Safe Software and Systems

David Brumley
Assistant Professor
ECE & CS

Four Topics

1. Capture the Flag Team

2. SplitScreen: Malware Scanning at 2x the speed, ½ the memory

3. Automated Patch-based Exploit Generation

4. Current Research Thrusts

Plaid Parliament of Pwing

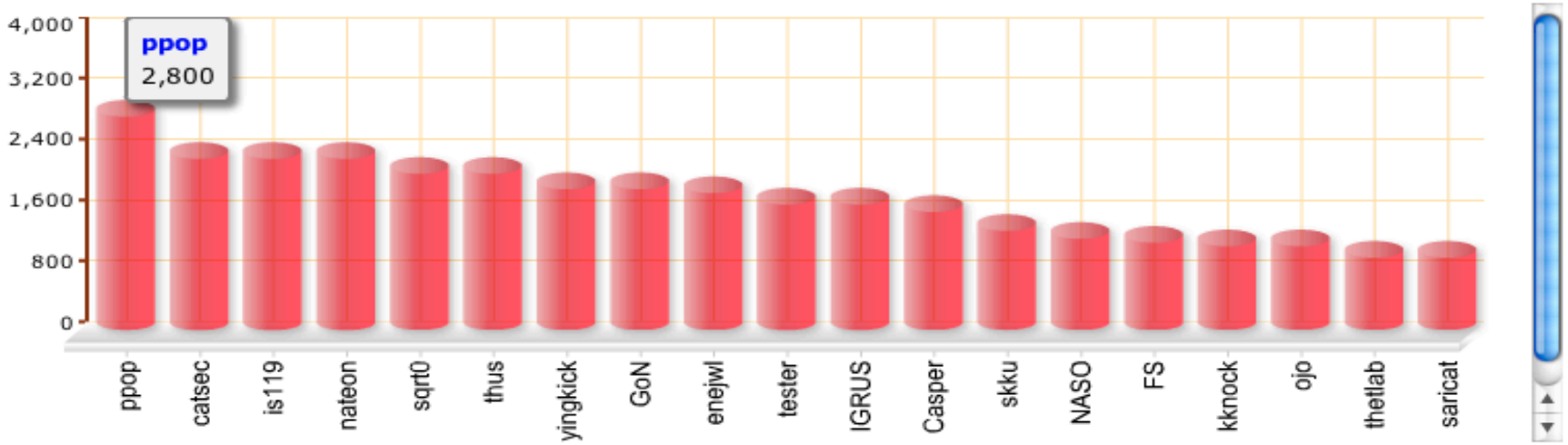
CMU's Capture the Flag Team

HackJam 2009

- Started 1am, Sept 19
- Ran 48 hours straight
- Over 100 teams entered from all over the world

H.U.S.T. Creative & Fun CTF 2009

- Started 5am Oct 6, 2009
- Ran 48 hours straight
- 8th year of competition
- Over 100 teams from around the world



순위	ID	점수	Pb.A	Pb.B	Pb.C	Pb.D	Pb.E	Pb.F	Pb.G	Pb.H	Pb.I	Pb.J	Pb.K	Pb.L	Pb.M	Pb.N	Pb.O
	ppop	2800	O	O	O	O	O	O	O	O	O	O	O	O	X	O	O

NYU-Poly CSAW Competition

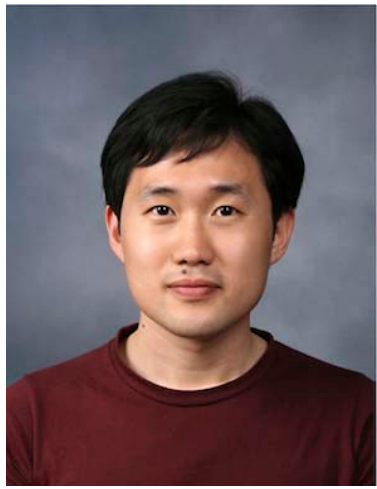
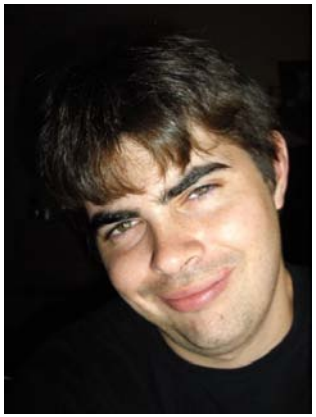
- 6th annual CTF
- Only officially open to undergrads
  - Grad team scores unofficial
- Winning team gets $500, trip to NYC

Place	Team	Score
1st	CMU Undergrad	16450
2nd	CMU Grad	9750
3rd	RPI	9600

≈71% higher than next undergrad team

Recruit Students with Skills







Sponsorship Opportunity

- Entering DefCON, the most prestigious CTF competition in the world
- 15 members, approx $1,200 per member
  - Travel
  - Hotel
  - Entrance fees

Hackjam: 3rd overall

HUST: 1st overall

CSAW: 1st in qualifying, 1st in finals

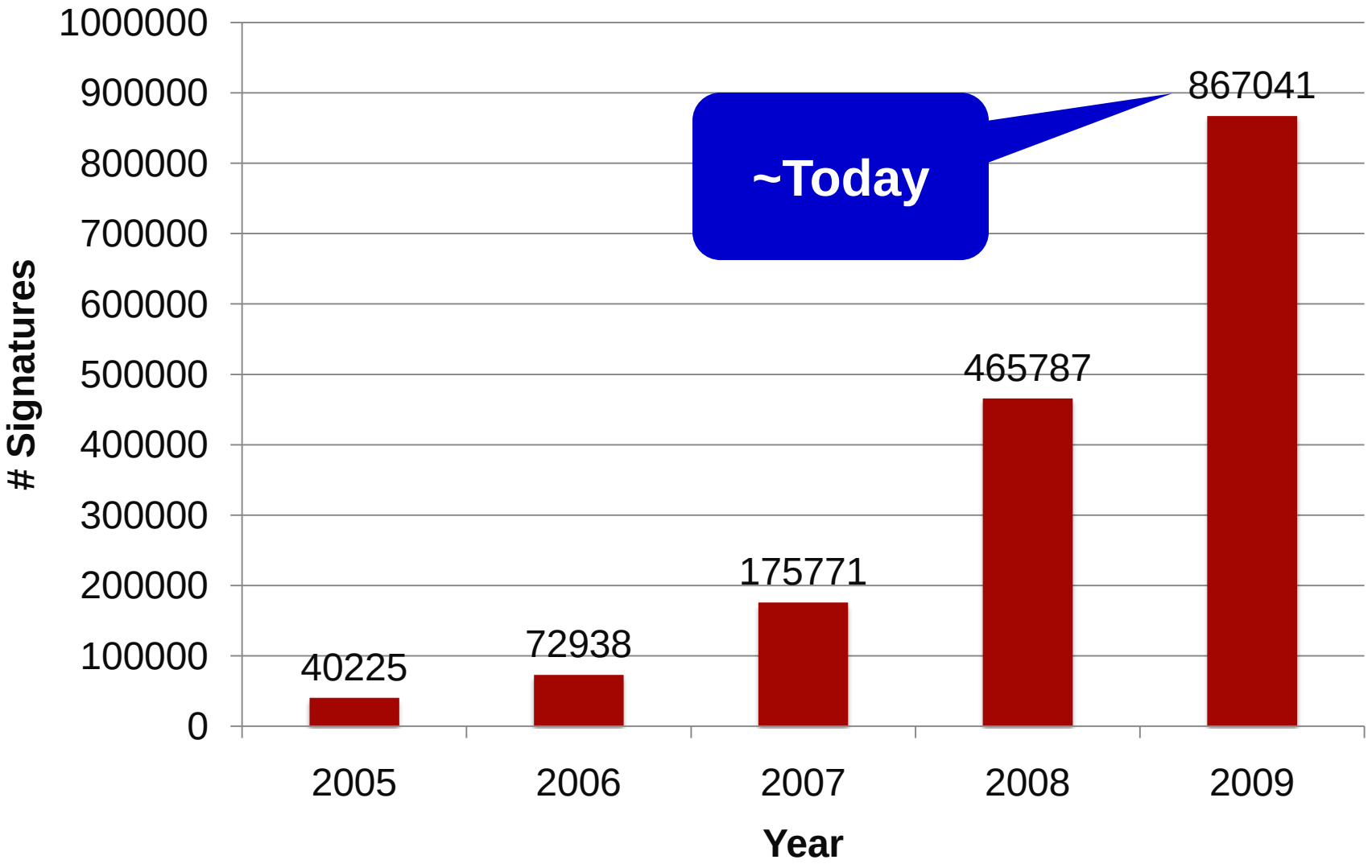
***New:* iCTF: 1st in US, 4th overall**

***New:* PadoCon: 3rd overall**

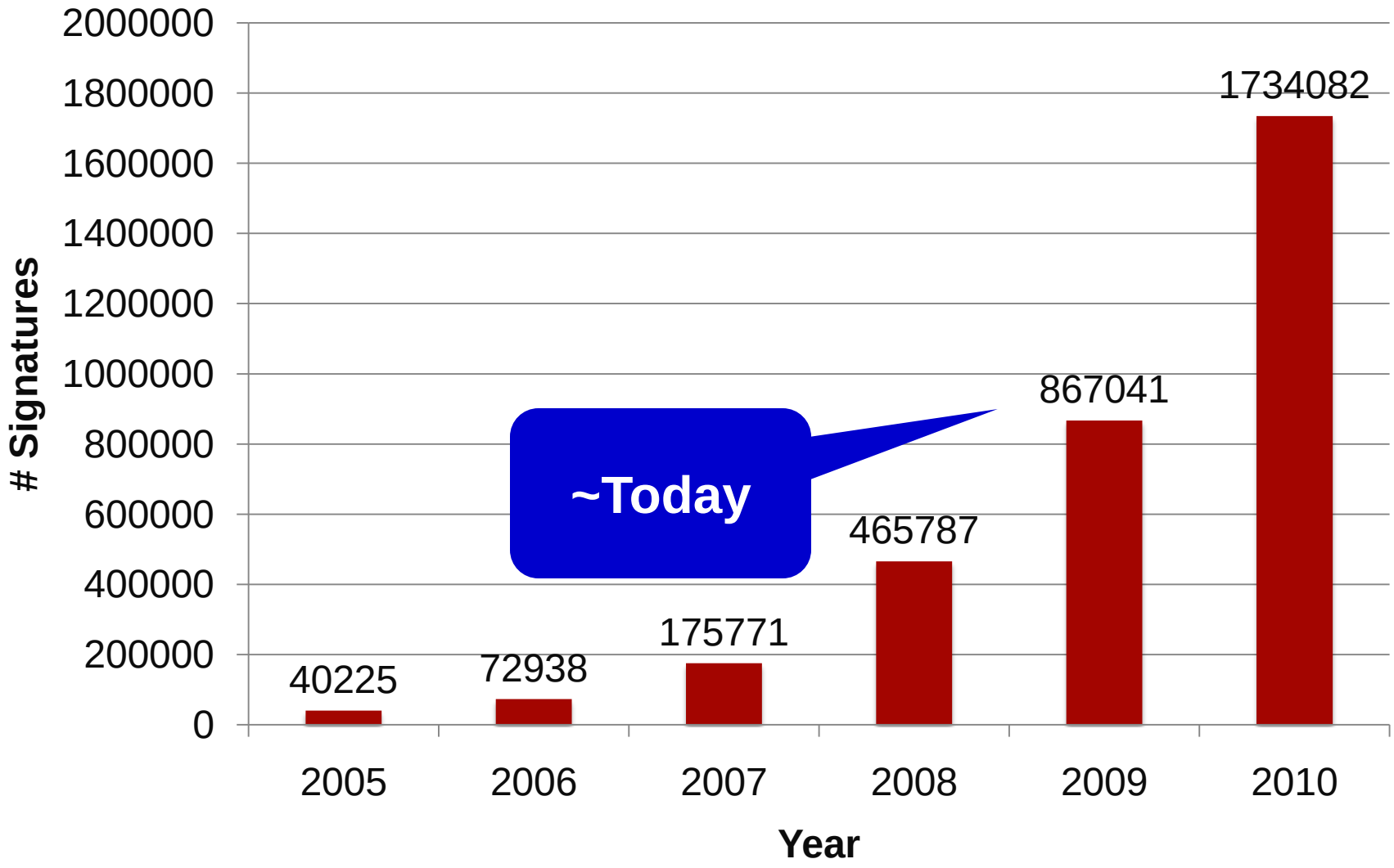
SplitScreen:

