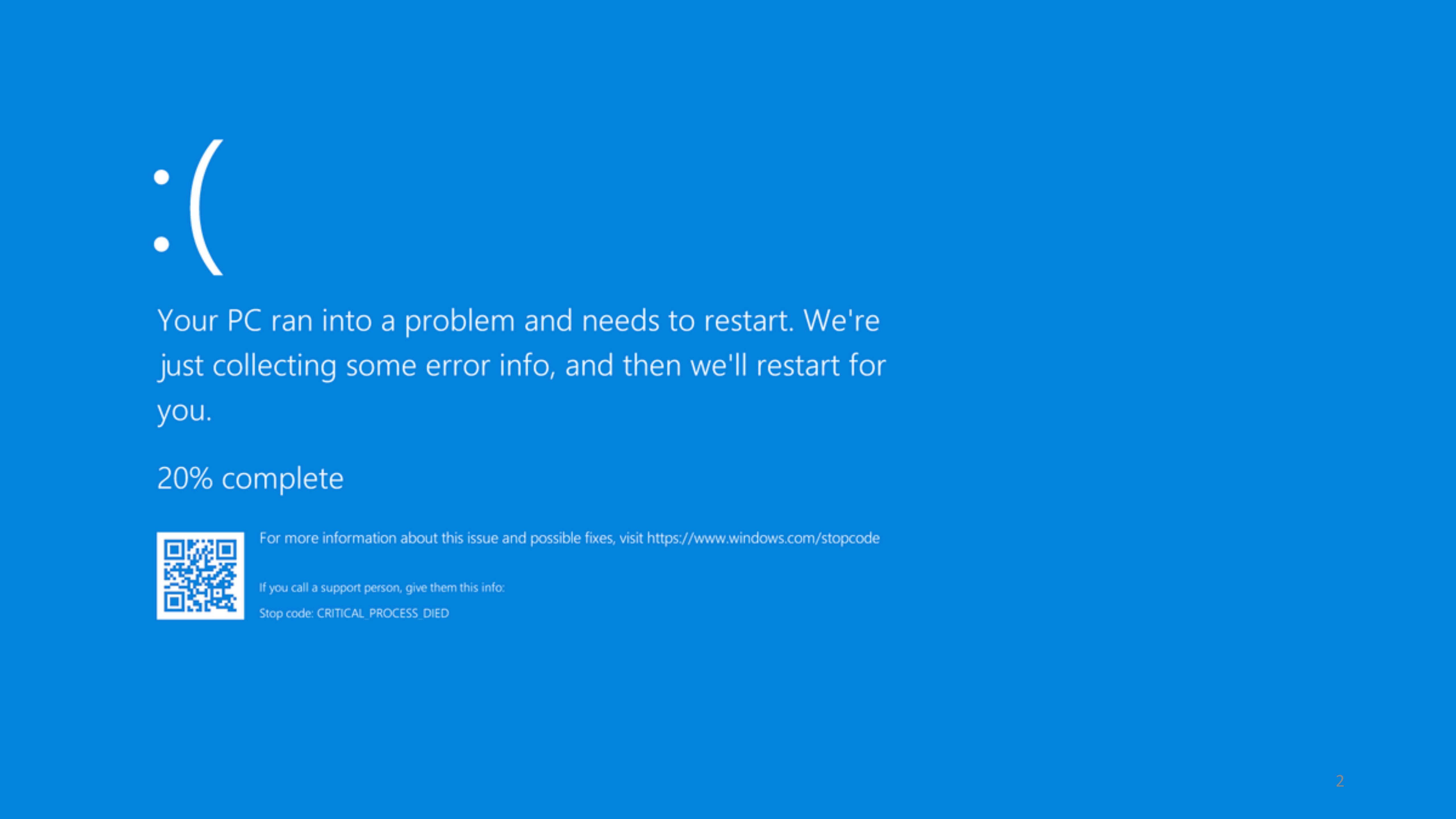
DESENSITIZATION:

