Model-Based Whitebox Fuzzing for Program Binaries

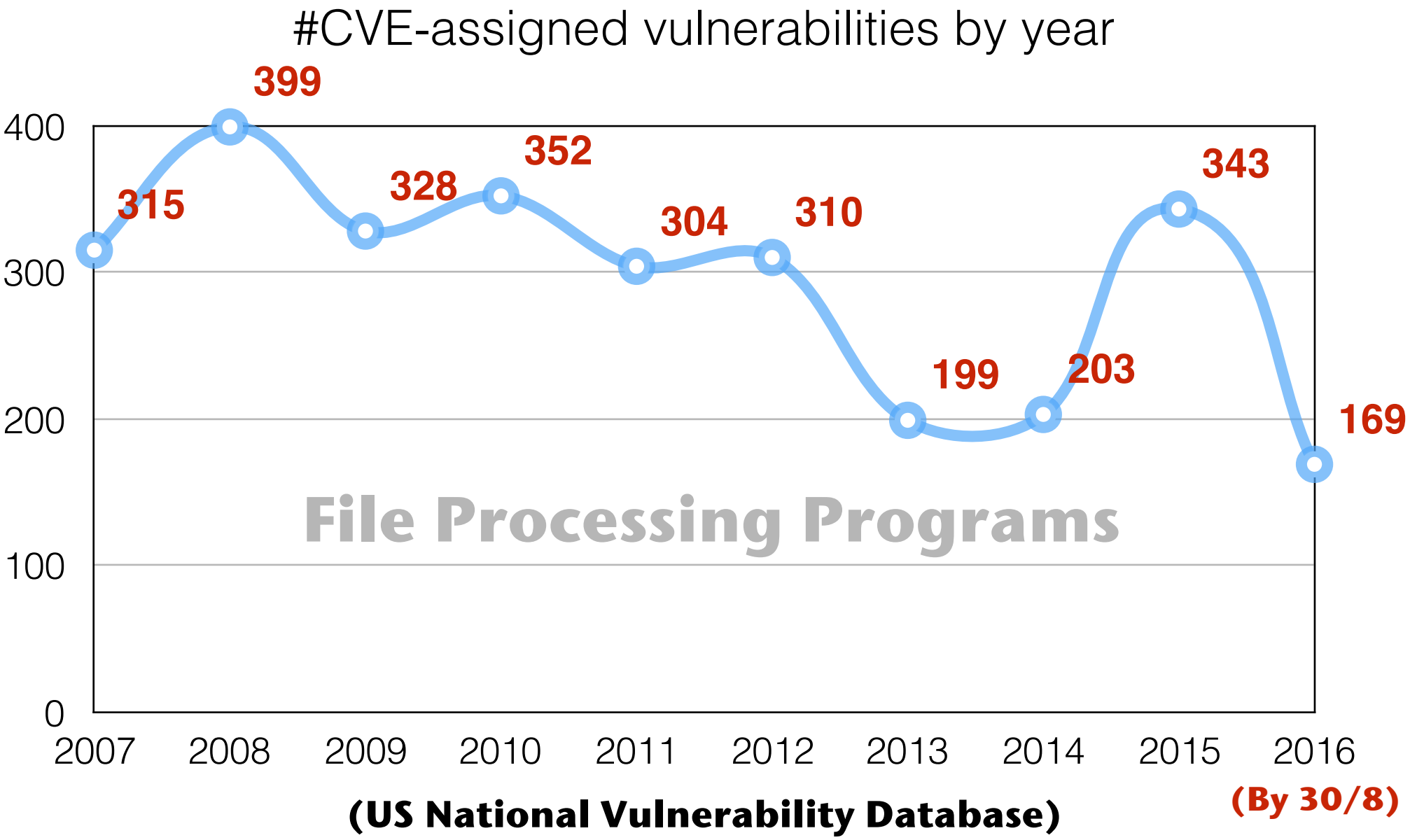
Thuan Pham

Marcel Böhme

Abhik Roychoudhury

ASE 2016
September 3-7, 2016
Singapore