Malware Scanning at
2x the speed, ½ the memory

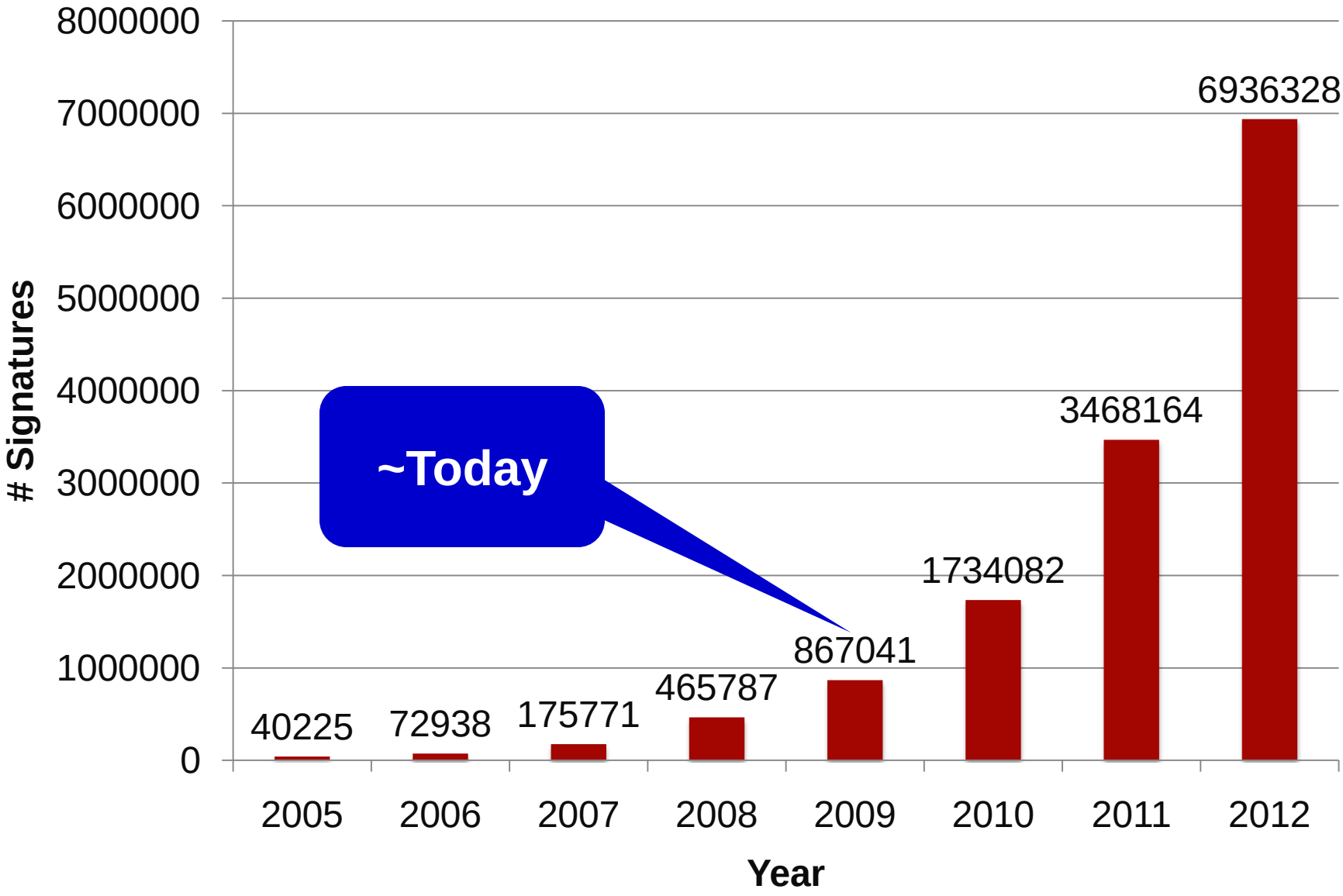
Number of Malware Signatures 2005-2009



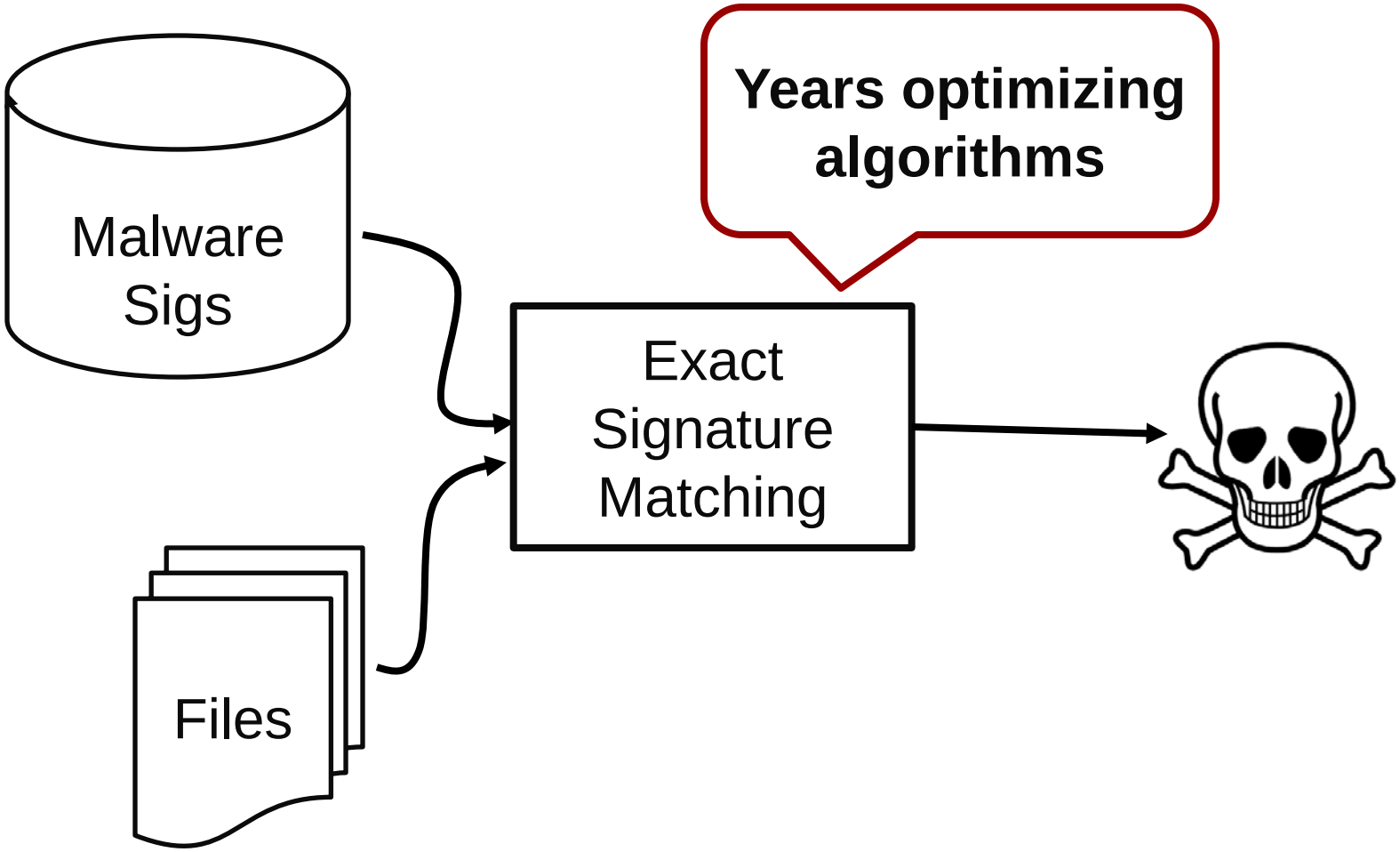
2010


2000000
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
Signatures
40225
72938
175771
465787
867041
1734082
~Today
2005
2006
2007
2008
2009
2010
Year

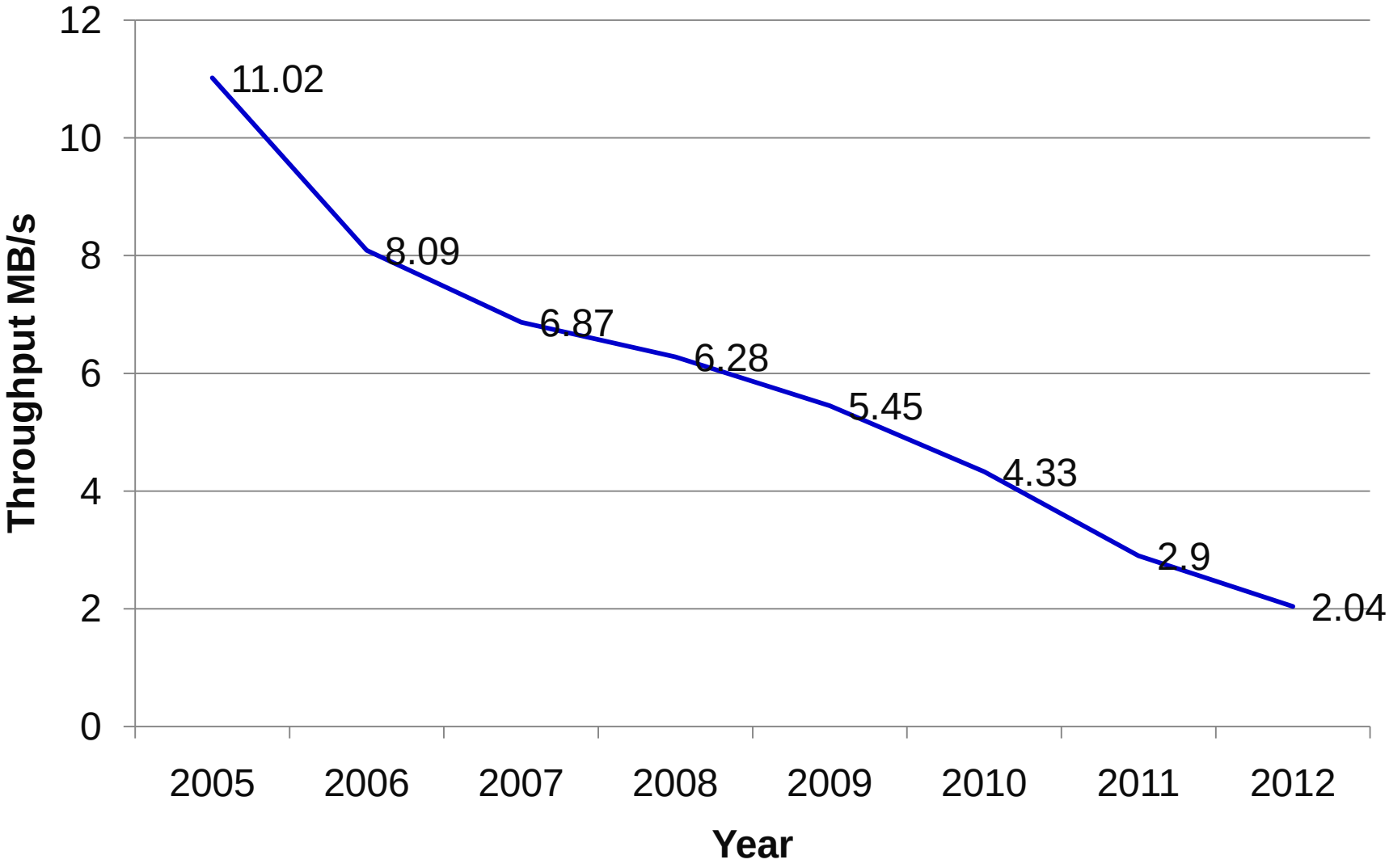
And Beyond


8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
Signatures
~Today
40225
72938
175771
465787
867041
1734082
3468164
6936328
2005
2006
2007
2008
2009
2010
2011
2012
Year

Typical Malware Scanning (ClamAV)



Scanning Throughput

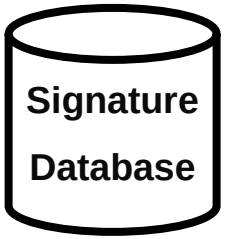


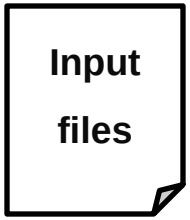
Current Anti-Malware Techniques Are Not Scaling