Privacy-Aware and Attack-Preserving Crash Report

Ren Ding, Hong Hu, Wen Xu, Taesoo Kim

Georgia Institute of Technology



:(

Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

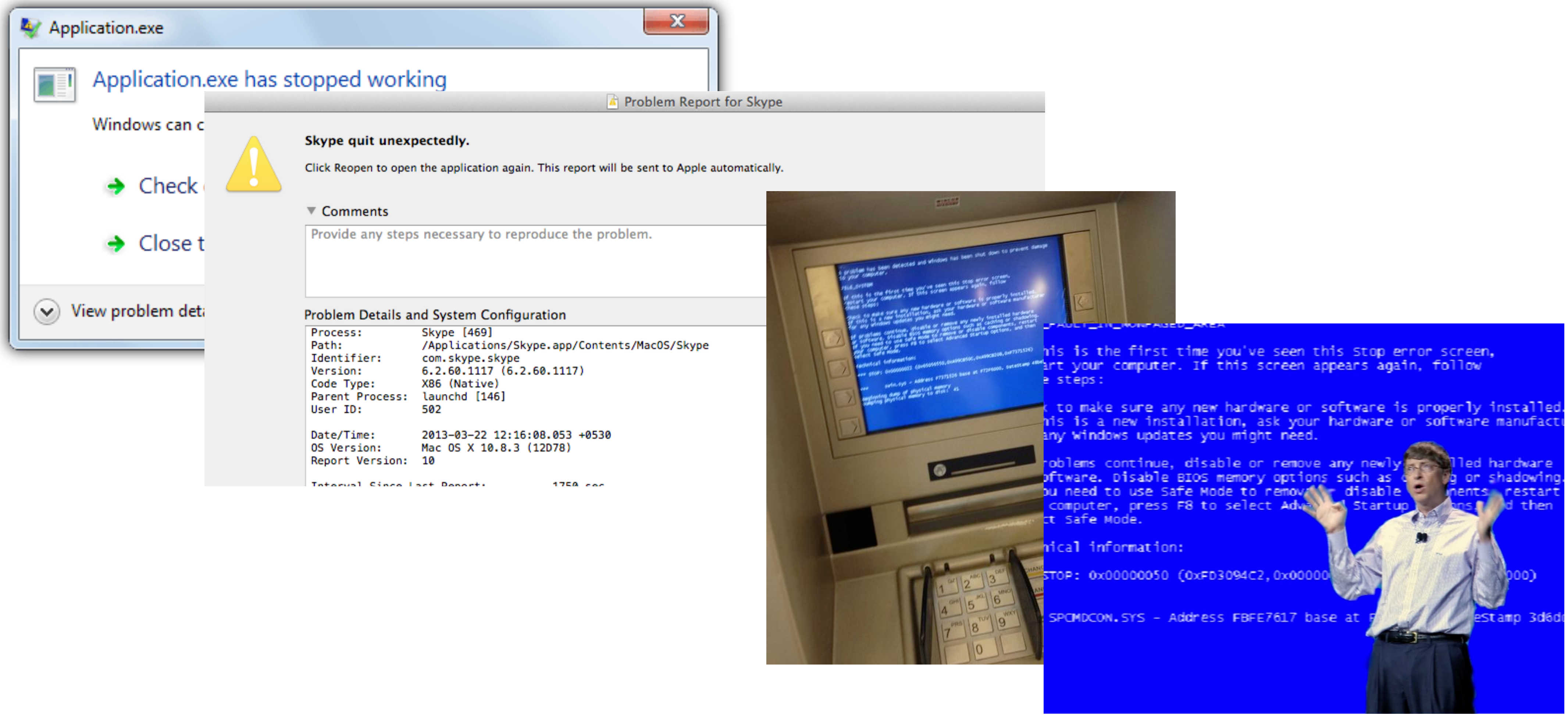
20% complete

For more information about this issue and possible fixes, visit https://www.windows.com/stopcode

If you call a support person, give them this info:

Stop code: CRITICAL_PROCESS_DIED

Software Crashes



Application.exe

Application.exe has stopped working

Windows can c

➔ Check

➔ Close t

View problem det

Problem Report for Skype

Skype quit unexpectedly.

Click Reopen to open the application again. This report will be sent to Apple automatically.

▼ Comments

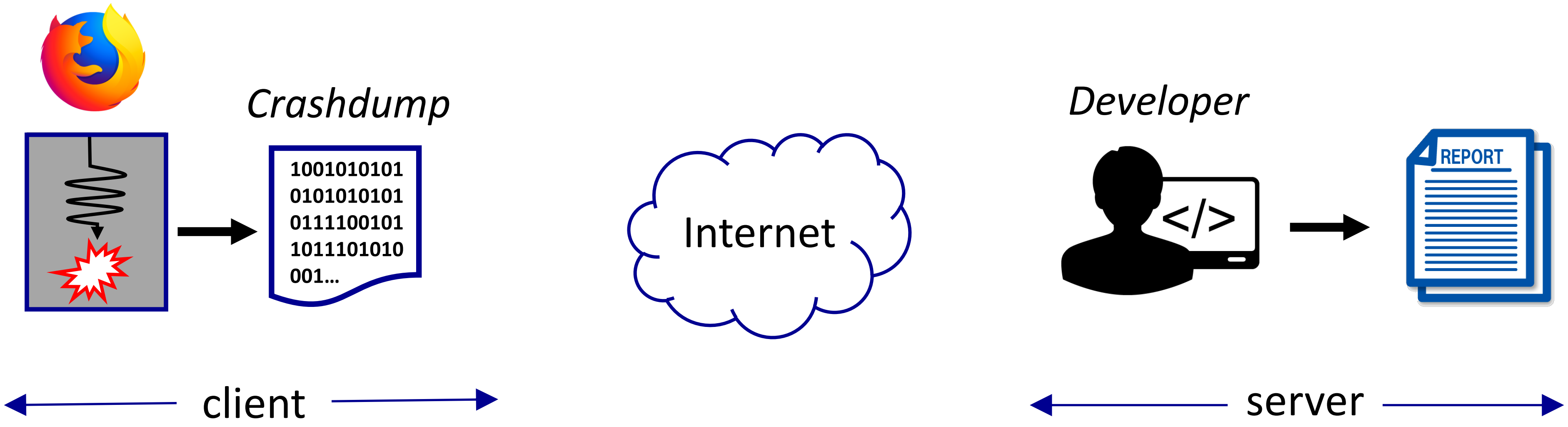
Provide any steps necessary to reproduce the problem.

Problem Details and System Configuration

```
Process:         Skype [469]
Path:            /Applications/Skype.app/Contents/MacOS/Skype
Identifier:      com.skype.skype
Version:         6.2.60.1117 (6.2.60.1117)
Code Type:       X86 (Native)
Parent Process:  launchd [146]
User ID:         502

Date/Time:       2013-03-22 12:16:08.053 +0530
OS Version:      Mac OS X 10.8.3 (12D78)
Report Version:  10

Interval Since Last Report:          1750 sec
```

```
_FAULT_IN_NONPAGED_AREA

his is the first time you've seen this Stop error screen,
art your computer. If this screen appears again, follow
e steps:

k to make sure any new hardware or software is properly installed.
his is a new installation, ask your hardware or software manufactu
any Windows updates you might need.

roblems continue, disable or remove any newly     lled hardware
oftware. Disable BIOS memory options such as c      g or shadowing.
ou need to use Safe Mode to remov   r disable      nents, restart
 computer, press F8 to select Adv     Startup      ons, a d then
ct Safe Mode.

nical information:

STOP: 0x00000050 (0xFD3094C2,0x00000                        000)


 SPCMDCON.SYS - Address FBFE7617 base at F                  eStamp 3d6d
```

After a Crash …



Crashdump

1001010101
0101010101
0111100101
1011101010
001…

Internet

Developer

client

server

Crash Dump

- A memory dump contains
  - CPU registers
  - Memory snapshot
  - Execution environment

- Formats
  - Coredump, Minidump

What could be wrong?