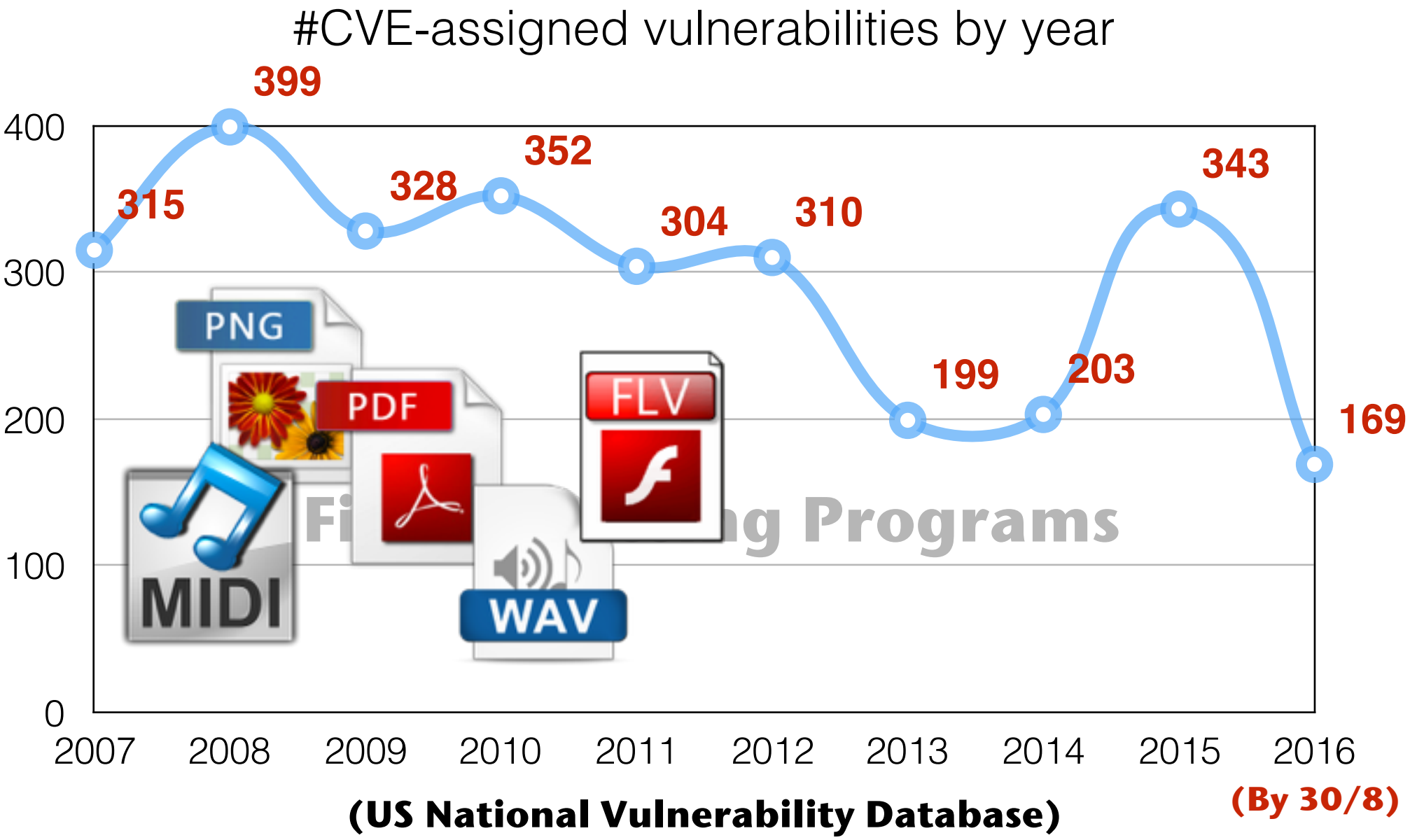
Vulnerabilities in file-processing programs



(US National Vulnerability Database)

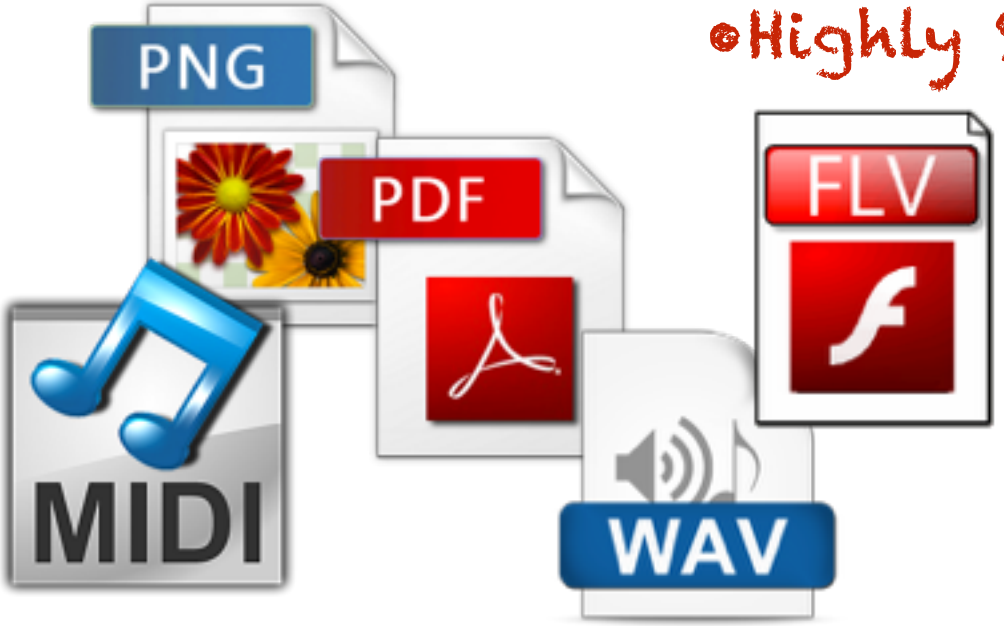
(By 30/8)