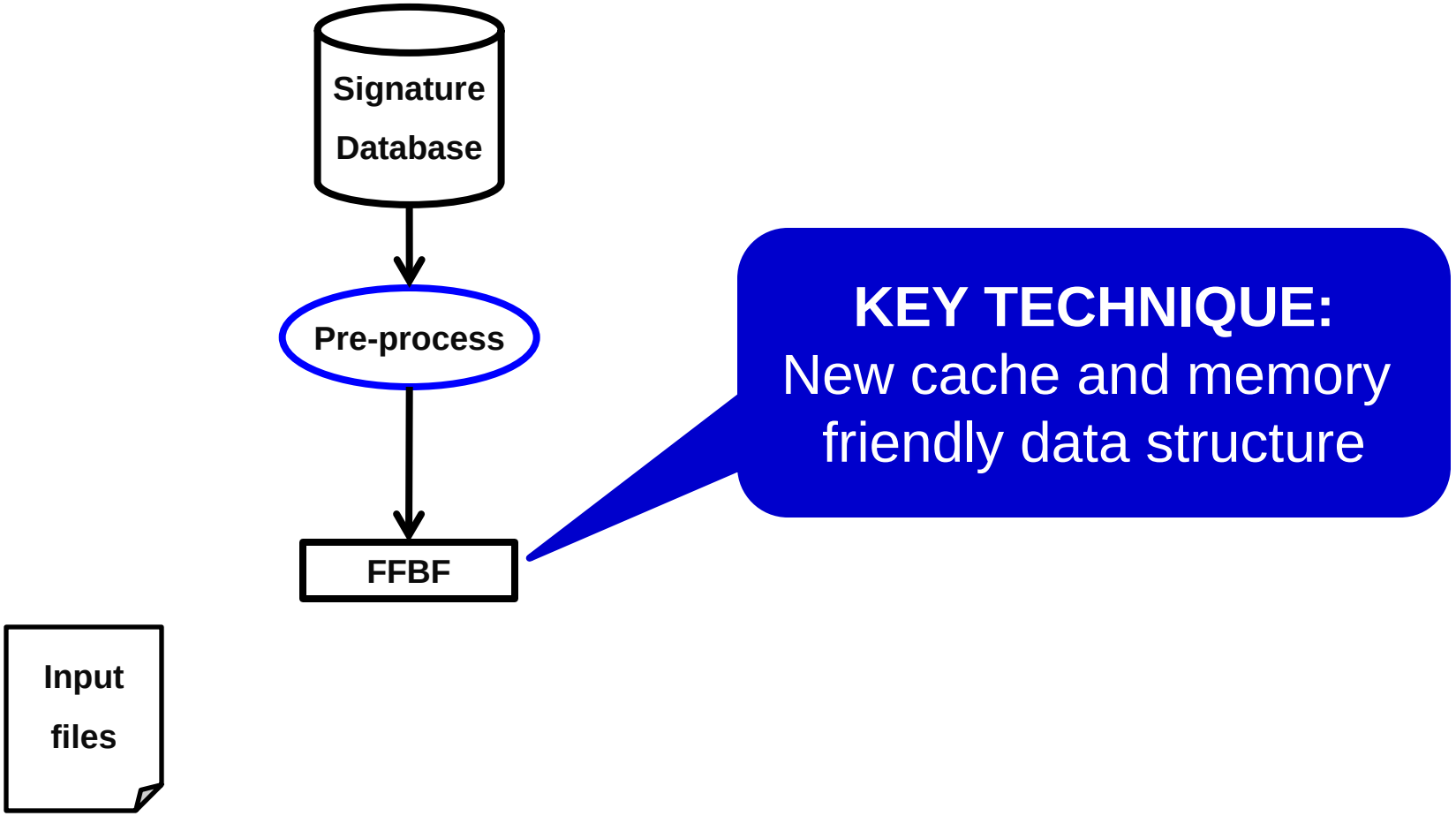
We Need Anti-Malware Scanning That

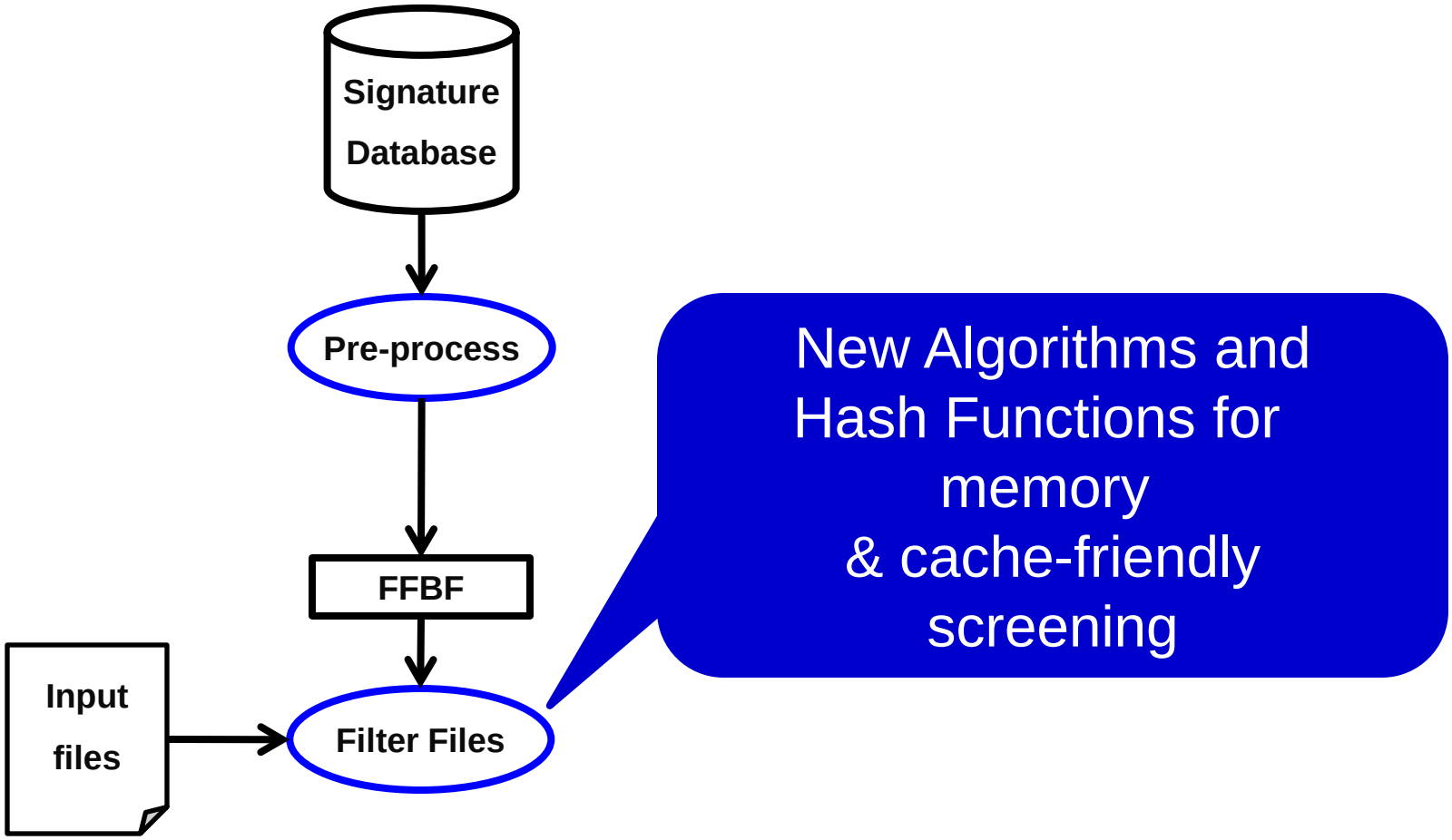
- Works with the millions of existing signatures
- Scales
- Supports emerging low-power systems
  - Netbooks, Cellphones, IPad

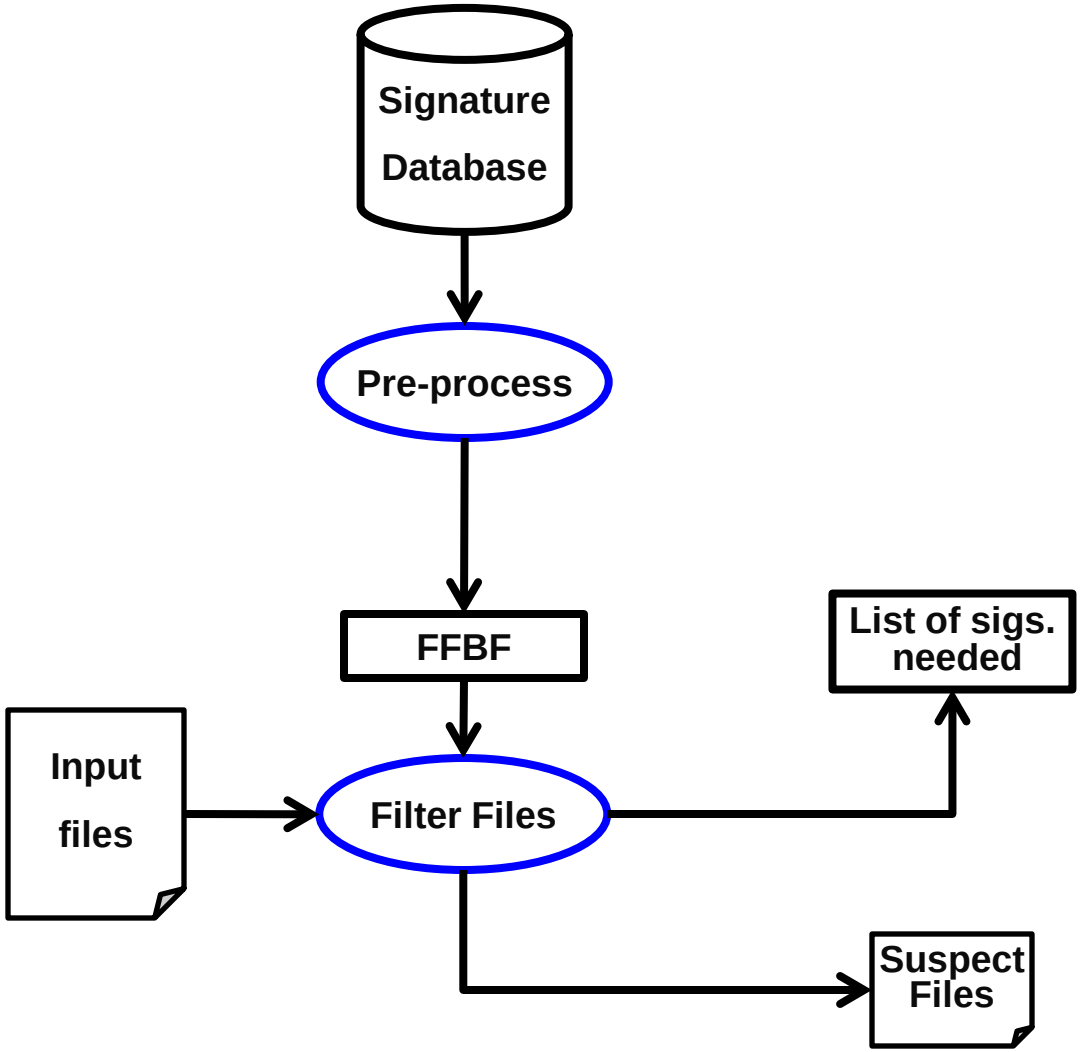
SplitScreen:
2x throughput at ½ the Memory

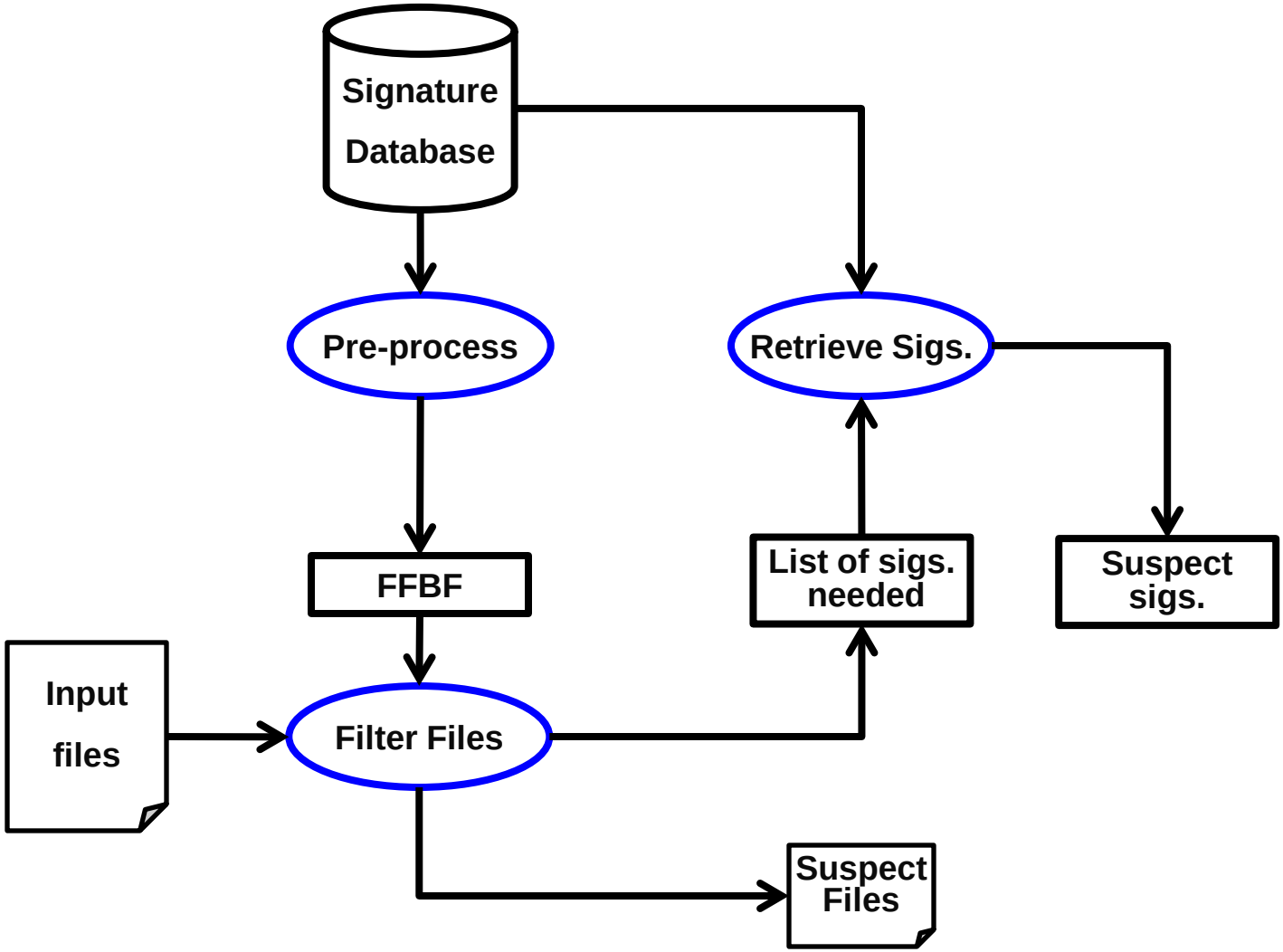

Signature
Database

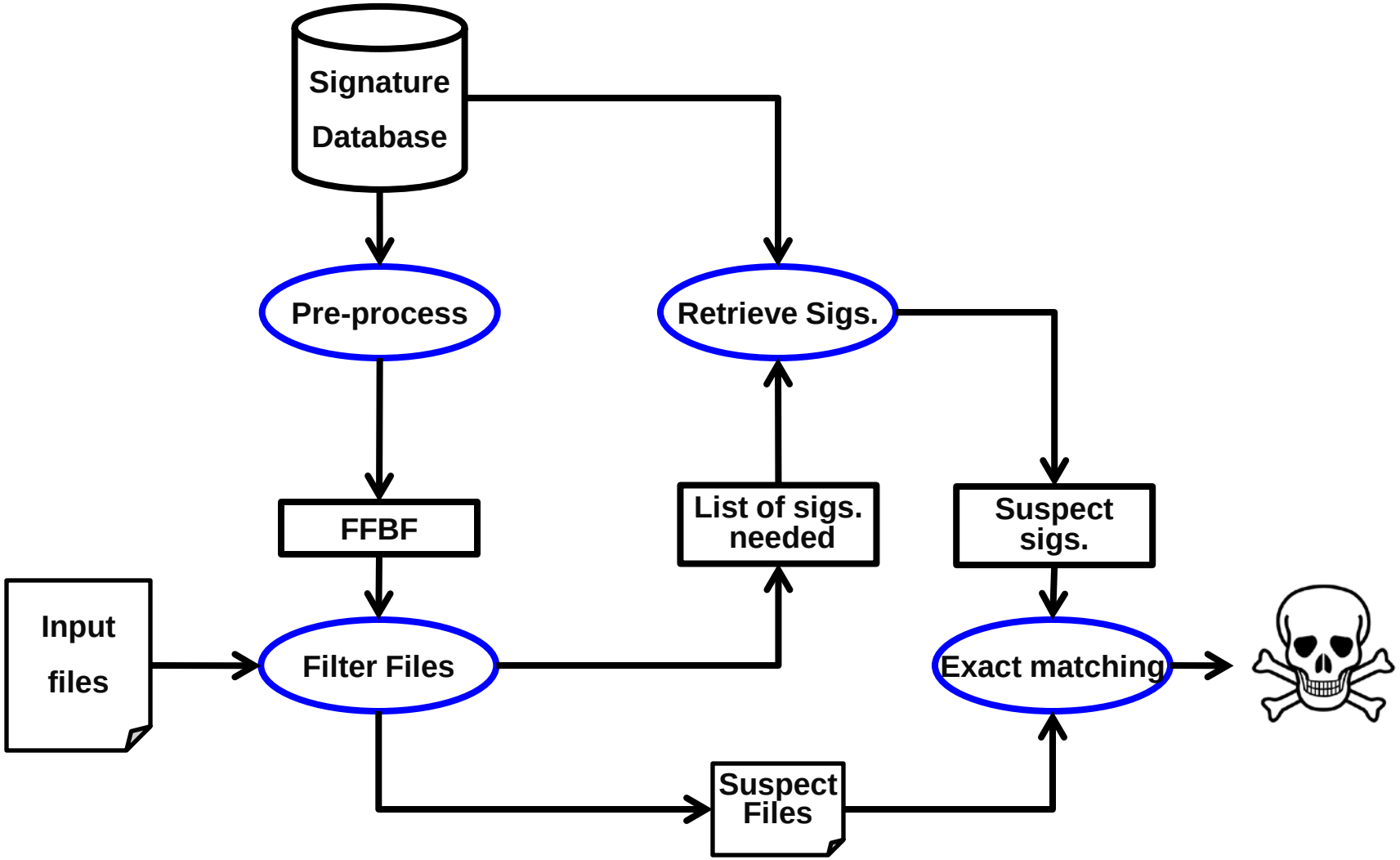

Input
files


Signature
Database
Pre-process
FFBF
Input
files
KEY TECHNIQUE:
New cache and memory
friendly data structure