Privacy leakage

Recent Study

2.5M crash reports contain private user data!

- ➢20K sessions tokens
- ➢700 passwords
- ➢9K emails

We need a Privacy-Aware Crash Reports

Potential Solutions

- Manual annotation
  - Time-consuming
  - Program-dependent
- Pattern-based searching
  - Error-prone
- Input-logging
  - Heavy computation
  - Incompatible

Our Solution: Desensitization

A new crash-reporting framework

- privacy-aware
- attack-/bug-preserving
- practical, extensible

Key Observation

Adopts generalized features from existing analysis techniques!

Existing Crash Analysis & Triage Techniques

Techniques	Callstack	IP	Signature	RevExec
Adobe-CR	✓		✓	
Apport	✓		✓	
Backtrace	✓		✓	
Chrome-CR	✓		✓	
CREDAL				✓
CrashGraphs	✓			
KLEE	✓	✓		
Liblit et al.	✓	✓		
Mac-CR	✓		✓	
Modani et al.	✓			
POMP				✓
Rebucket	✓			
REPT				✓
RETracer				✓
Schroter et al.	✓			
Socorro	✓		✓	
WER	✓		✓	
!analyze	✓		✓	
DESENSITIZATION	yes	yes	yes	

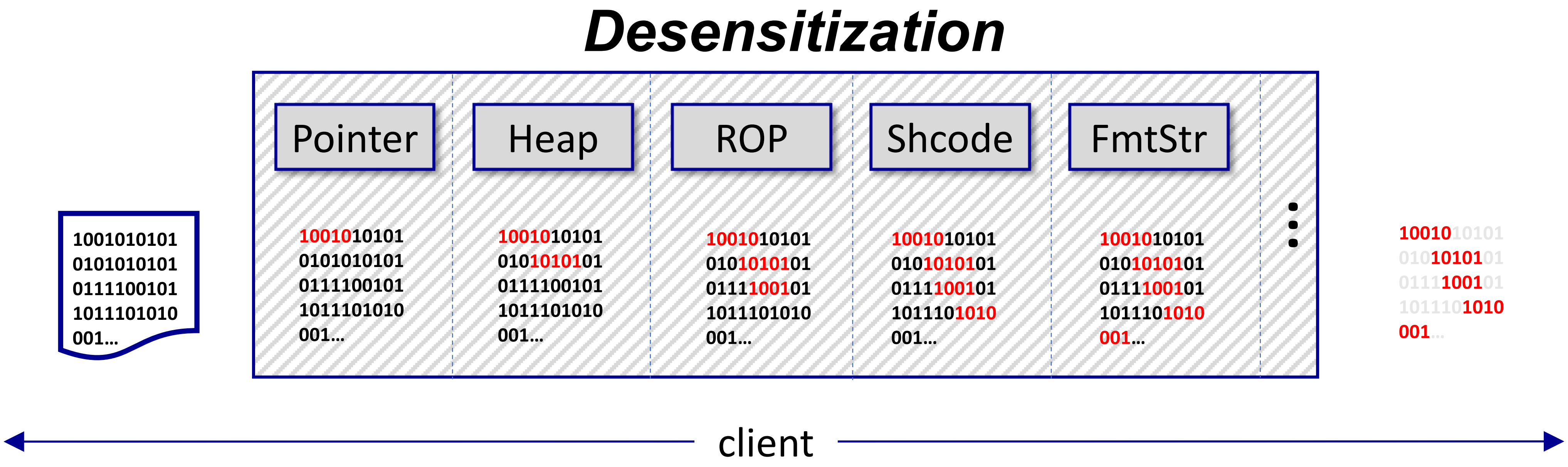
- IP-based
- Callstack-based
- Signature-based
- Fault-based

Existing Anomaly Detection Schemes

- Heap metadata
- Program-specific structures
- ROP gadgets
- Shellcode payload

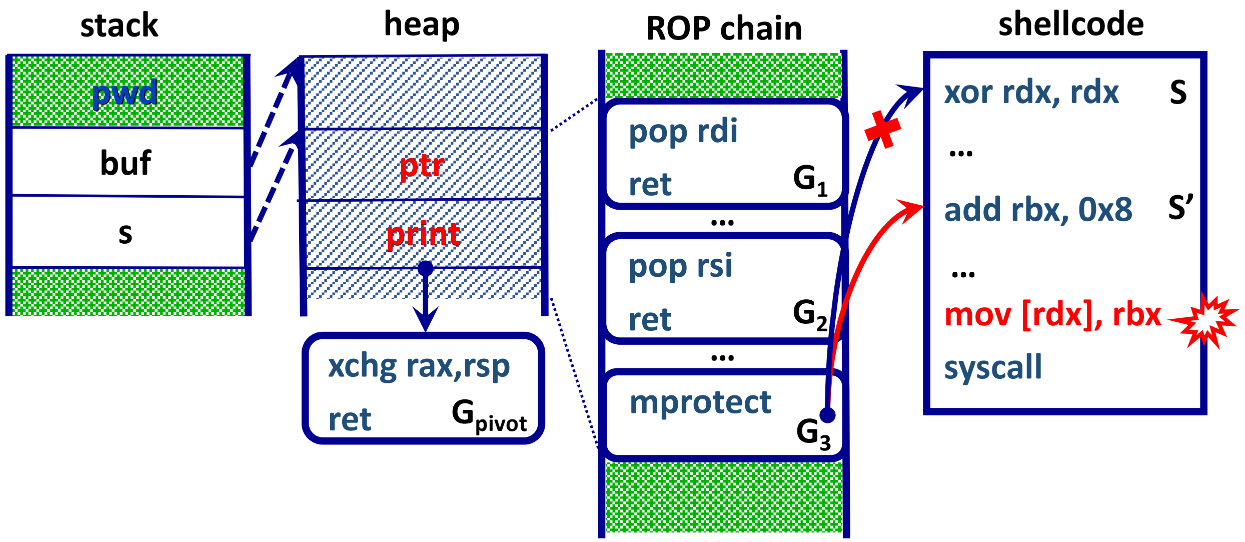
Schemes	Heap	Struct.	ROP	Shcode
CAVER		✓		
DANGNULL	✓			
HOTracer	✓			
KOP		✓		
Polychronakis et al.				✓
ROPecker			✓	
ROPMEMU			✓	
ROPScan			✓	
SBCFI			✓	
SHELLOS			✓	
SigGraph		✓		
DESENSITIZATION	yes	yes	yes	yes