Vulnerabilities in file-processing programs



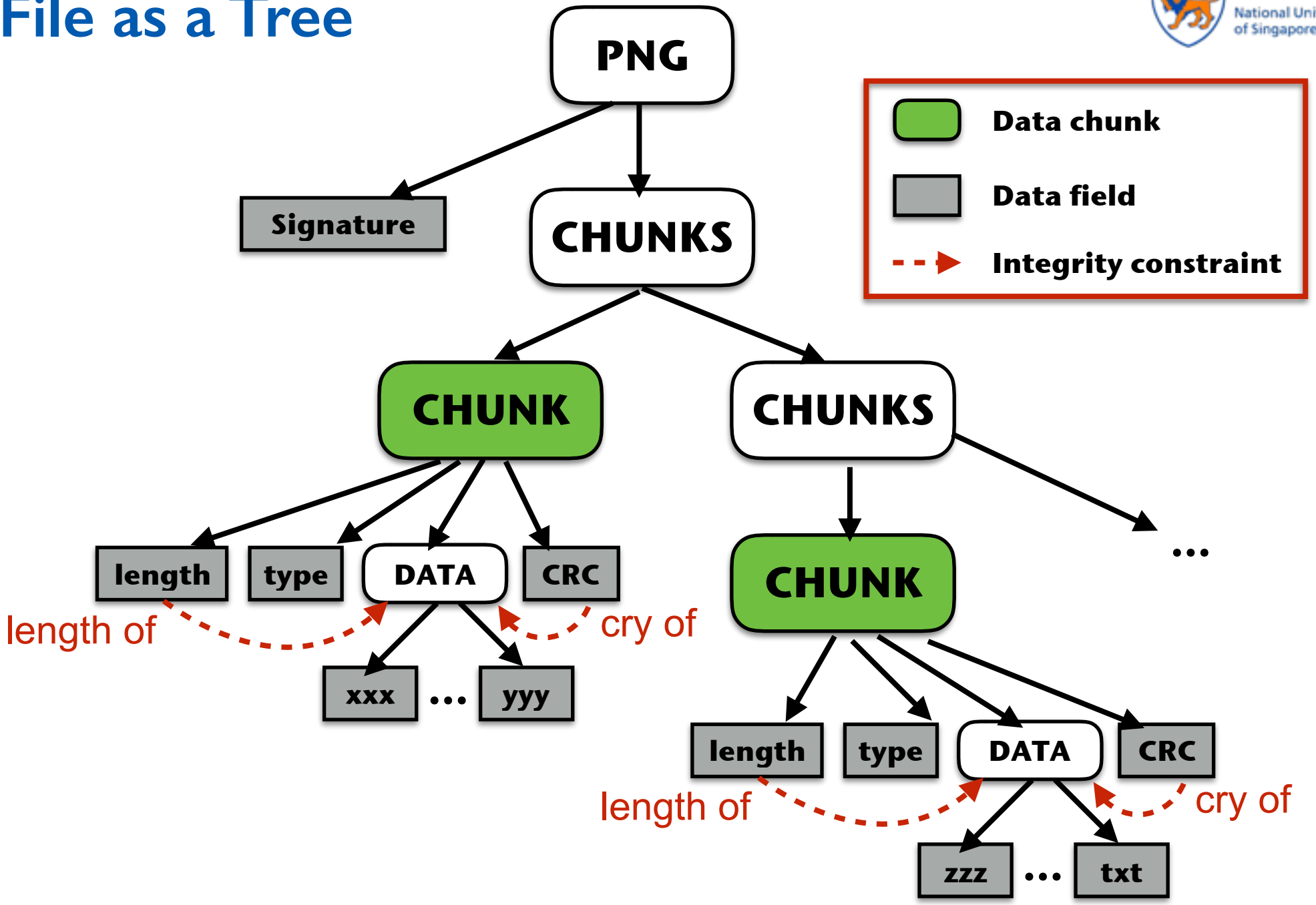
Challenge

- Generating test cases to expose vulnerabilities in file-processing software is challenging !