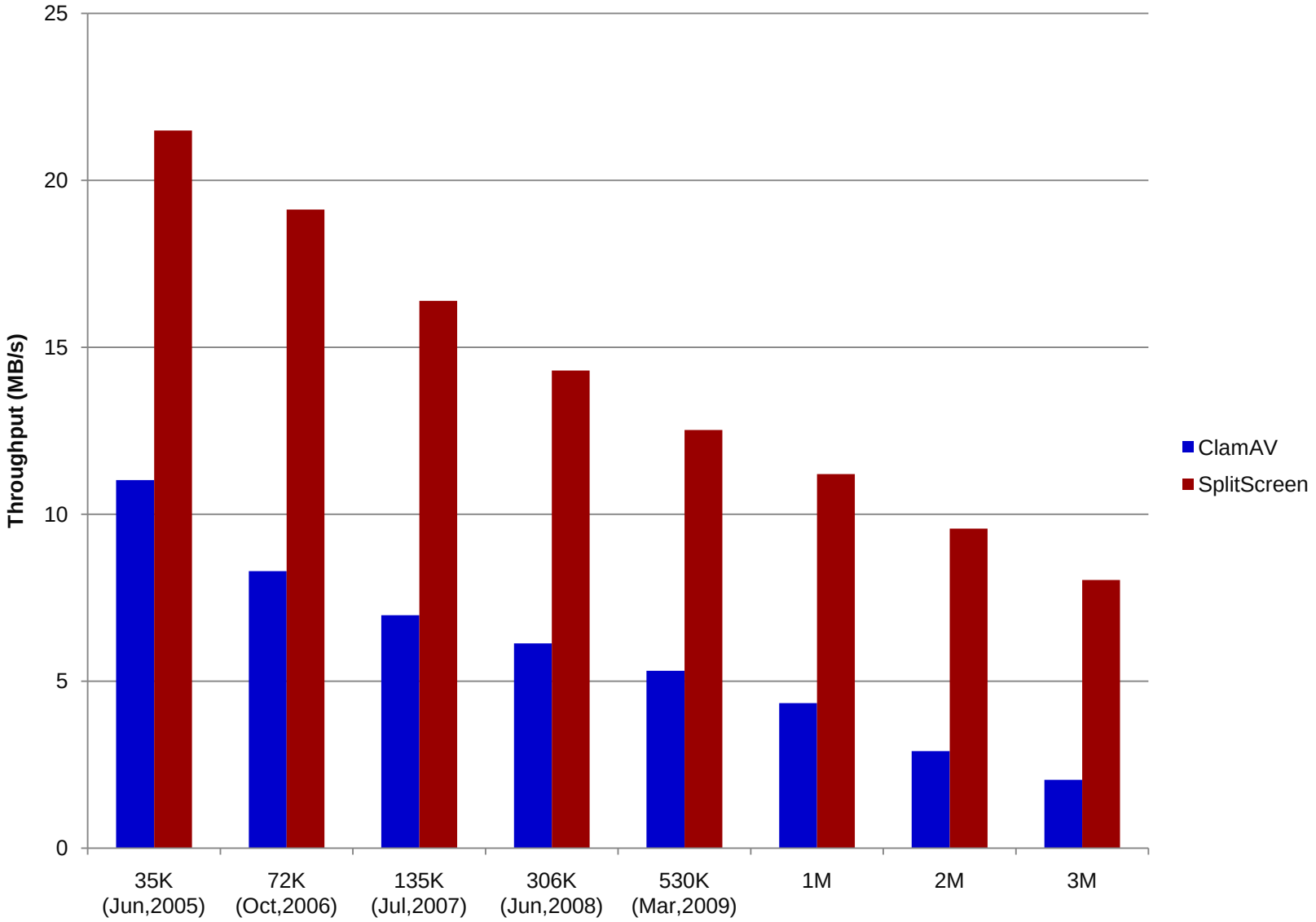

Signature Database
Pre-process
FFBF
Input files
Filter Files
New Algorithms and Hash Functions for memory & cache-friendly screening


Signature
Database
Pre-process
FFBF
List of sigs.
needed
Input
files
Filter Files
Suspect
Files

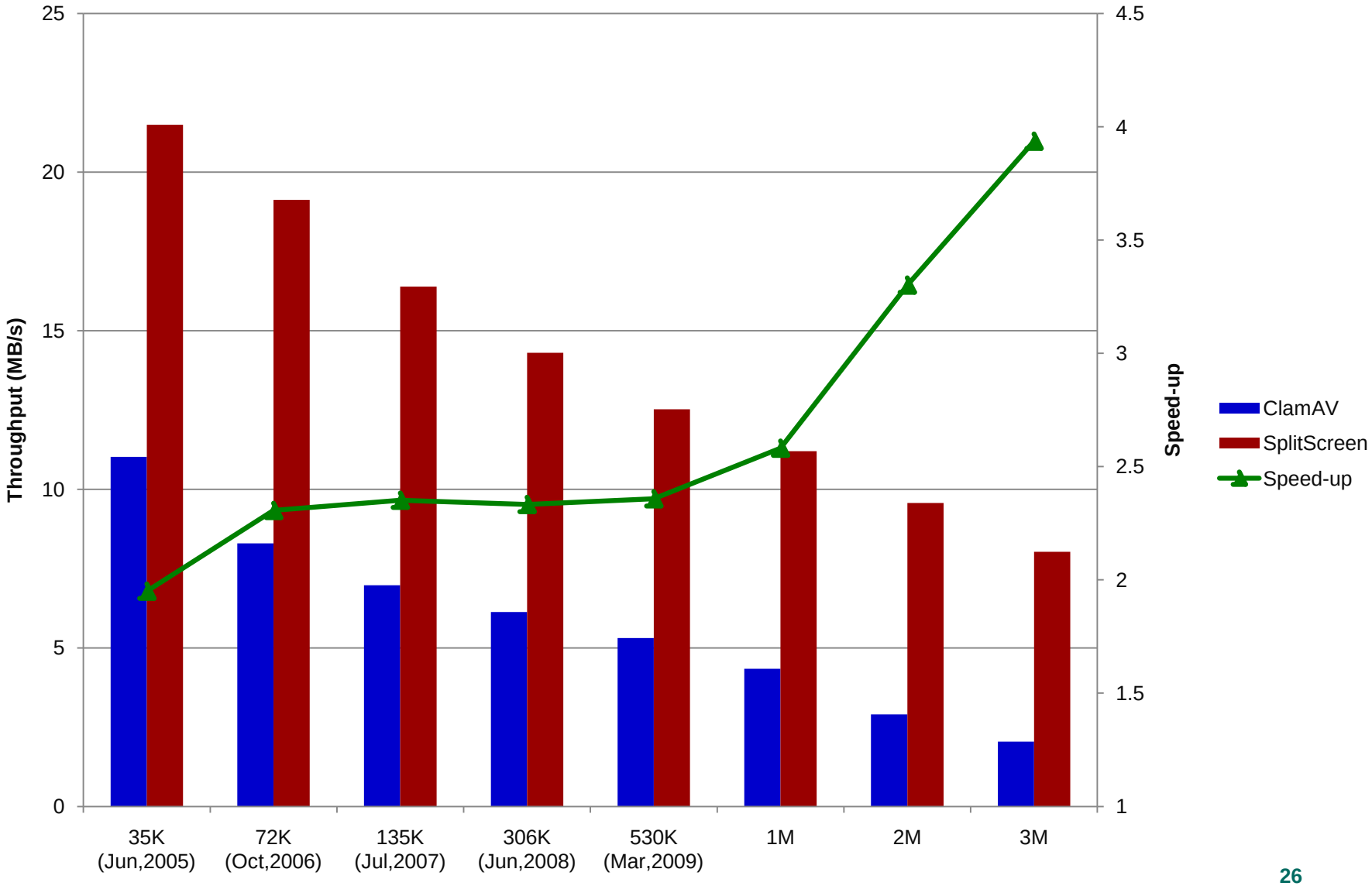

Signature Database
Pre-process
Retrieve Sigs.
FFBF
List of sigs. needed
Suspect sigs.
Input files
Filter Files
Suspect Files


Signature Database
Pre-process
Retrieve Sigs.
FFBF
List of sigs. needed
Suspect sigs.
Input files
Filter Files
Exact matching
Suspect Files

