`	ftpd.c:2893	(0, 0, 37715)	◔			
	`rval`	ftpd.c:2708	(8, 0, 0)	◔			
jhead	`RegenThumbnail`	jhead.c:978	(0, 0, 2856)	◕	2016-3822	IO	AW
	`EditComment`	jhead.c:1003	(0, 0, 2856)	◕	2016-3822	IO	AW
	`CommentInsertfileName`	jhead.c:1003	(0, 0, 2856)	◕	2016-3822	IO	AW
	`CommentInsertLiteral`	jhead.c:1003	(0, 0, 2856)	◕	2016-3822	IO	AW
jasper	`fileobj->flags`	jas_stream.c:1392	(0, 219062, 0)	◕	2020-27828	HBoF	AW
pdfalto	`first`	XRef.cc:240	(1952, 214, 0)	◔			
	`offsets[0]`	XRef.cc:240	(92, 117, 0)	◔			
gzip	`fd`	gzip.c:2111	(0, 0, 11886)	◕	2010-0001	IO	AW
v8	`enable_os_system`	d8-posix.cc:762	(0, 0, 93512607)	●	2021-30632	TC	AW



Evaluation (time costs)

Program	Version	kLoC	Time Cost					Stitch
			Record	Flip	Rate	Total	Total/A	
sqlite	3.40.1	273	288"	112"	378"	778"	87"	
curl	97f7f66	160	23"	32"	689"	744"	248"	
harfbuzz	1.3.2	41	17"	8"	8"	33"	33"	
systemd	v252	543	69"	40"	-	>109"	>109"	
mbedtls	10ada35	128	2"	6"	-	>8"	>8"	
openssl	3.0.7	483	13"	61"	-	>74"	>74"	
freetype2	cd02d35	119	18"	26"	-	>44"	>44"	
nginx	1.20.2	141	238"	22"	329"	589"	118"	121"
openssh	36b00d3	119	1"	4722"	10624"	15347"	5116"	1110"
sudo	1.9.9	110	16"	16"	260"	292"	18"	393"
null httpd	0.5.1	2	1"	10"	31"	42"	42"	358"
ghttpd	1.4.4	1	1"	36"	72"	109"	55"	48"
orzhttpd	0.0.6	3	1"	32"	-	>33"	>33"	93"
wu-ftpd	2.6.2	18	1"	533"	189"	723"	91"	200"
telnet	3f35287	11	1"	144"	-	>145"	>145"	
jhead	3.04	4	1"	2"	288"	291"	25"	
jasper	4.0.0	34	37"	16"	84"	137"	137"	
pdfalto	0.4	76	342"	116"	107"	565"	282"	
gzip	1.12	6	6"	1"	19"	26"	26"	
v8	8.5.188	3,586	1"	5833"	874"	6708"	6708"	

We can combine VIPER with other tools for automatic exploit generation

Case Study: New Attack on V8

V8: Chromium JavaScript engine

- Used in Google Chrome, Microsoft Edge, Opera, Node.js …
- 3,586 KLoC in the latest version

VIPER result

- 2 potential syscall-guard variables
- 1 highly corruptible variable
  - Location: global variable
  - Memory-Write instructions: 93,512,607

Case Study: New Attack on V8

```
1 void Shell::AddOSMethods(Isolate* isolate,
2                           Local<ObjectTemplate> os_templ) {
3   if (options.enable_os_system) {
4     os_templ->Set(isolate, "system",
5       FunctionTemplate::New(isolate, System));
6   } ...
7 }
```

Our Attack (CVE-2021-30632)

- Arbitrary read privilege
  - Bypass ASLR
- Arbitrary write privilege
  - Set options.enable_os_system to 1

Demo

Link to data-only attack on v8

Conclusion

- *VIPER*: automatically spotting syscall-guard variables for data-only attacks
  - Design branch force and corruptibility assessment
  - Find 34 previous unknown syscall-guard variables
  - Build 4 new data-only attacks on SQLite and V8
- Open Source
  - VIPER: https://github.com/psu-security-universe/viper
  - Exploits: https://github.com/psu-security-universe/data-only-attacks

Thank You

Question?

hengkai@psu.edu