Overview


Desensitization
1001010101
0101010101
0111100101
1011101010
001…
Pointer
Heap
ROP
Shcode
FmtStr
1001010101
0101010101
0111100101
1011101010
001…
1001010101
0101010101
0111100101
1011101010
001…
1001010101
0101010101
0111100101
1011101010
001…
1001010101
0101010101
0111100101
1011101010
001…
1001010101
0101010101
0111100101
1011101010
001…
1001010101
0101010101
0111100101
1011101010
001…
client

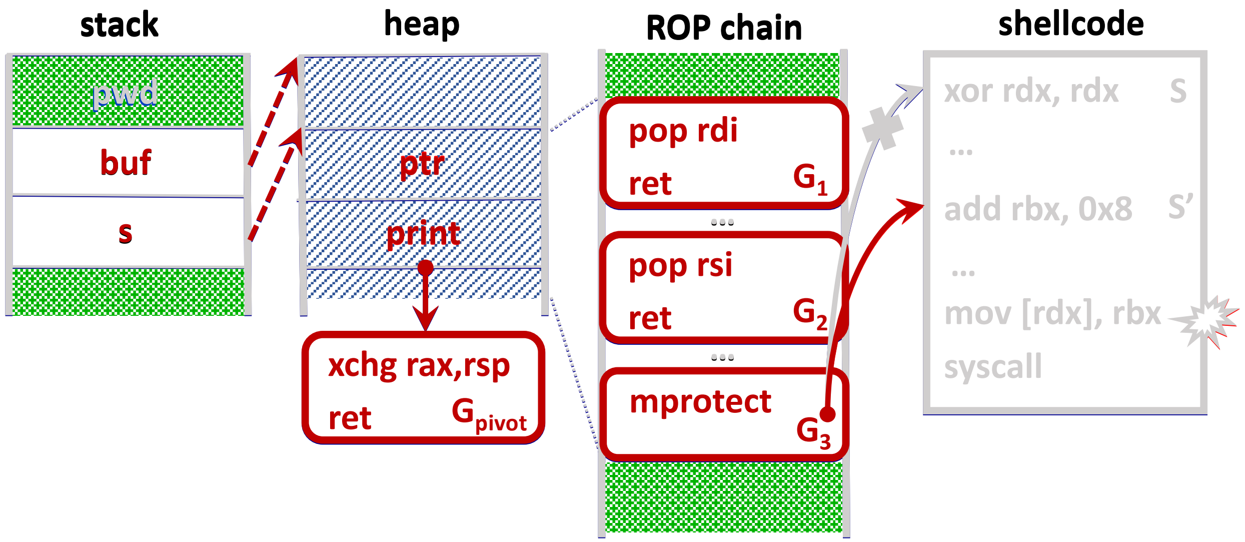
Motivating Example

```
#define MAX_LEN 64
typedef struct { char *ptr; void (*print)(); } String;
void printString() { ... }
void vuln(char *input) {
    char pwd[MAX_LEN];  load_passwd(pwd);
    char *buf = (char *) malloc(MAX_LEN);
    String *s = (String *) malloc(sizeof(String));
    s->ptr = buf;  s->print = &printString;
    strcpy(buf, input);
    s->print();
}
```