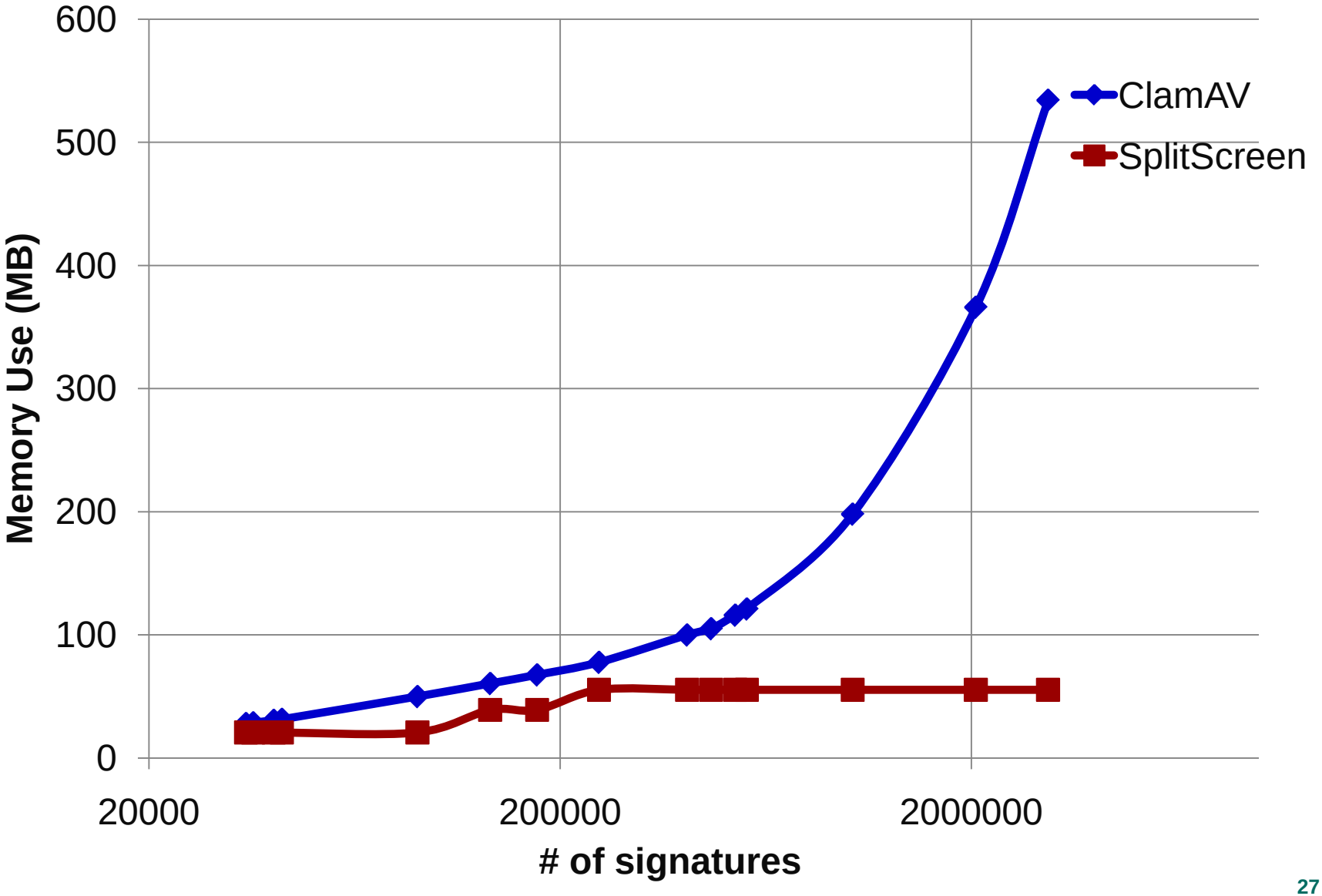
SplitScreen Throughput



SplitScreen Speed-up

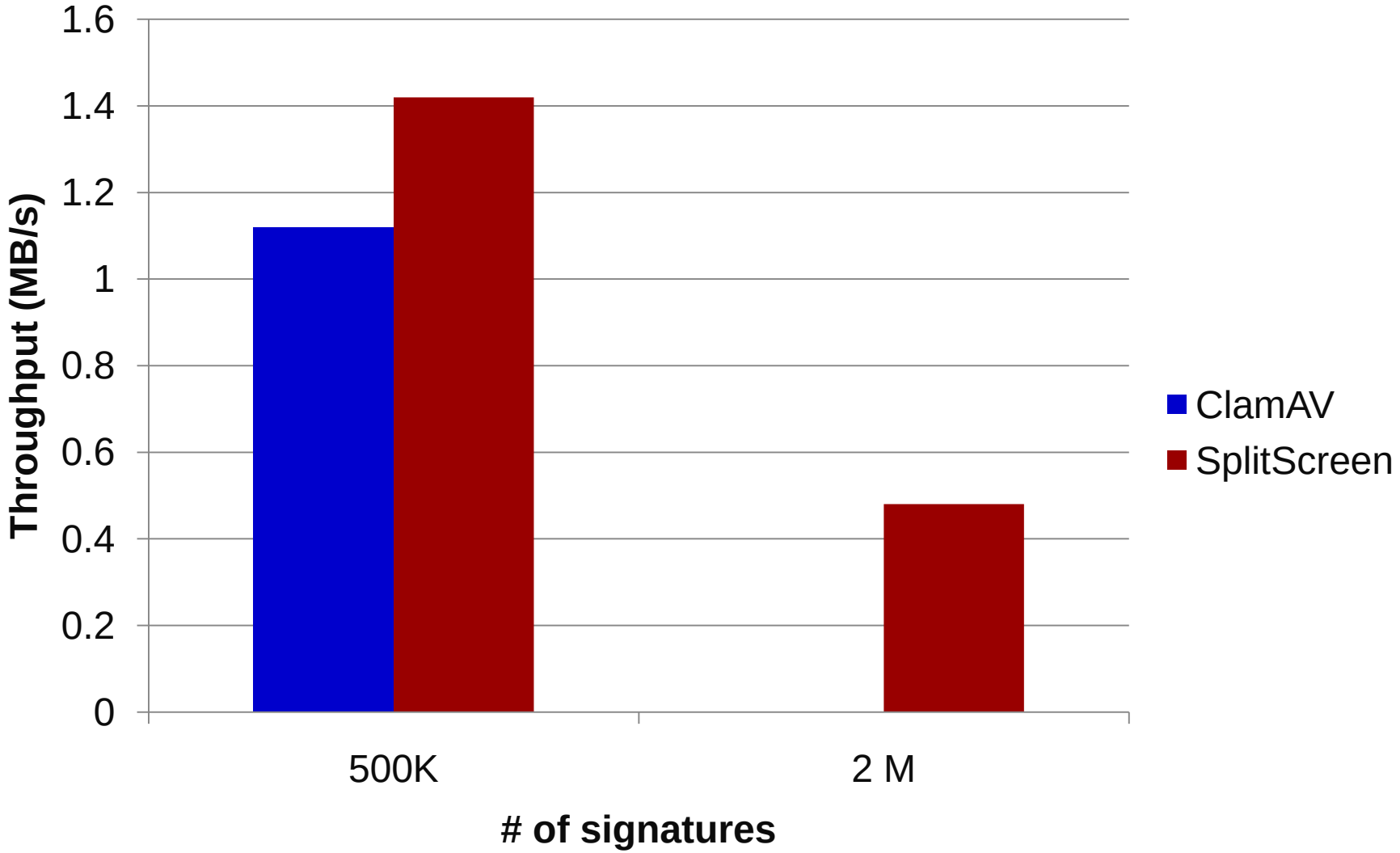


SplitScreen Memory vs. ClamAV

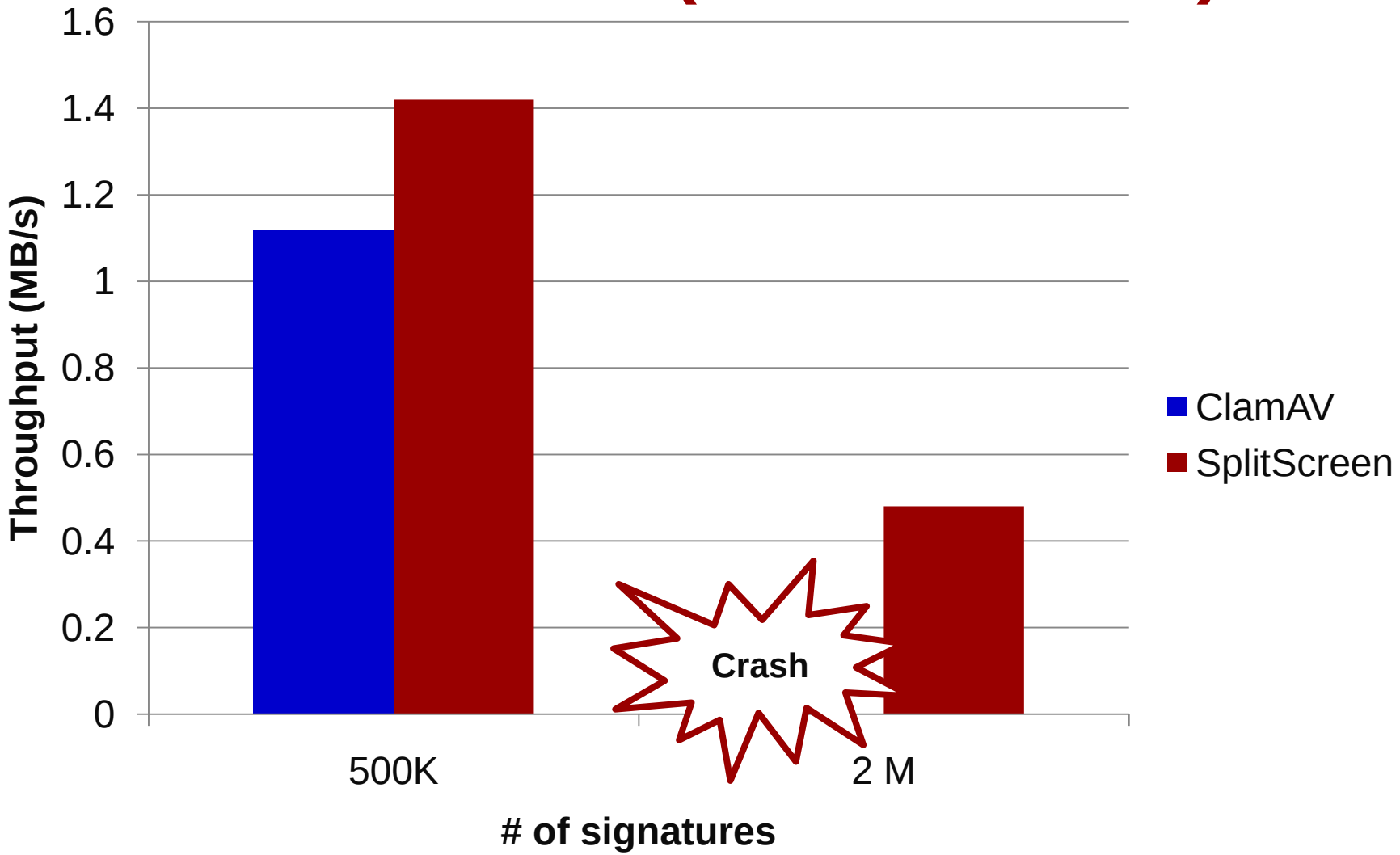


SplitScreen does Anti-Malware on Weaker Devices

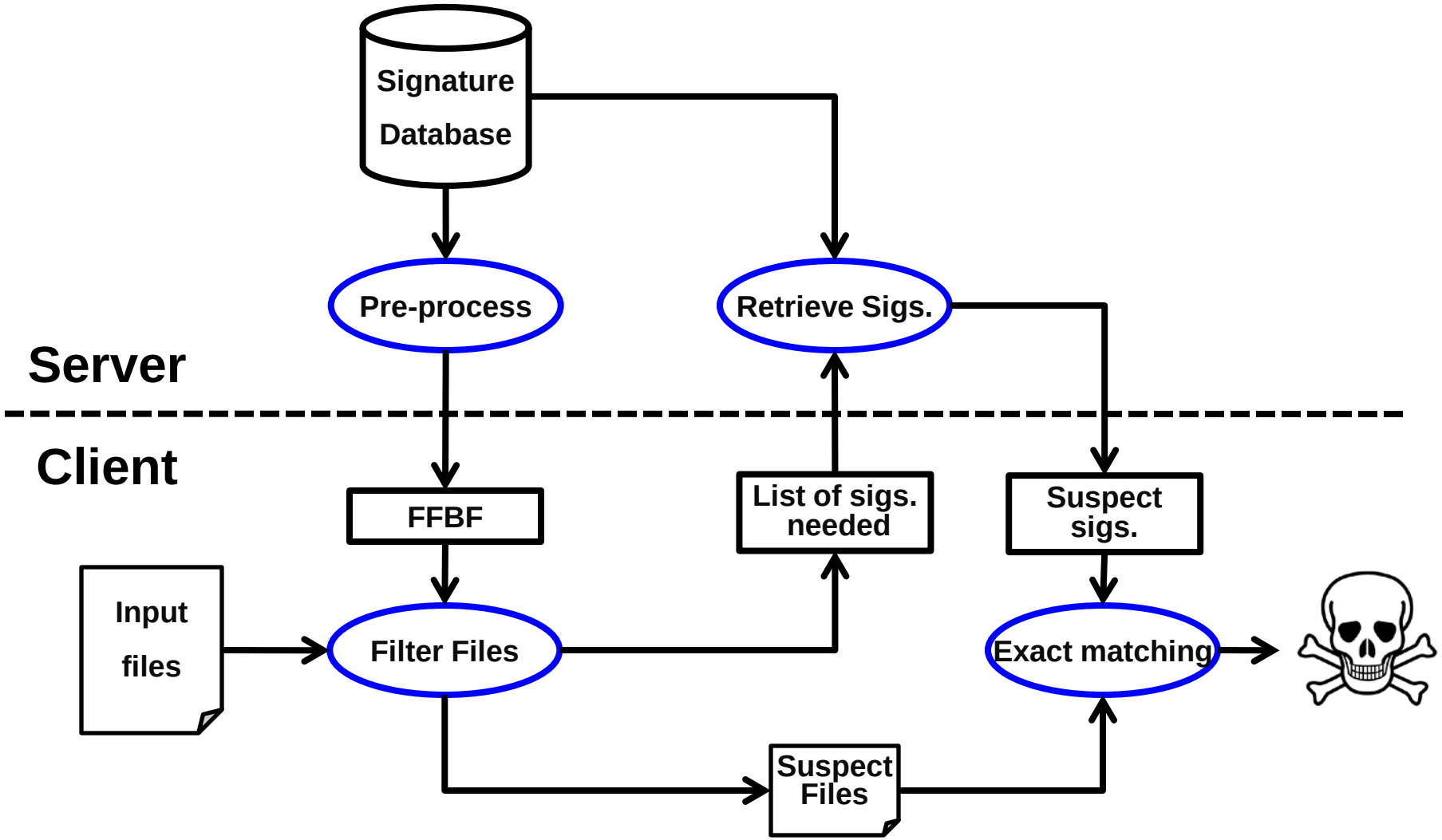
Low-powered AMD Geode (≈ IPhone 3GS)



Low-powered AMD Geode (≈ IPhone 3GS)

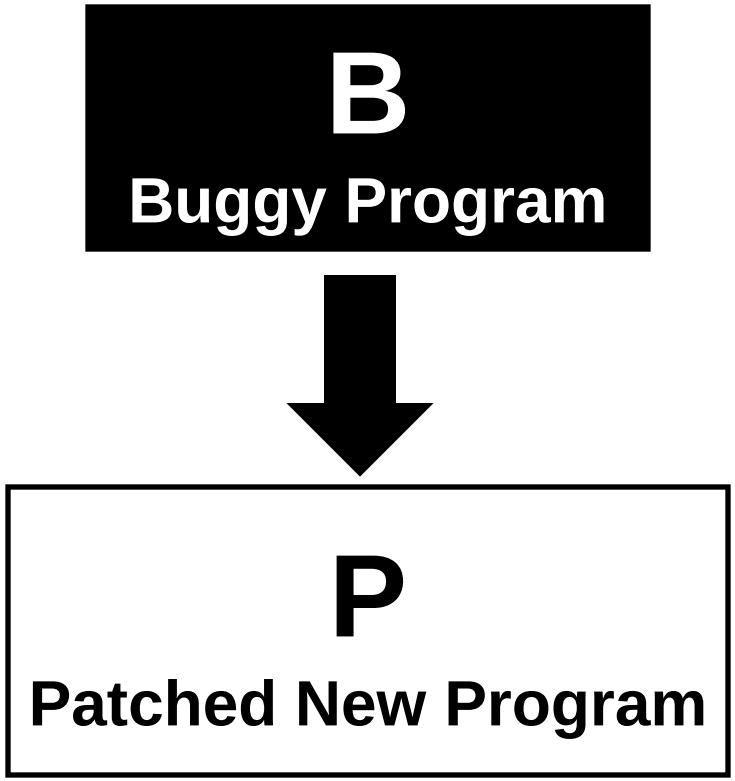


Distributed SplitScreen



- SplitScreen paper at NSDI 2010
- Implementation available to partners

Automatic Patch-Based Exploit Generation (APEG)