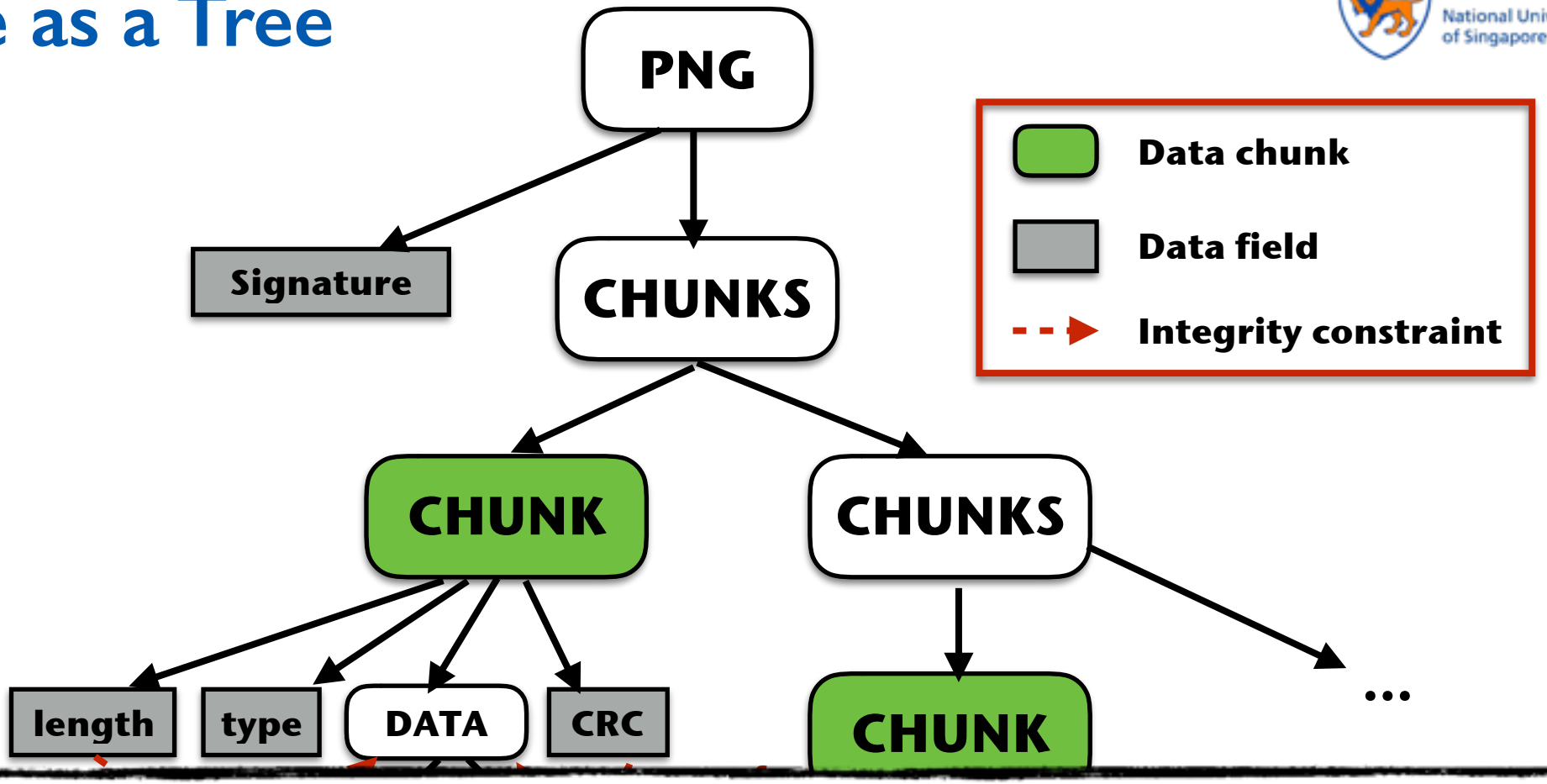


- Highly Structured
- Having both syntactic and semantic relationships
- Integrity constraints e.g., Checksums
- Compression/decompression algorithms