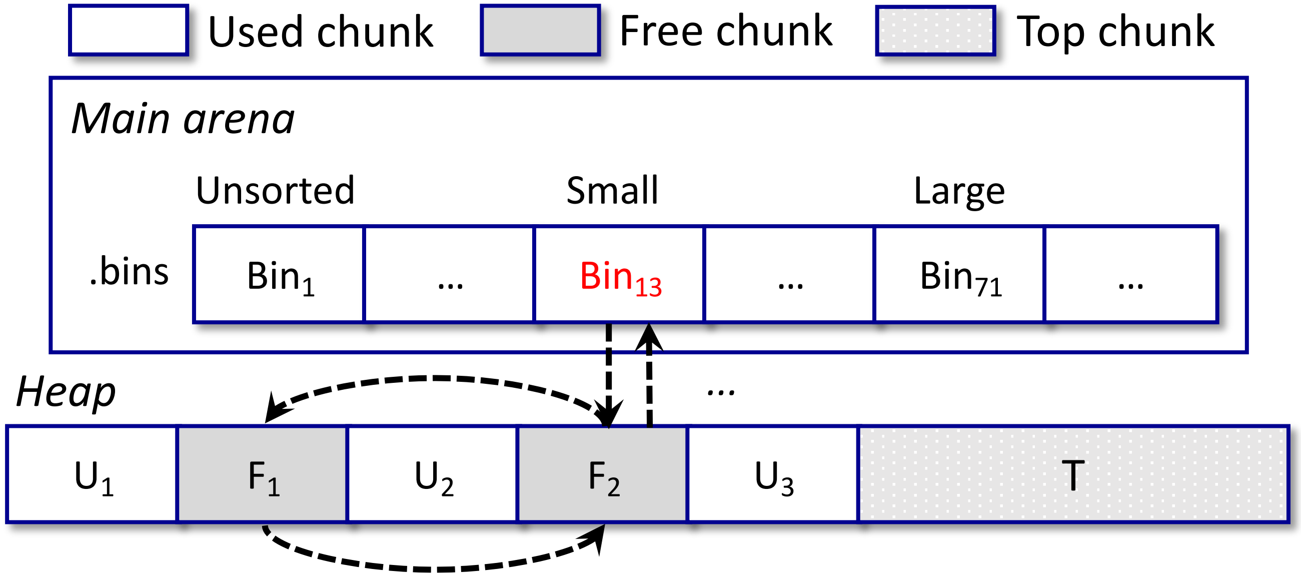
Pointer Identification

- Pointers
  - Scan every 4/8 bytes, ***if and only if*** points to a valid memory region
- Code ptr vs. data ptr
  - Proper access permission



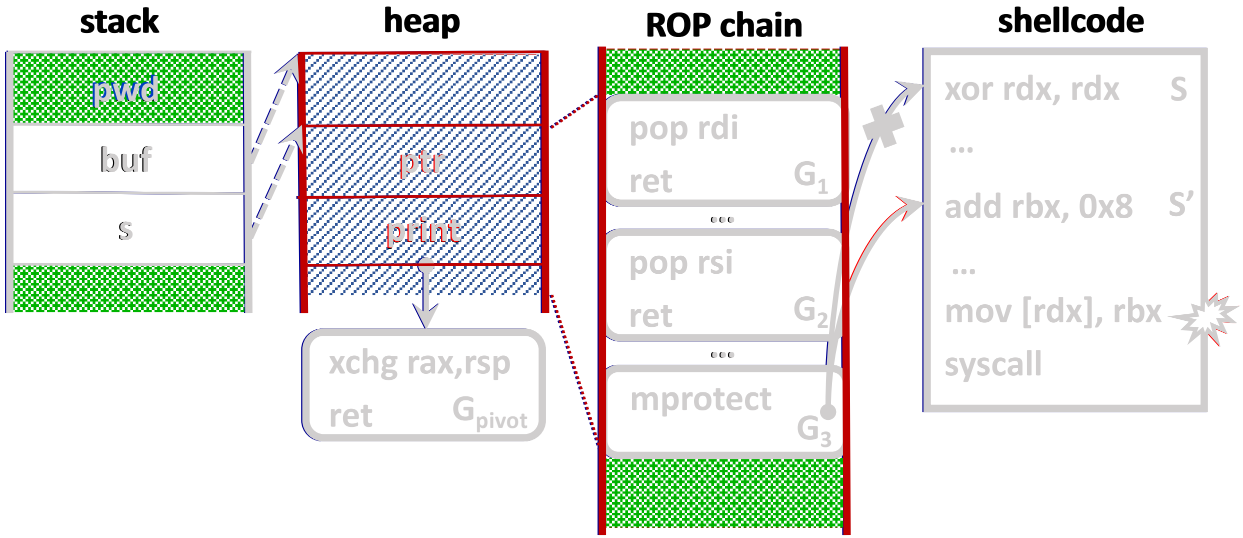
Heap Structures

- ptmalloc
  - arenas
  - bins
  - chunk metadata



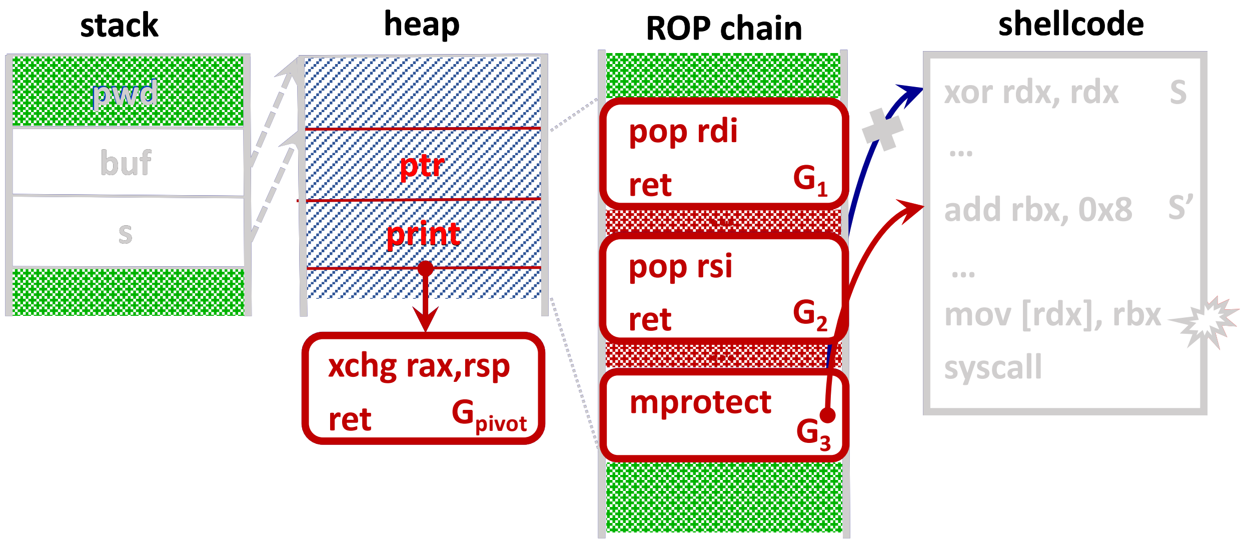
Heap Structures

- ptmalloc
  - arenas
  - bins
  - chunk metadata
- jemalloc



ROP Chains

- Save non-pointer data in-between pointers
  - ***if and only if*** there are [>=N] code pointers in [K] bytes



Extensible!