Patches Help Security

Patches Can Help Attackers

– *Evil David*

Evil David

Delayed Patch Attack

Evil David

Gets Patch

Use Patch to Reverse Engineer Bug

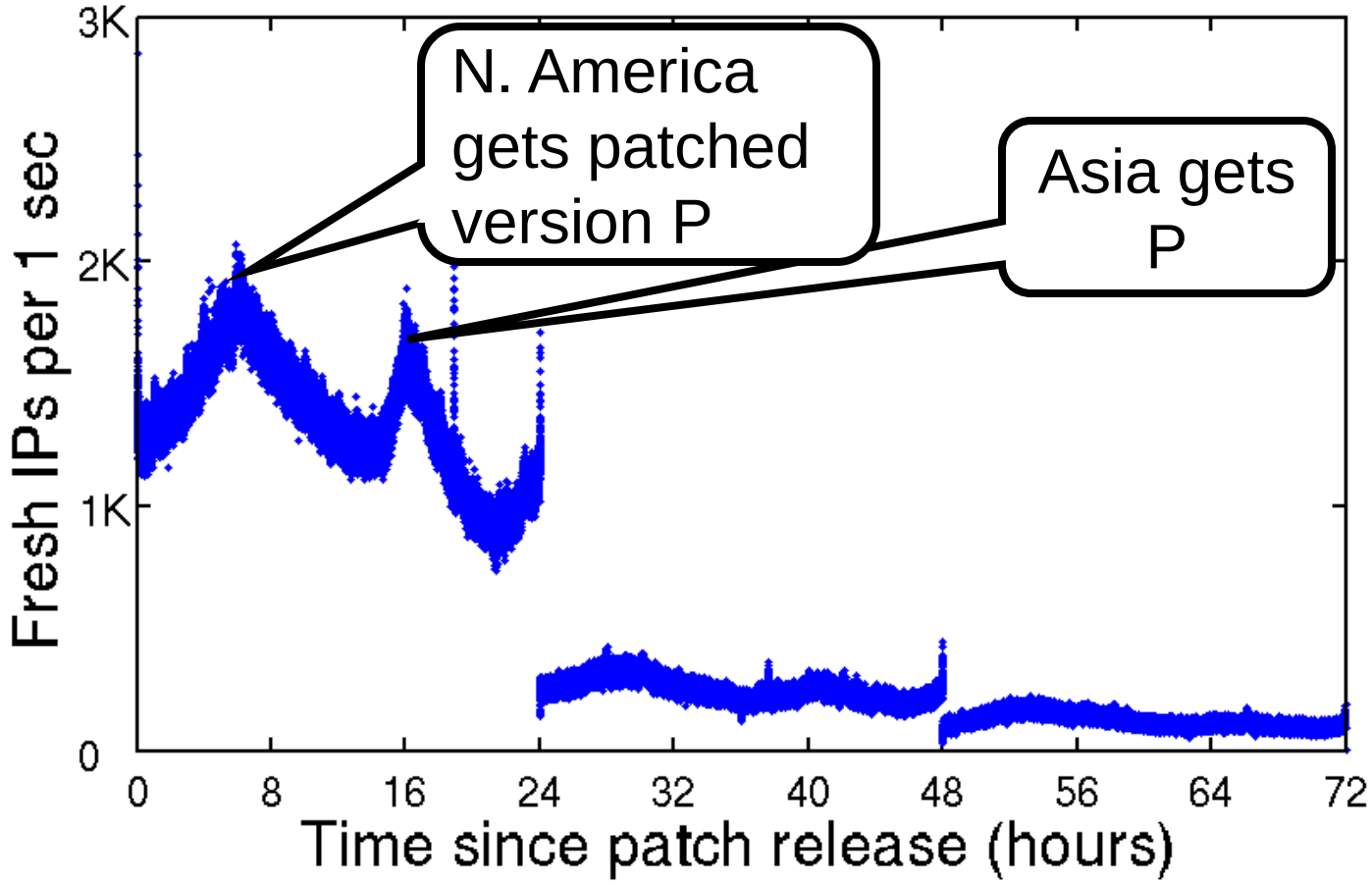
T1

T2

Attack Unpatched Users

Evil David's Timeline

Patch Delay



[Gkantsidis et al 06]

I can reverse engineer the patched bug and create an ***exploit*** in ***minutes***

Minutes

Gets Patch

Reverse Engineer Bug & Create Exploit

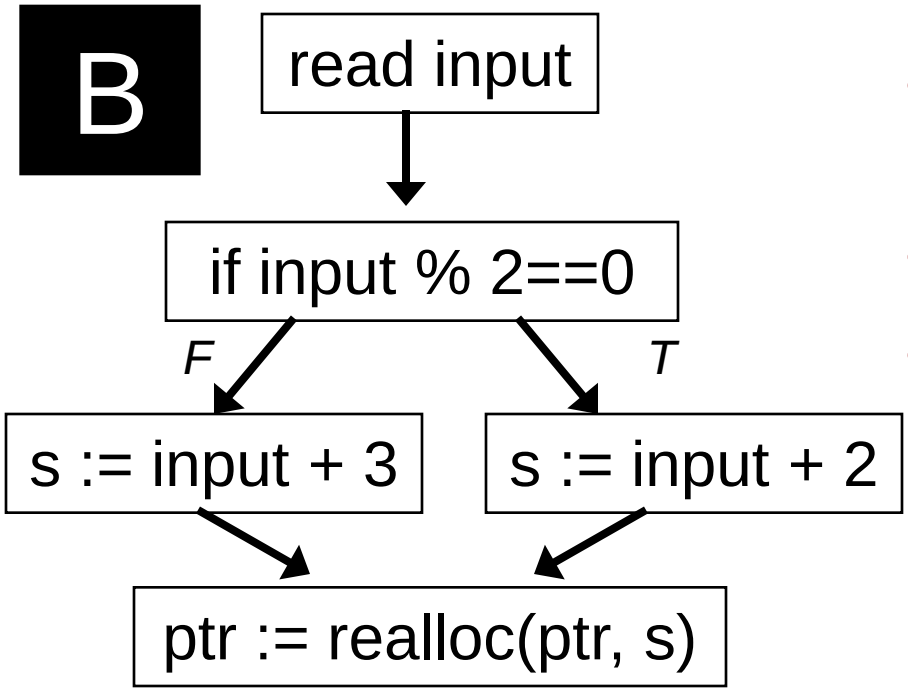
T1

T2

Attack Unpatched Users

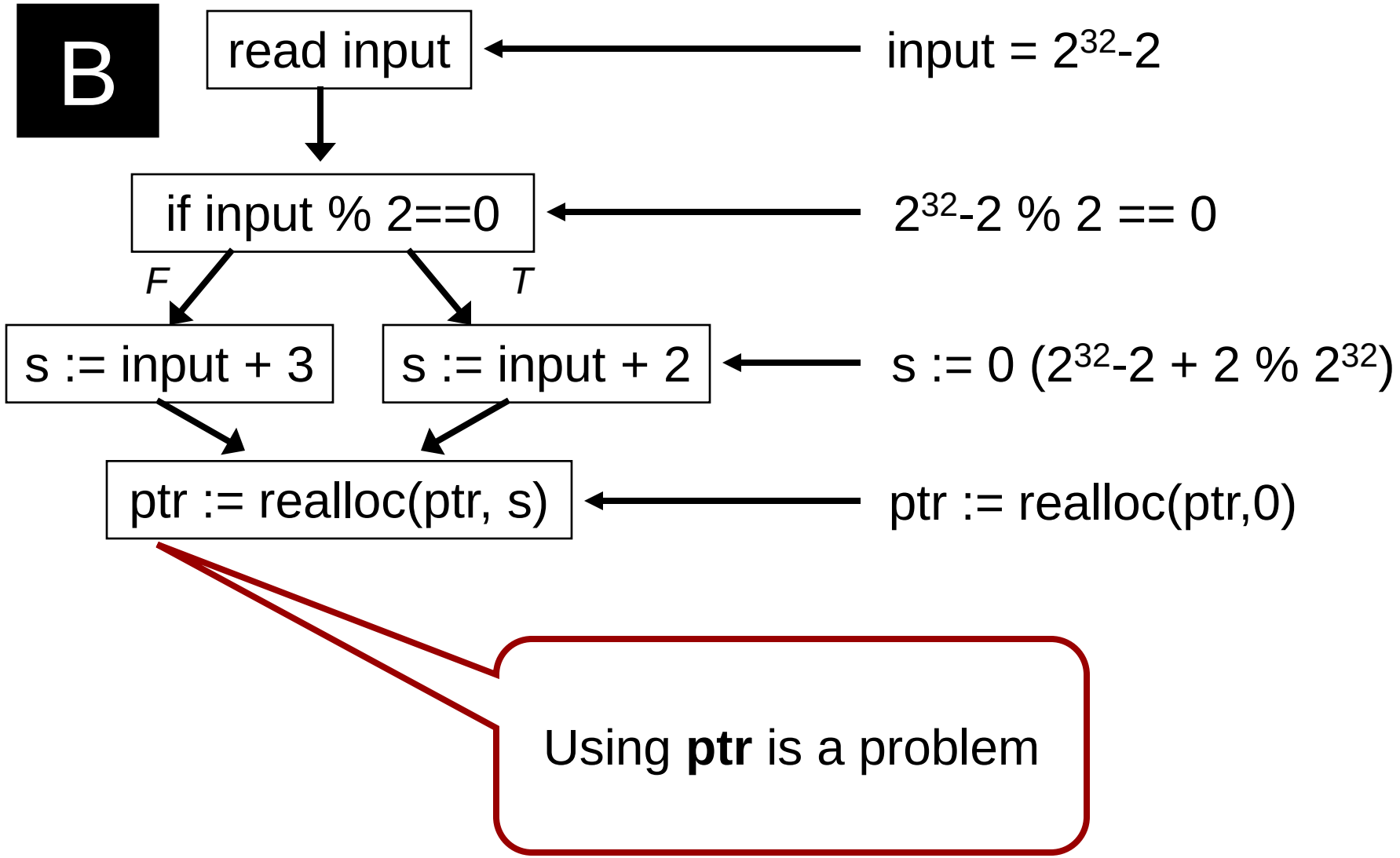
Evil David's Timeline

Example

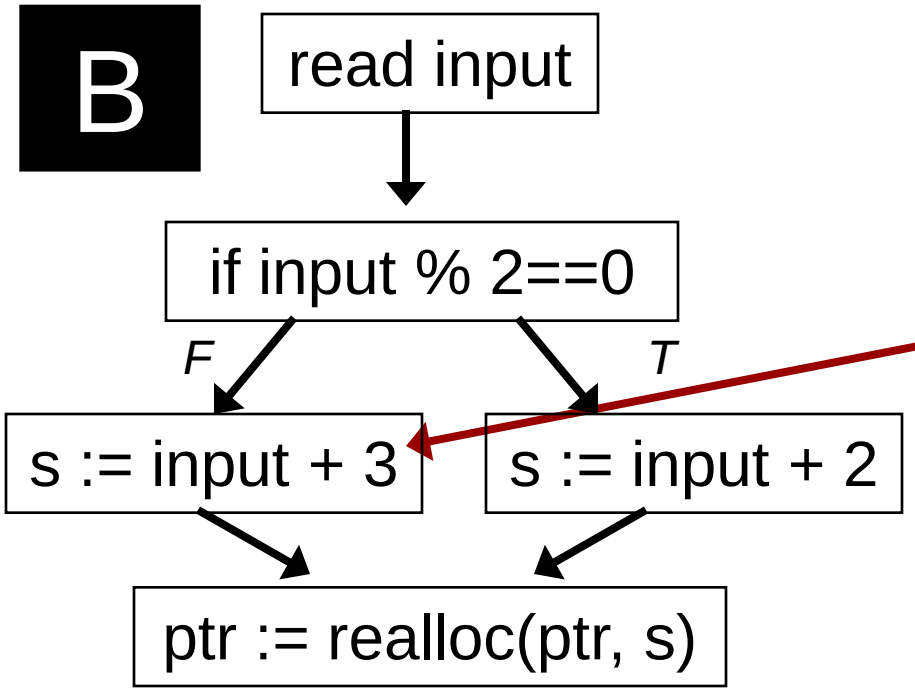


- All integers unsigned 32-bits
- All arithmetic mod 2^{32}
- B is binary code

Example


B
read input
input = 2^{32}-2
if input % 2==0
2^{32}-2 % 2 == 0
F
T
s := input + 3
s := input + 2
s := 0 (2^{32}-2 + 2 % 2^{32})
ptr := realloc(ptr, s)
ptr := realloc(ptr,0)
Using **ptr** is a problem