File as a Tree



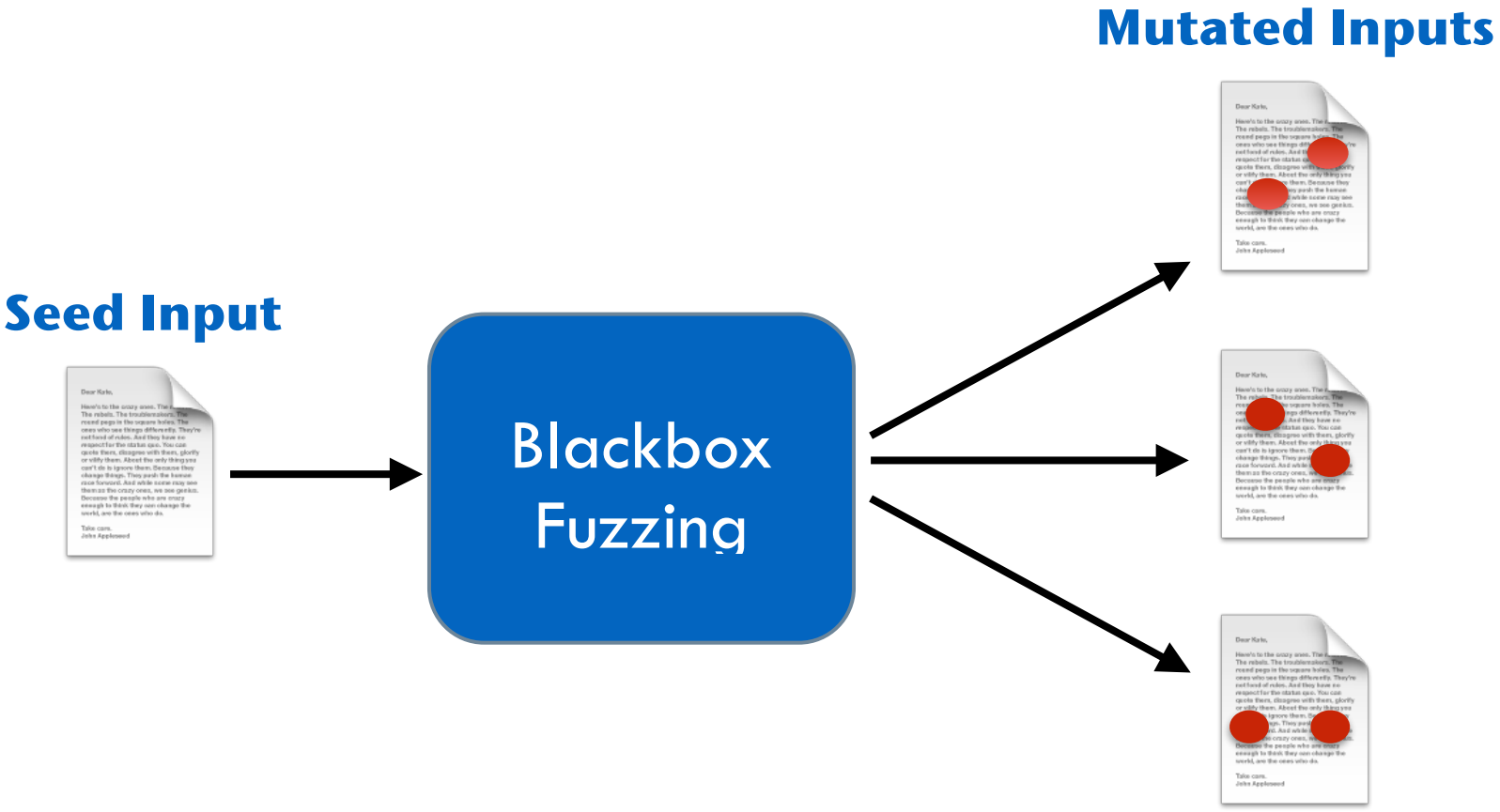
File as a Tree



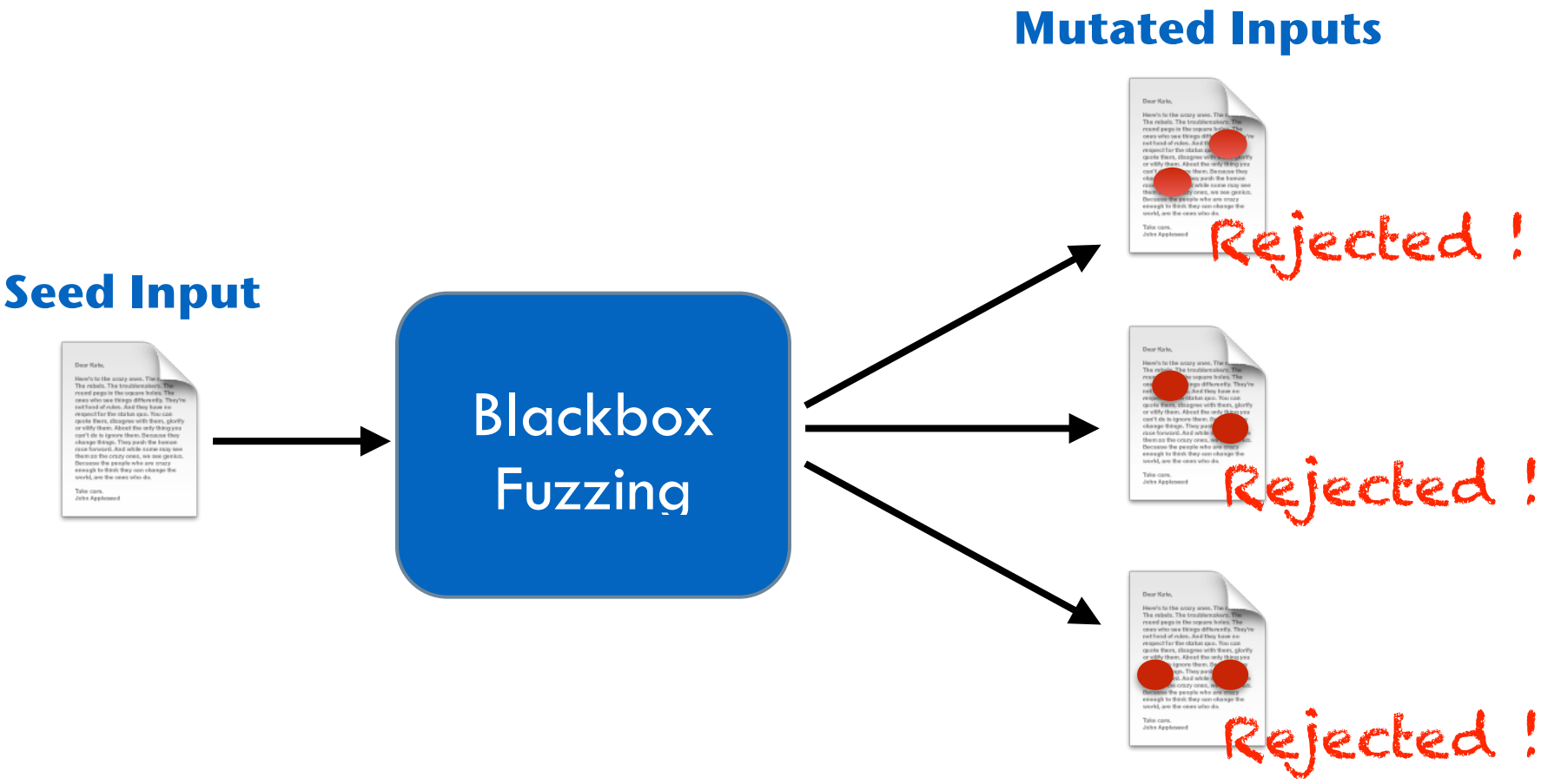
1. (Model-Based) Blackbox Fuzzing
2. Whitebox Fuzzing