Modules	Collected Data	Related Bugs & Attacks
Pointer	code pointers	ROP, JOP, COP, GOT.PLT corruption, vtable injection, etc.
	data pointers	DOP, type confusion, UAF, etc.
Heap	chunk size	heap ovf, overlapping chunks, heap spray, etc.
	chunk status	UAF, double-free, unsafe-unlink, etc.
ROP	gadgets & args	ROP, JOP, COP, etc.
Fmtstr	strings & args	format string attack
Shcode	payloads	shellcode injection

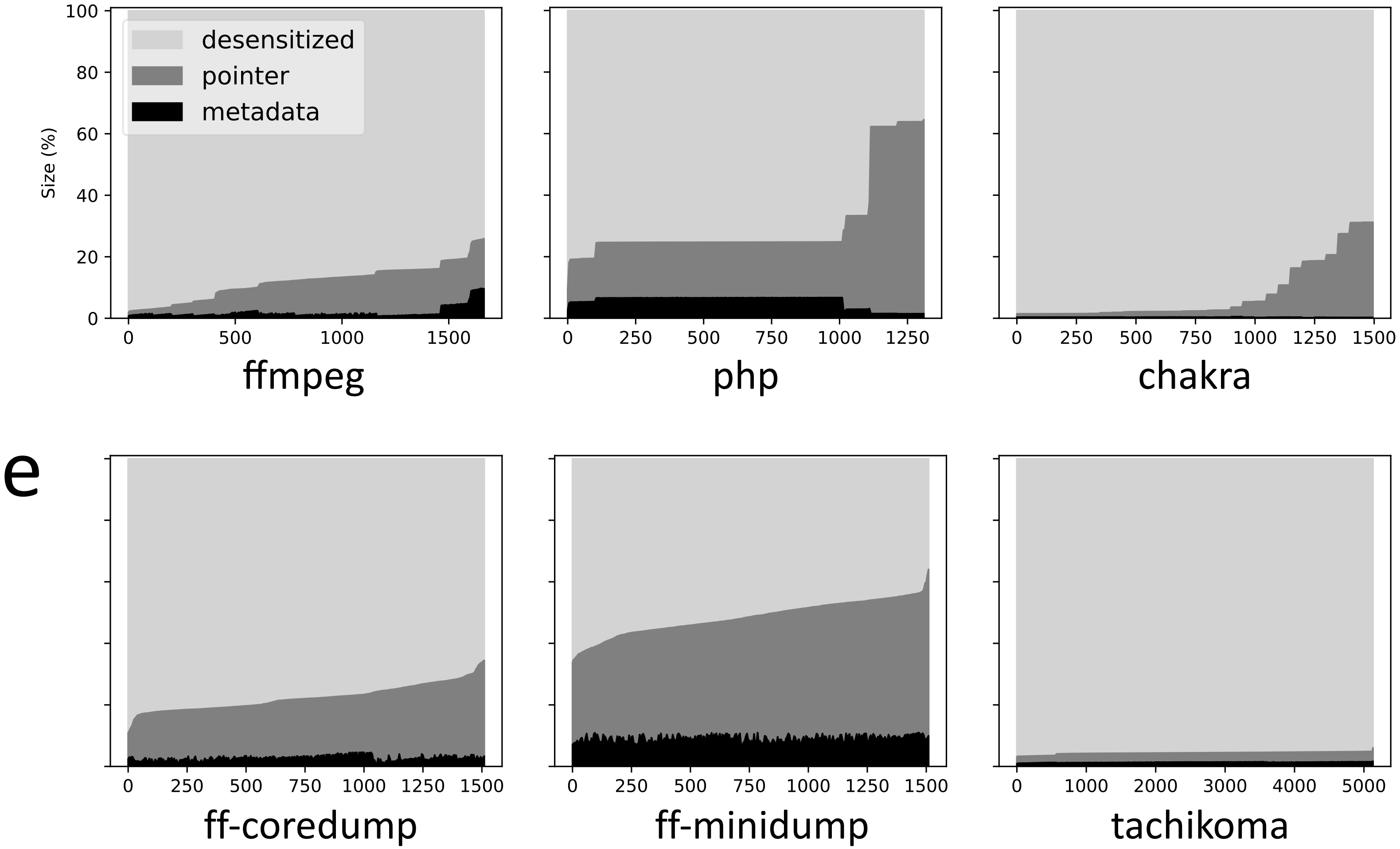
Tools in Action: Experimental Setup

- Hard to find raw crashes
  - Fuzzing comes to rescue!

- 13,390 crashes collected
  - Normal crashes: 7507
  - Attack-relevant crashes: 5883