Example

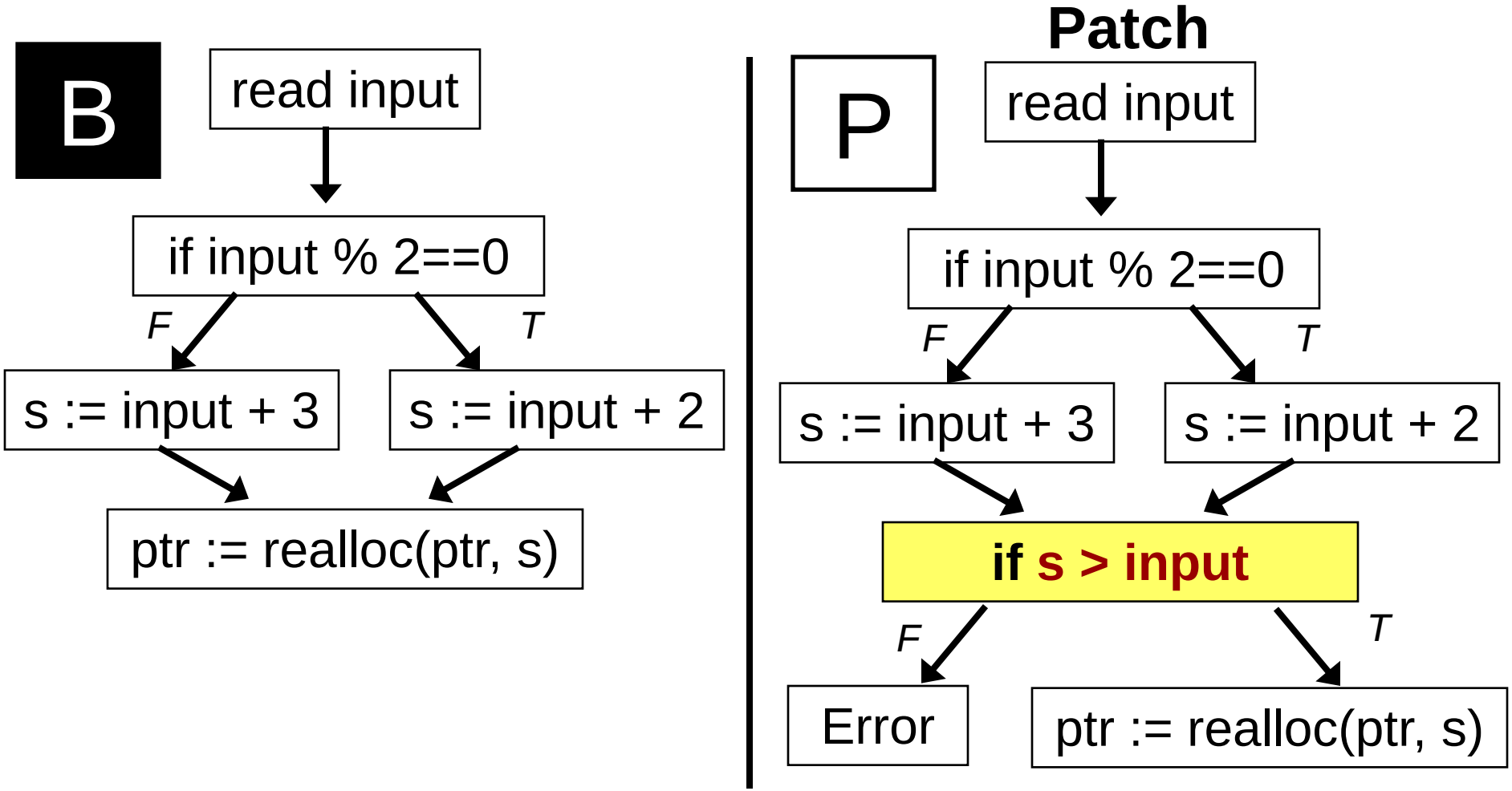


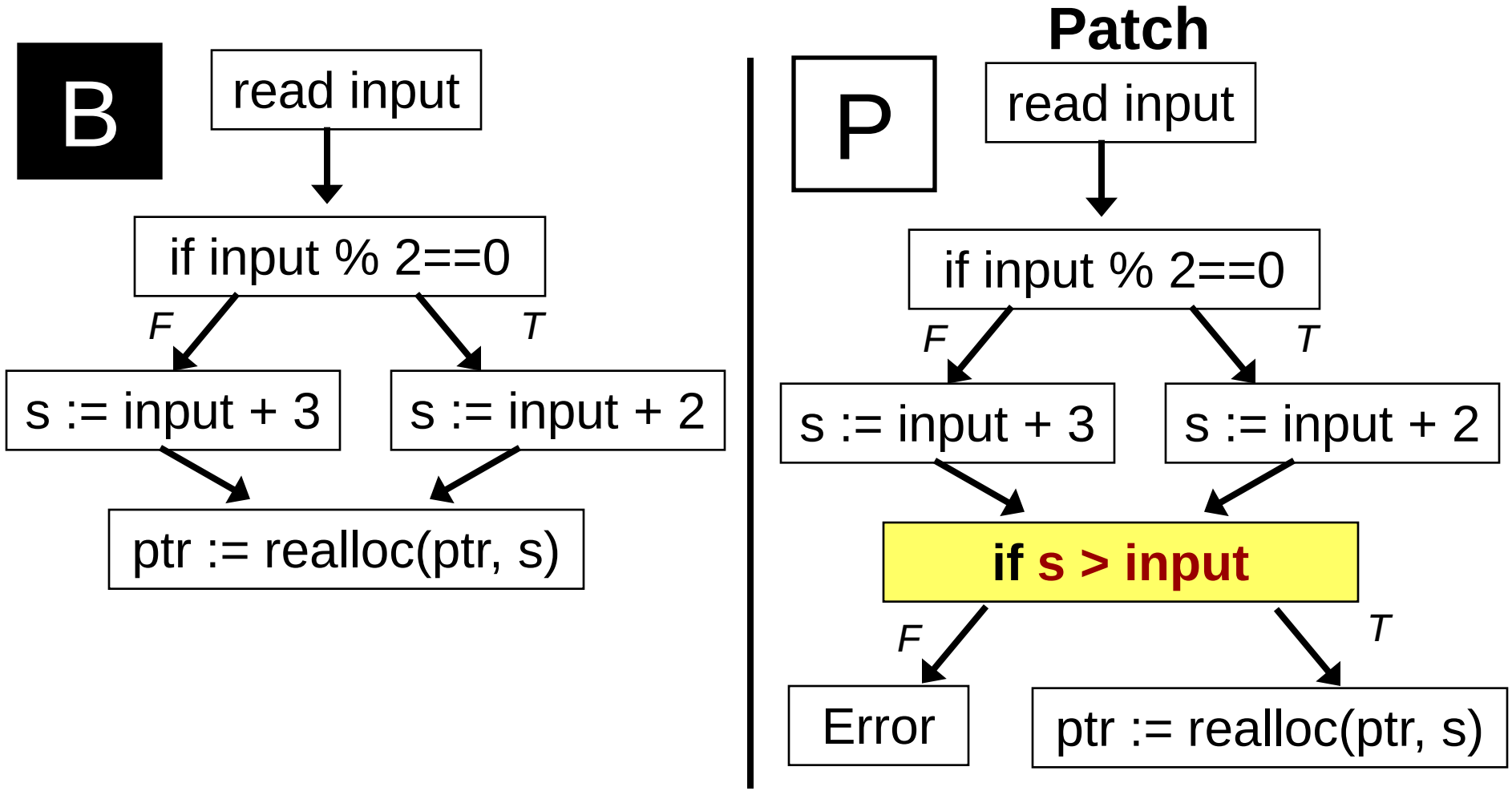
Wanted:

s > input

Integer Overflow when:

¬(s > input)

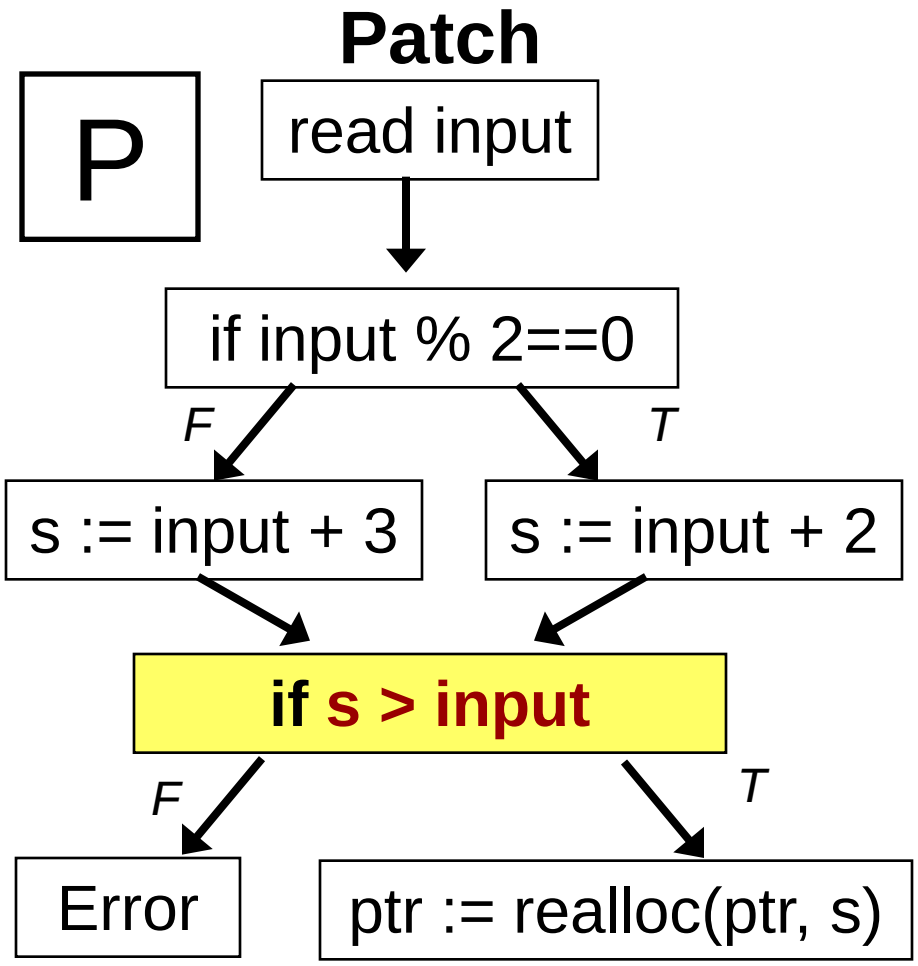

B
read input
if input % 2==0
F
T
s := input + 3
s := input + 2
ptr := realloc(ptr, s)
Patch
P
read input
if input % 2==0
F
T
s := input + 3
s := input + 2
if s > input
F
T
Error
ptr := realloc(ptr, s)



Exploits for B are inputs that fail
new safety condition check in P
(s > input) = false

Automated Patch-Based Exploit Generation

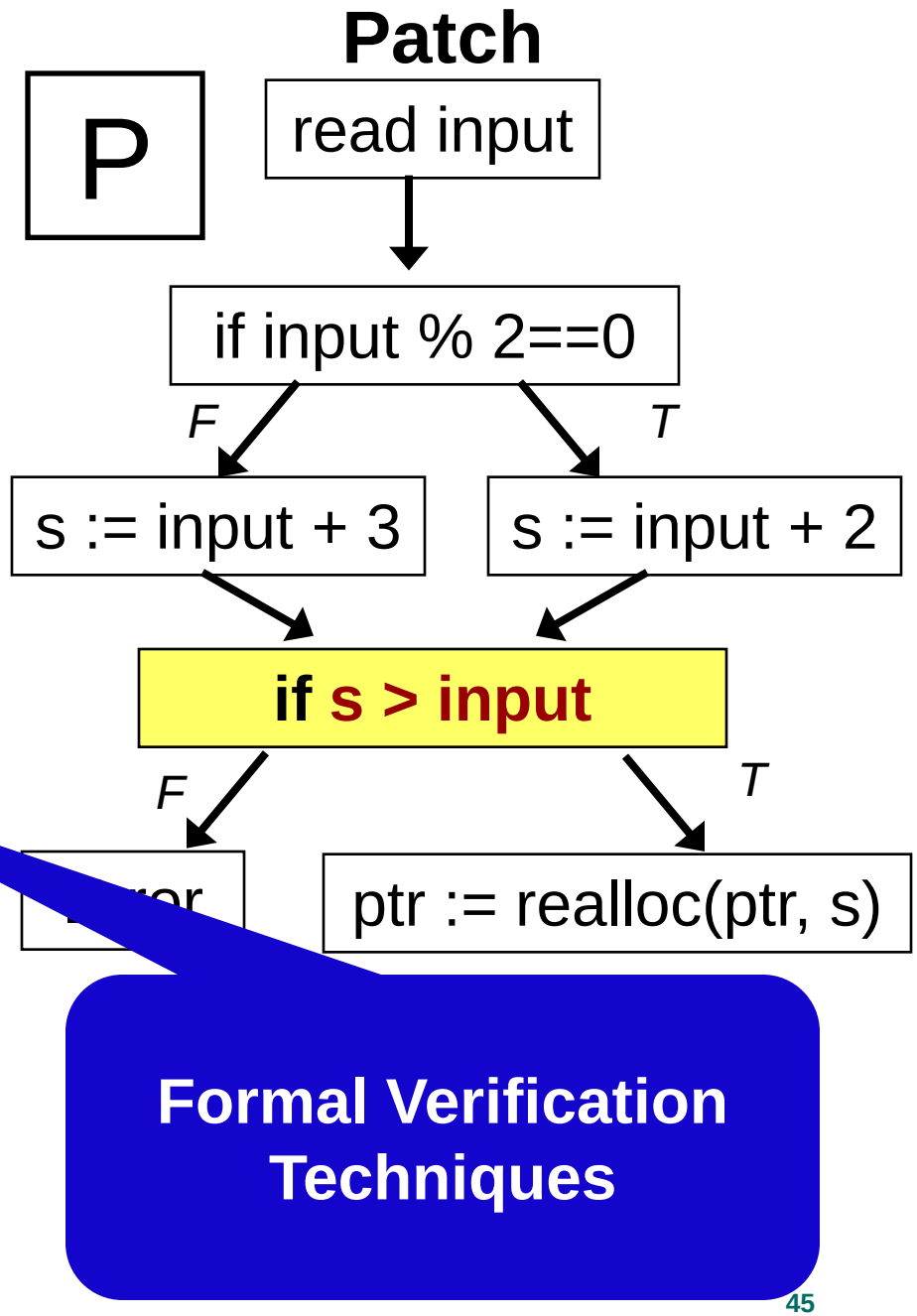
1. **Diff B and P to identify location of new safety check**
2. **Create input that fails safety condition in P using Vine**
3. **Verify input is exploit on original buggy program B**



Off the shelf tools

Exploit Generation

1. **Diff B and P to identify location of new safety check**
2. **Create input that fails safety condition in P**
3. **Verify input is exploit on original buggy program B**

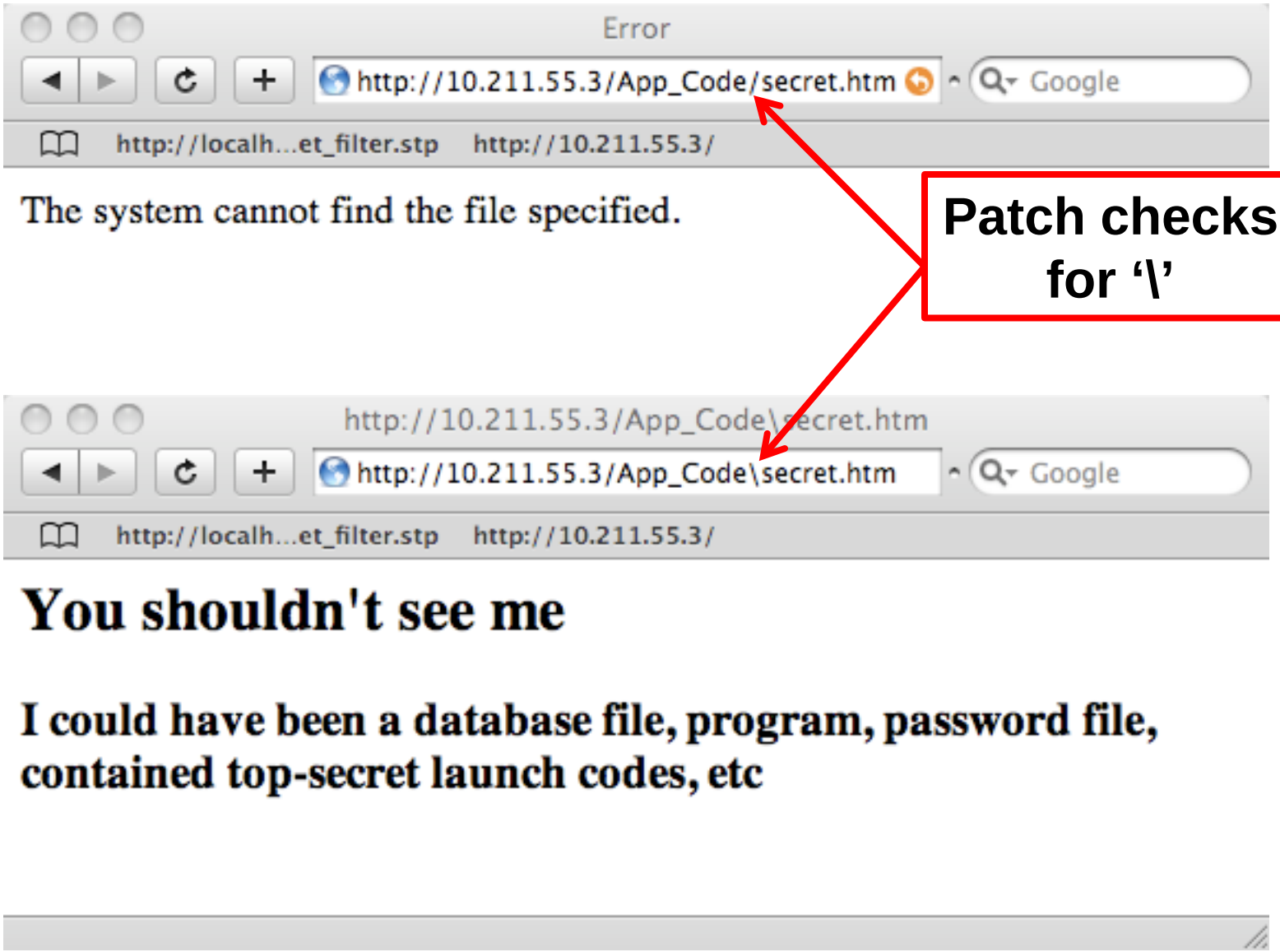


Exploiting Microsoft Patches

ASPNet_Filter	Information Disclosure	29 sec
GDI	Hijack Control	135 sec
PNG	Hijack Control	131 sec
IE COMCTL32 (B)	Hijack Control	456 sec
IGMP	Denial of Service	186 sec

- No public exploit for 3 out of 5
- Exploit unique for other 2

Windows ISS .Net Example



B

Does Automatic Patch-Based Exploit Generation Always Work?