zzz ... txt

Blackbox Fuzzing



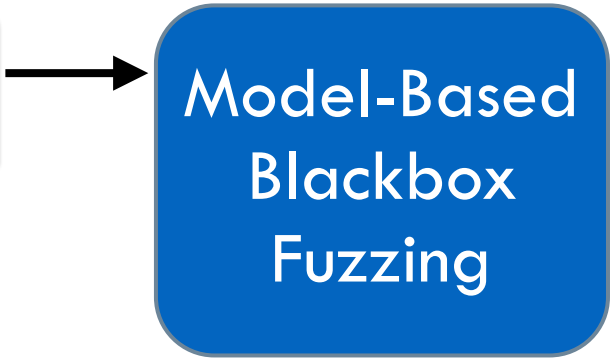
Blackbox Fuzzing



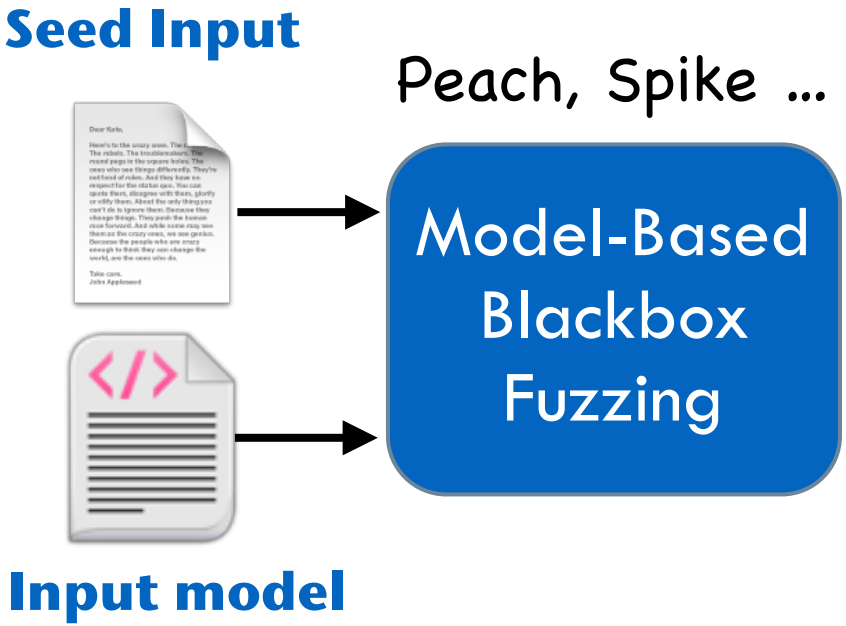
Model-Based Blackbox Fuzzing

Seed Input

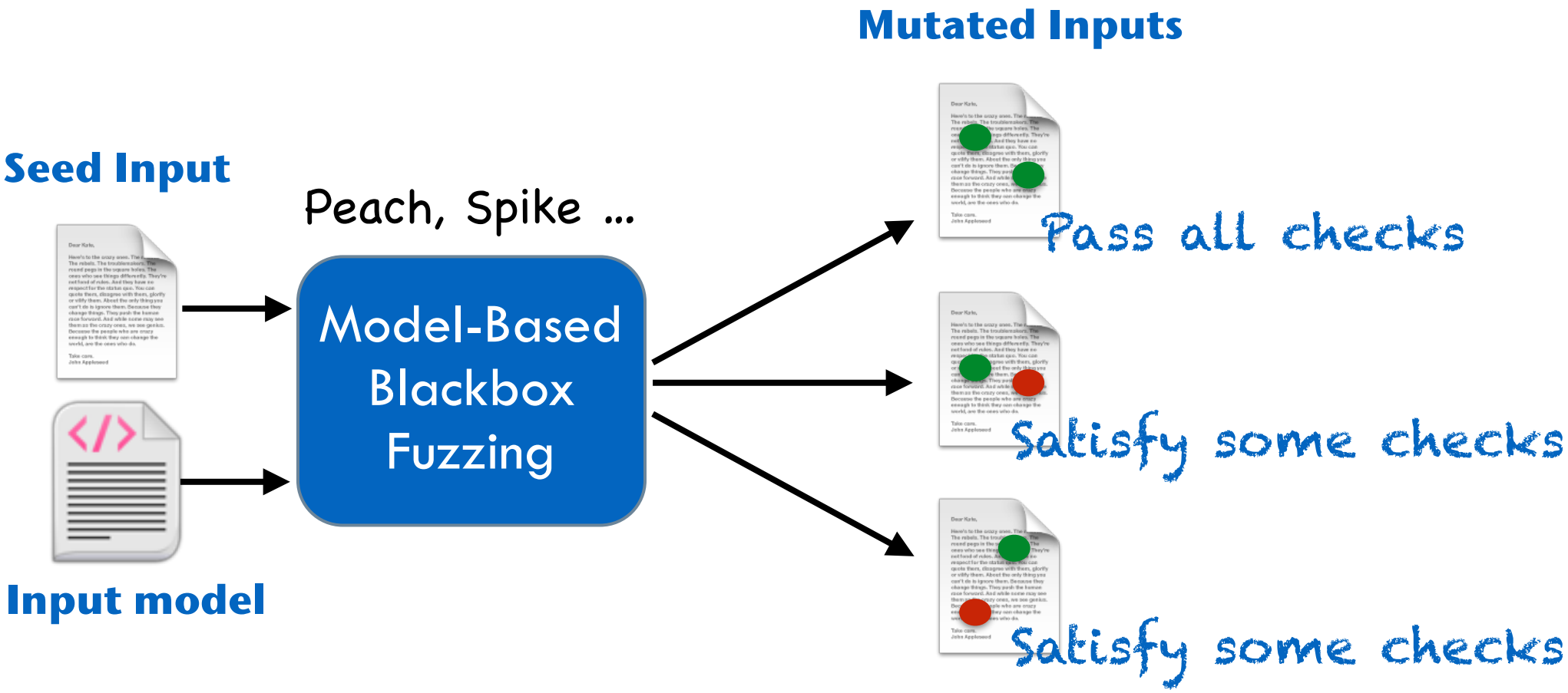
Peach, Spike ...



Model-Based Blackbox Fuzzing



Model-Based Blackbox Fuzzing



Model-Based Blackbox Fuzzing (MoBF)

MoBF struggles at generating specific values for data fields !

Model-Based Blackbox Fuzzing (MoBF)

MoBF struggles at generating specific values for data fields !

Probability to generate correct value(s) for

Model-Based Blackbox Fuzzing (MoBF)

MoBF struggles at generating specific values for data fields !

Probability to generate correct value(s) for
One 32-bit data field: 1/2^32

Model-Based Blackbox Fuzzing (MoBF)

MoBF struggles at generating specific values for data fields !

Probability to generate correct value(s) for

One 32-bit data field: 1/2^32

Two 32-bit data fields: 1/2^64

Model-Based Blackbox Fuzzing (MoBF)

MoBF struggles at generating specific values for data fields !