Tools in Action: Privacy-Awareness

- Coredumps:
  - 80.9% nullified
- Minidumps:
  - 49.0% nullified
- Pointers dominate



ffmpeg

php

chakra

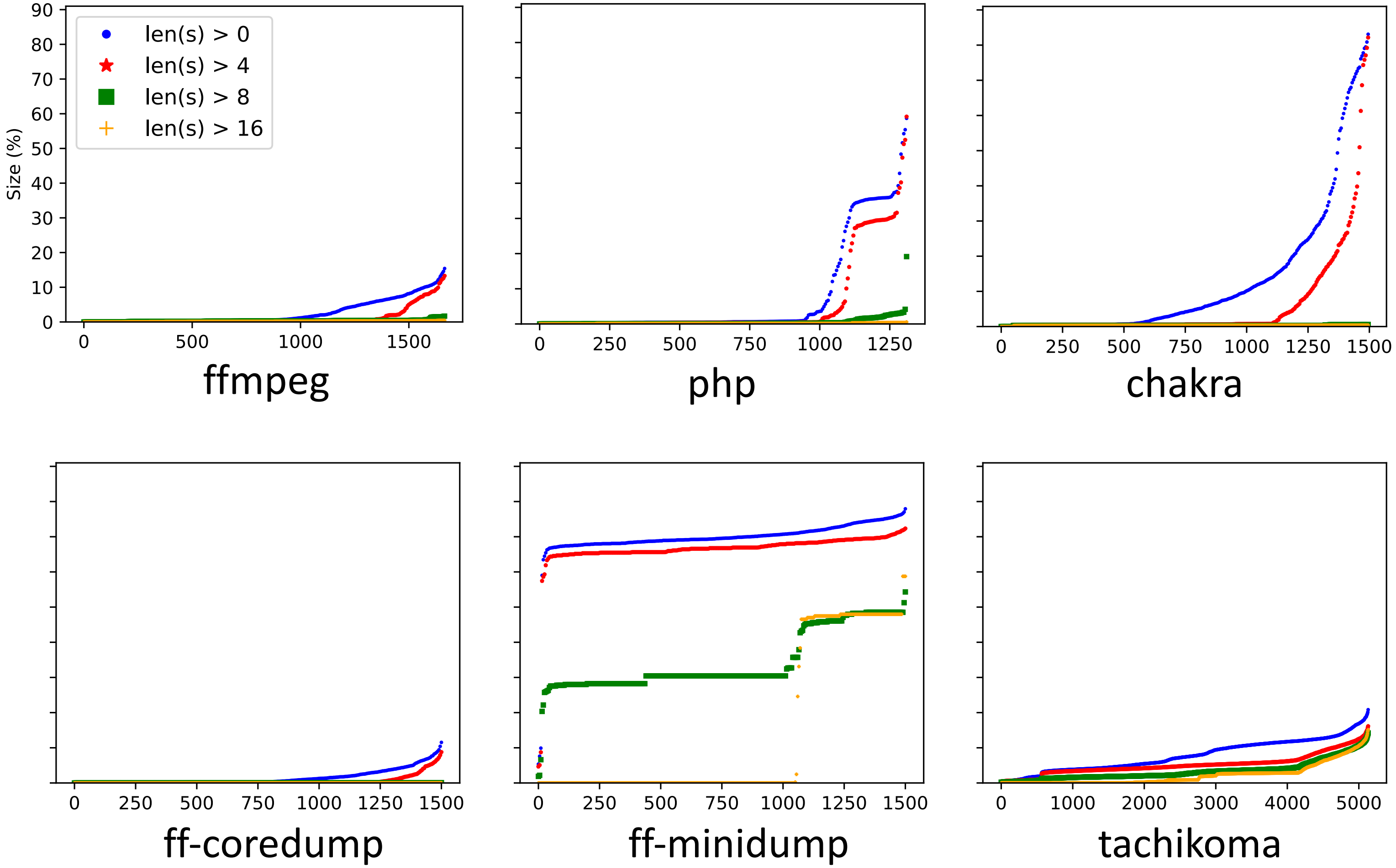
ff-coredump

ff-minidump

tachikoma

Tools in Action: Privacy-Awareness

- Printable strings
- Coredumps:
  - 95.0% removed
- Minidumps:
  - 37.2% removed



Case Study: Leakage in Firefox

```
class nsTAutoStringN : public nsTString<T> {
    public:
        nsTAutoStringN() : string_type(mStorage, ...)
        static const size_t kStorageSize = N;
    private:
        char_type mStorage[N];
}
```

```
nsresult nsAutoCompleteController::EnterMatch( ... ) {
    ...
    nsAutoString value;
    if (selectedIndex >= 0) {
        GetResultValueAt(selectedIndex, true, value);
    } else if (shouldComplete) {
        GetFinalDefaultCompleteValue(value)
    }
    ...
}
```