NO!

However, in security attackers get lucky, defenders do not

1. **Current Delayed Patch Distribution Insecure**
2. **Automatically generating exploits is possible**

Current Research Thrusts

My Research Philosophy:
Develop fundamental advances in science demonstrated in compelling scenarios

Significant Research Thrust:
Fully Automated Exploit Generation

Given program, find bugs, generate exploits

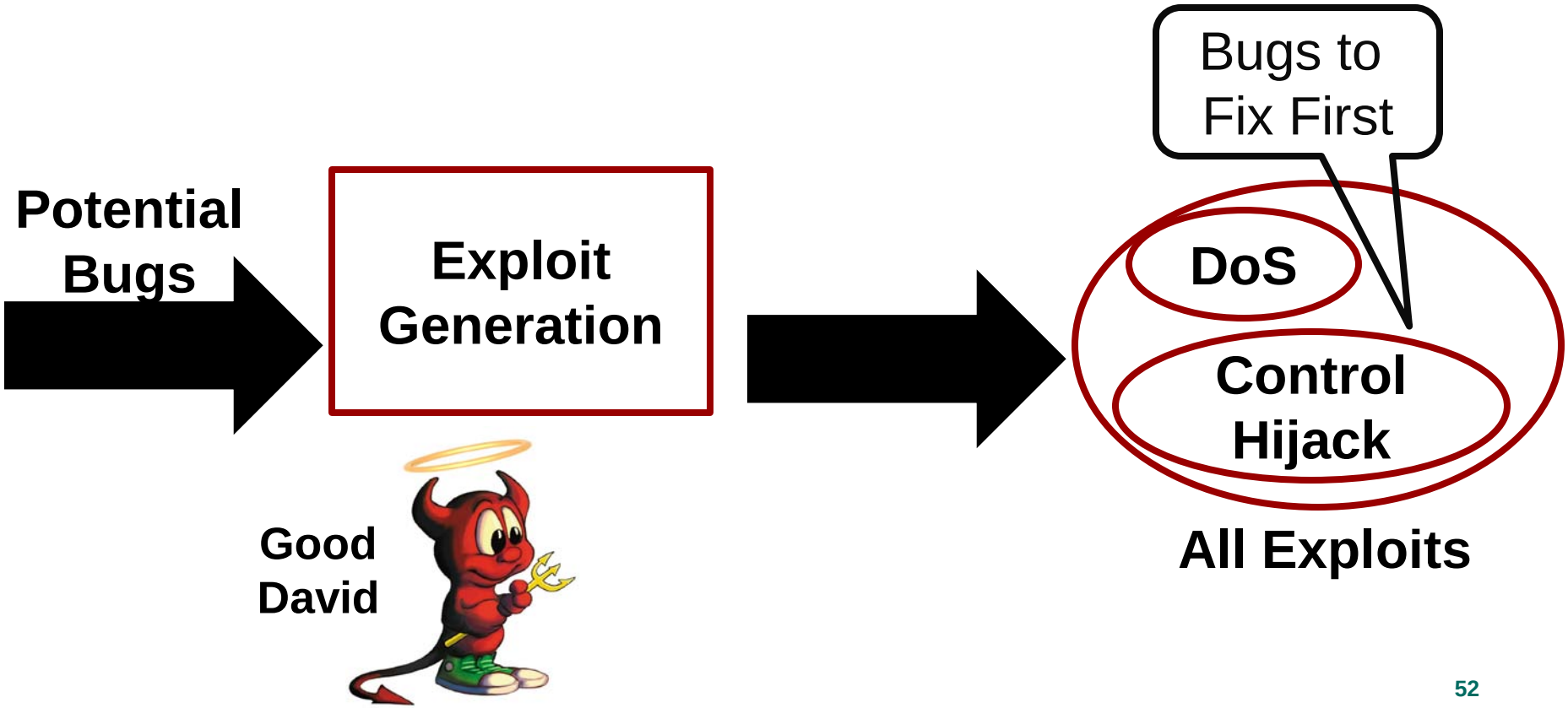
Isn’t that for evil?

Yes. That is why it is awesome

Evil David

For Good: Prioritizing Bugs

Ubuntu has over 53,000 bugs to fix.
Which one should be fixed first?



Exploit Generation: Scientific Contributions

- Tons of applications
  - Cyber-warfare, Bug Prioritization
  - Checking Patches
  - Automatically generating filters for intrusion detection systems
- Understand capabilities of attackers
  - We are often good at provably good defenses once we know what to defend against

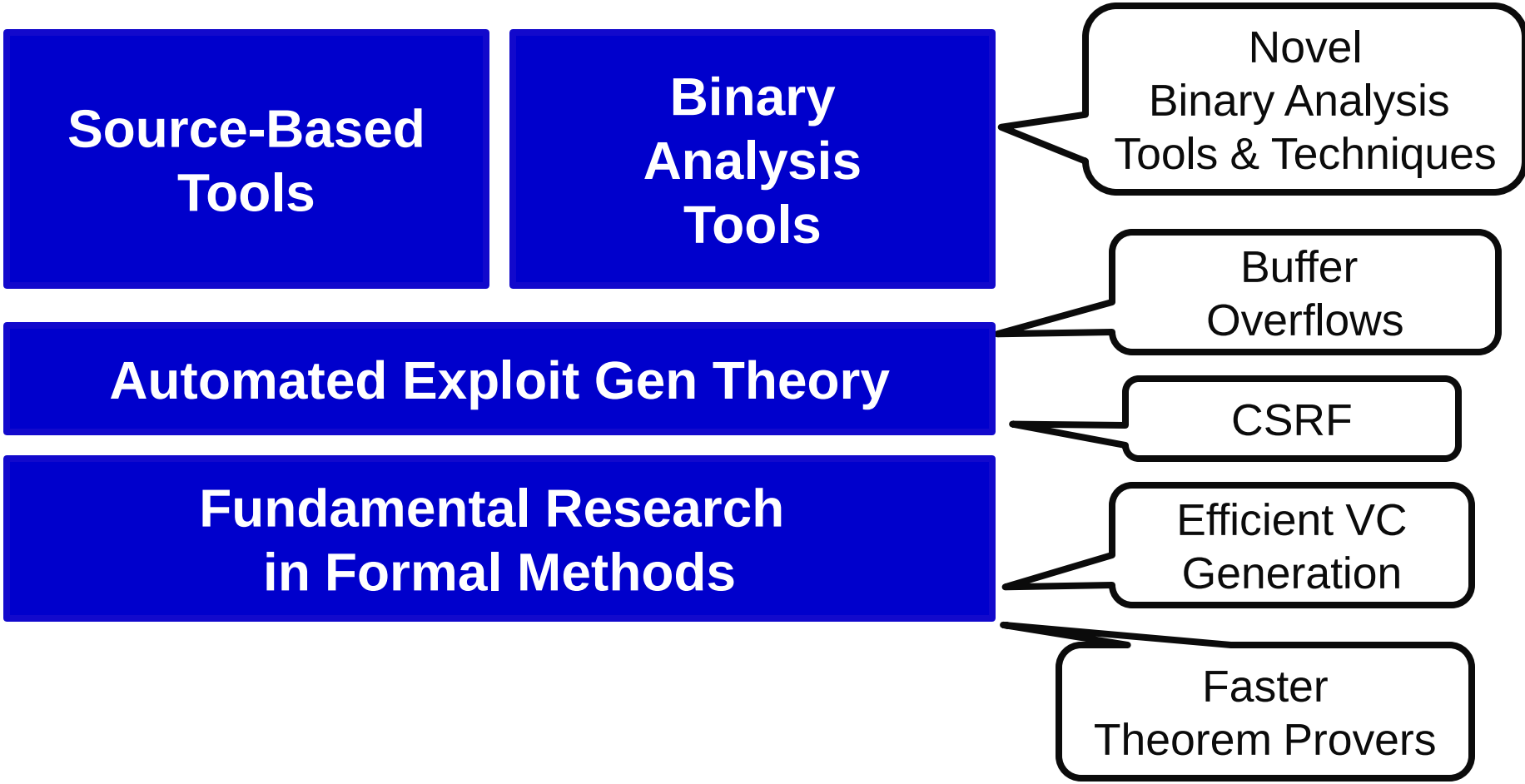
Compelling scenario for researching age-old questions about program behavior, correctness, protecting algorithms, and so on.

Demo

```
int test(char* str) {
    char buf3[10];
        int i;
    *buf3 = 1;
    strcpy(buf3, str);
    return 0;
}

int main()
{
    char buf[500];
 return test(buf);
}
```

Automated Exploit Generation Research Project



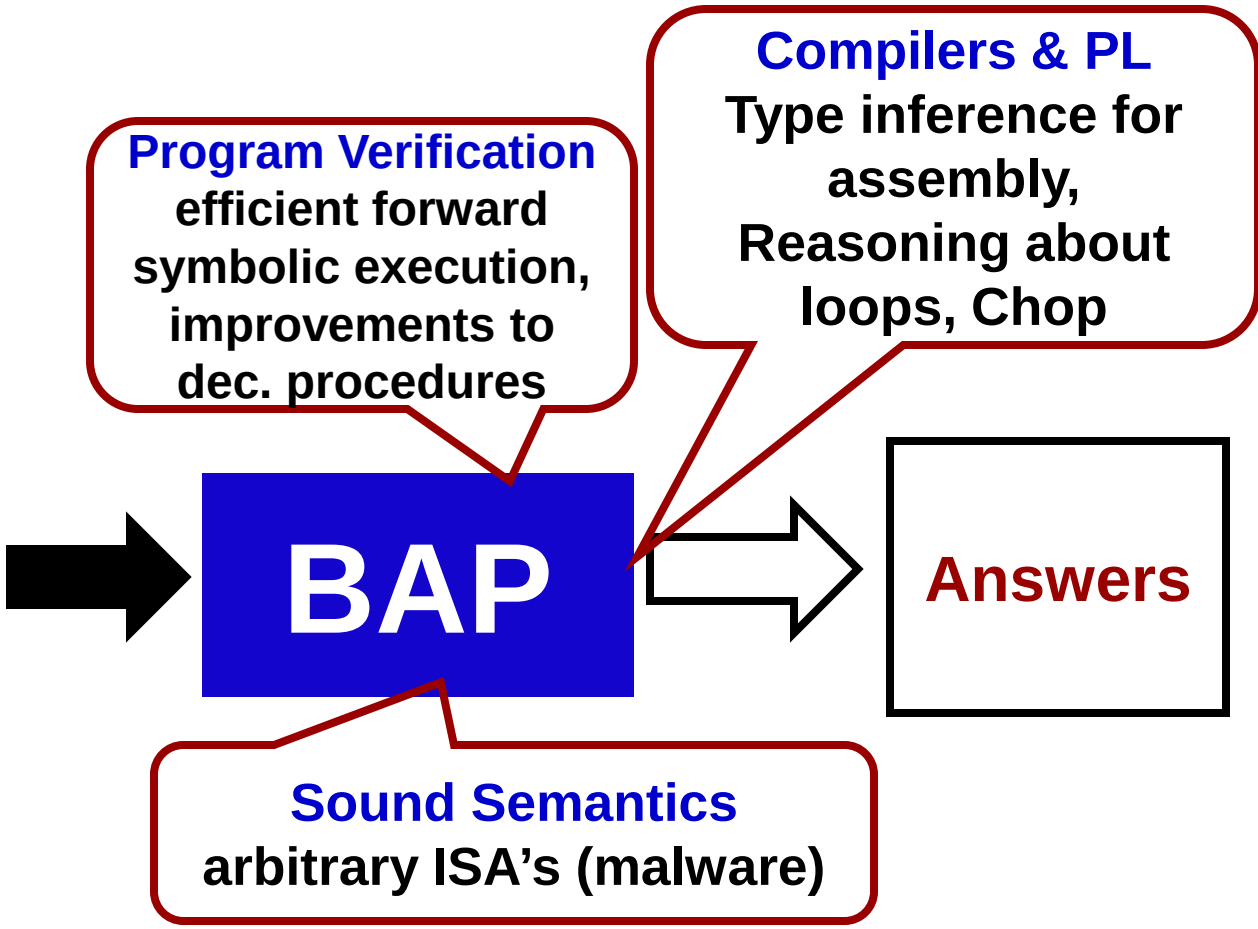
BAP Project:
Next-Generation Binary Analysis Platform

Reason about binary code as well as source

Enabling Technology

Specific Projects:

- Bug Prioritization
- APEG
- Signature Gen
- Anti-Malware
- Sound Reverse Engineering
- ....



http://bap.ece.cmu.edu

Summary

1. SplitScreen
   - Malware Scanning at 2x the speed, ½ the memory
   - Scanning on emerging platforms like cellphones, iPads, etc.

2. Automated Patch-based Exploit Generation
   - Patches help attackers

3. Research Thrusts
   - Significant Thrust: Automate Exploit Generation
   - Next-Generation Binary Analysis

Questions?

Thank you for your attention.

dbrumley@cmu.edu
http://security.ece.cmu.edu/
http://bap.ece.cmu.edu