Probability to generate correct value(s) for

One 32-bit data field:	**1/2^32**
Two 32-bit data fields:	**1/2^64**
Three 32-bit data fields:	**1/2^96**

Model-Based Blackbox Fuzzing (MoBF)

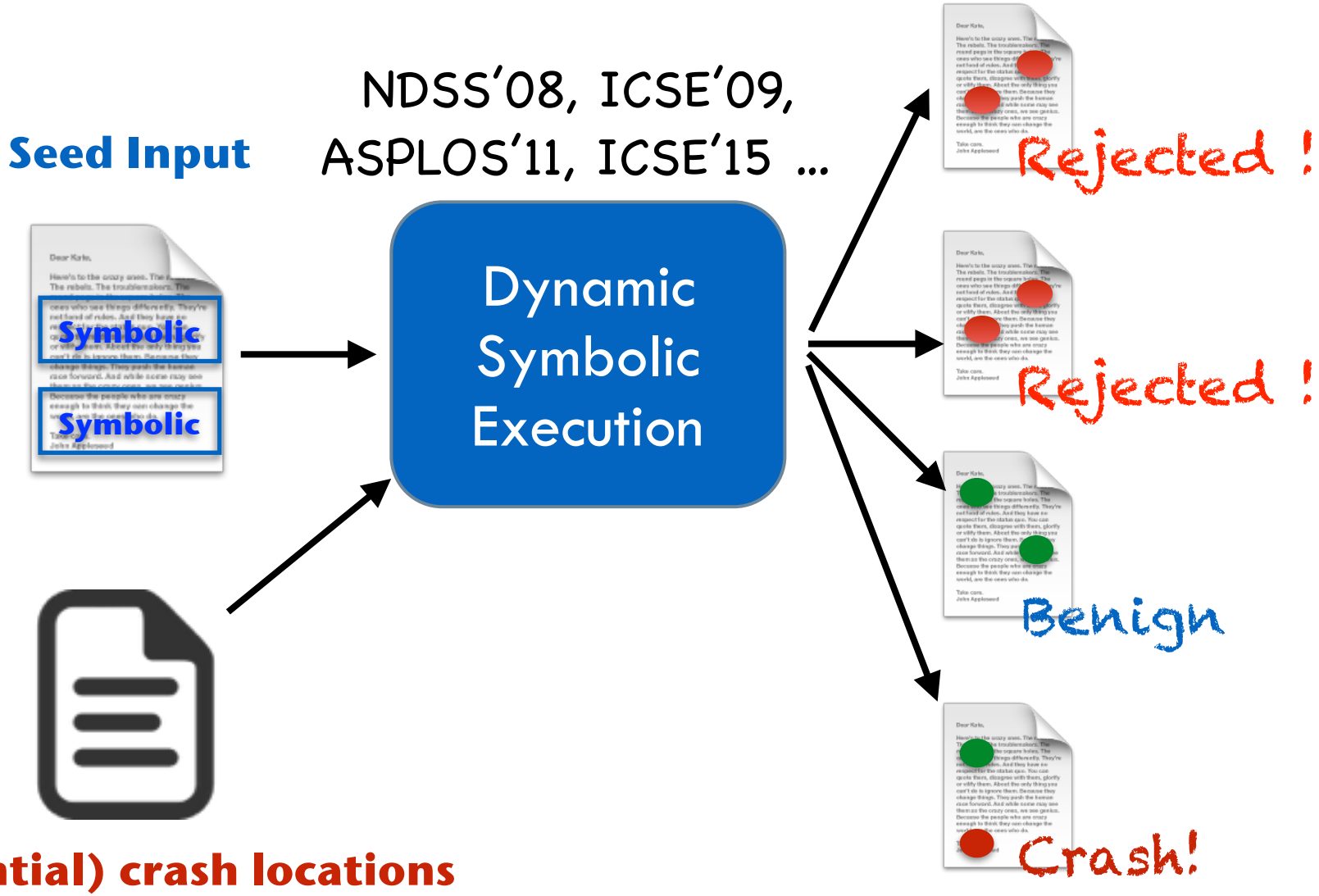
MoBF struggles at generating specific values for data fields !

Probability to generate correct value(s) for

One 32-bit data field:	**1/2^32**
Two 32-bit data fields:	**1/2^64**
Three 32-bit data fields:	**1/2^96**

…

Whitebox Fuzzing



Whitebox Fuzzing (WF)

Whitebox Fuzzing (WF)

WF comfortably generates specific values for data fields

Whitebox Fuzzing (WF)

WF comfortably generates specific values for data fields

WF easily gets bogged down by large space of invalid inputs while

- adding missing data chunk(s) or
- enforcing integrity constraints like checksums, size-of, offset-of ...

Motivating Example

A PNG file triggers a crash in VLC media player

```
"\x89\x50\x4E\x47\x0D\x0A\x1A\x0A" # PNG signature
"\x00\x00\x00\x0D" # IHDR size
"\x49\x48\x44\x52" # IHDR chunk
"\x7F\xFF\xFF\xFF" # width
"\x00\x00\x01\x02" # height
"\x01" # bit depth
"\x03" # color type
"\x00" # compression method
"\x00" # filter method
"\x00" # interlace method
"\xBA\x1B\xD8\x84" # IHDR chunk CRC
"\x00\x00\x00\x03" # PLTE size
"\x50\x4C\x54\x45" # PLTE chunk
"\xFF" # red
"\xFF" # green
"\xFF" # blue
"\xA7\xC4\x1B\xC8" # PLTE chunk CRC
"\x00\x00\x00\x01" # tRNS size
"\x74\x52\x4E\x53" # tRNS chunk
"\x00" # alpha
"\x40\xE6\xD8\x66" # tRNS chunk CRC
"\x00\x00\x00\x01" # IDAT size
"\x49\x44\x41\x54" # IDAT chunk
"\xFF" # image data
"\x05\x3A\x92\x65" # IDAT chunk CRC
"\x00\x00\x00\x00" # IEND size
"\x49\x45\x4E\x44" # IEND chunk
"\xAE\x42\x60\x82" # IEND chunk CRC
```