Tools in Action: Bug- & Attack-Preservation

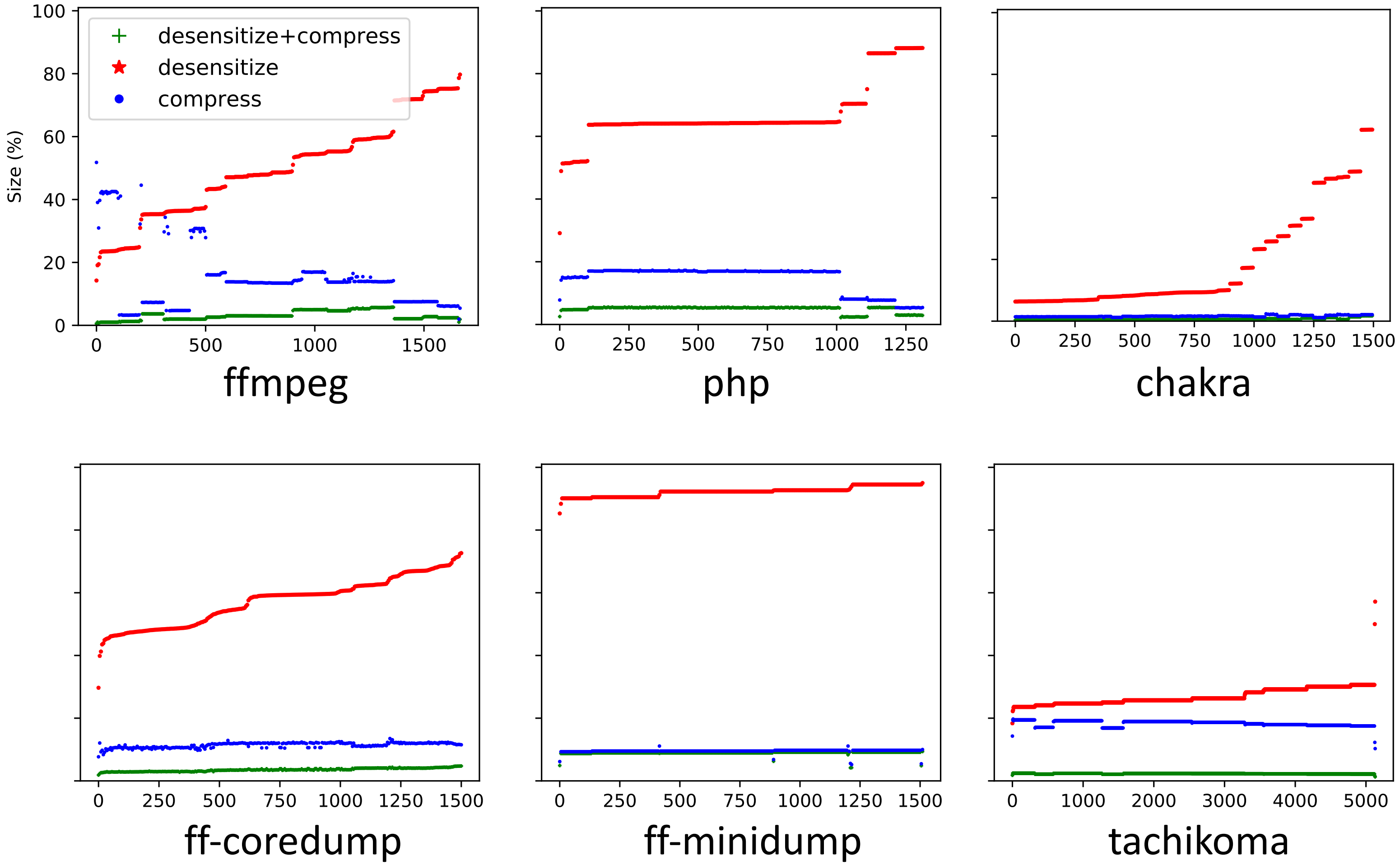
- Normal crashes:
  - State-of-the-art classification tools
  - e.g., Socorro, Backtrace

- Attack-relevant crashes:
  - Self-implemented attack detection tools

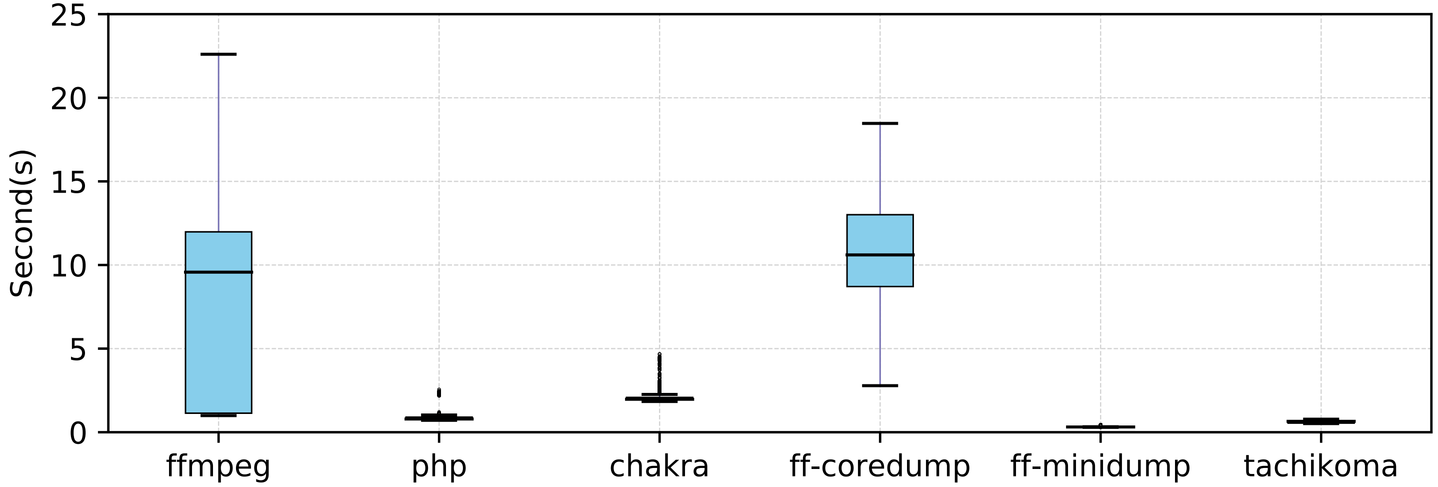
- Same result before and after desensitization

Tools in Action: File Size Reduction

- Coredumps:
  - 44.2% reduced
- Minidumps:
  - 7.7% reduced



Tools in action: Efficiency



- < 15 seconds
- Size & complexity matters

Summary

- DESENSITIZATION generates privacy-aware and bug-/attack-preserving crash reports

Summary

- DESENSITIZATION generates privacy-aware and bug-/attack-preserving crash reports
    - Program-independent

Summary

- DESENSITIZATION generates privacy-aware and bug-/attack-preserving crash reports
  - Program-independent
  - Resource-saving

Summary

- DESENSITIZATION generates privacy-aware and bug-/attack-preserving crash reports
  - Program-independent
  - Resource-saving
  - Practical, efficient

Summary

- DESENSITIZATION generates privacy-aware and bug-/attack-preserving crash reports
  - Program-independent
  - Resource-saving
  - Practical, efficient

Open-sourced at:
https://github.com/sslab-gatech/desensitization!