Motivating Example

A PNG file triggers a crash in VLC media player

```
"\x89\x50\x4E\x47\x0D\x0A\x1A\x0A" # PNG signature
"\x00\x00\x00\x0D" # IHDR size
"\x49\x48\x44\x52" # IHDR chunk
"\x7F\xFF\xFF\xFF" # width
"\x00\x00\x01\x02" # height
"\x01" # bit depth
"\x03" # color type
"\x00" # compression method
"\x00" # filter method
"\x00" # interlace method
"\xBA\x1B\xD8\x84" # IHDR chunk CRC
"\x00\x00\x00\x03" # PLTE size
"\x50\x4C\x54\x45" # PLTE chunk
"\xFF" # red
"\xFF" # green
"\xFF" # blue
"\xA7\xC4\x1B\xC8" # PLTE chunk CRC
"\x00\x00\x00\x01" # tRNS size
"\x74\x52\x4E\x53" # tRNS chunk
"\x00" # alpha
"\x40\xE6\xD8\x66" # tRNS chunk CRC
"\x00\x00\x00\x01" # IDAT size
"\x49\x44\x41\x54" # IDAT chunk
"\xFF" # image data
"\x05\x3A\x92\x65" # IDAT chunk CRC
"\x00\x00\x00\x00" # IEND size
"\x49\x45\x4E\x44" # IEND chunk
"\xAE\x42\x60\x82" # IEND chunk CRC
```

Requires an optional data chunk

Motivating Example

A PNG file triggers a crash in VLC media player

```
"\x89\x50\x4E\x47\x0D\x0A\x1A\x0A" # PNG signature
"\x00\x00\x00\x0D" # IHDR size
"\x49\x48\x44\x52" # IHDR chunk
"\x7F\xFF\xFF\xFF" # width
"\x00\x00\x01\x02" # height
"\x01" # bit depth
"\x03" # color type
"\x00" # compression method
"\x00" # filter method
"\x00" # interlace method
"\xBA\x1B\xD8\x84" # IHDR chunk CRC
"\x00\x00\x00\x03" # PLTE size
"\x50\x4C\x54\x45" # PLTE chunk
"\xFF" # red
"\xFF" # green
"\xFF" # blue
"\xA7\xC4\x1B\xC8" # PLTE chunk CRC
"\x00\x00\x00\x01" # tRNS size
"\x74\x52\x4E\x53" # tRNS chunk
"\x00" # alpha
"\x40\xE6\xD8\x66" # tRNS chunk CRC
"\x00\x00\x00\x01" # IDAT size
"\x49\x44\x41\x54" # IDAT chunk
"\xFF" # image data
"\x05\x3A\x92\x65" # IDAT chunk CRC
"\x00\x00\x00\x00" # IEND size
"\x49\x45\x4E\x44" # IEND chunk
"\xAE\x42\x60\x82" # IEND chunk CRC
```

Requires specific values for some data fields

Requires an optional data chunk

Motivating Example

A PNG file triggers a crash in VLC media player

```
"\x89\x50\x4E\x47\x0D\x0A\x1A\x0A" # PNG signature
"\x00\x00\x00\x0D" # IHDR size
"\x49\x48\x44\x52" # IHDR chunk
"\x7F\xFF\xFF\xFF" # width
"\x00\x00\x01\x02" # height
"\x01" # bit depth
"\x03" # color type
"\x00" # compression method
"\x00" # filter method
"\x00" # interlace method
"\xBA\x1B\xD8\x84" # IHDR chunk CRC
"\x00\x00\x00\x03" # PLTE size
"\x50\x4C\x54\x45" # PLTE chunk
"\xFF" # red
"\xFF" # green
"\xFF" # blue
"\xA7\xC4\x1B\xC8" # PLTE chunk CRC
[redacted]
"\x00\x00\x00\x01" # IDAT size
"\x49\x44\x41\x54" # IDAT chunk
"\xFF" # image data
"\x05\x3A\x92\x65" # IDAT chunk CRC
"\x00\x00\x00\x00" # IEND size
"\x49\x45\x4E\x44" # IEND chunk
"\xAE\x42\x60\x82" # IEND chunk CRC
```

Requires specific values for some data fields

Requires an optional data chunk

Motivating Example

A PNG file triggers a crash in VLC media player

```
"\x89\x50\x4E\x47\x0D\x0A\x1A\x0A" # PNG signature
"\x00\x00\x00\x0D" # IHDR size
"\x49\x48\x44\x52" # IHDR chunk
"\x7F\xFF\xFF\xFF" # width
"\x00\x00\x01\x02" # height
"\x01" # bit depth
"\x03" # color type
"\x00" # compression method
"\x00" # filter method
"\x00" # interlace method
"\xBA\x1B\xD8\x84" # IHDR chunk CRC
"\x00\x00\x00\x03" # PLTE size
"\x50\x4C\x54\x45" # PLTE chunk
"\xFF" # red
"\xFF" # green
"\xFF" # blue
"\xA7\xC4\x1B\xC8" # PLTE chunk CRC
[redacted]
"\x00\x00\x00\x01" # IDAT size
"\x49\x44\x41\x54" # IDAT chunk
"\xFF" # image data
"\x05\x3A\x92\x65" # IDAT chunk CRC
"\x00\x00\x00\x00" # IEND size
"\x49\x45\x4E\x44" # IEND chunk
"\xAE\x42\x60\x82" # IEND chunk CRC
```

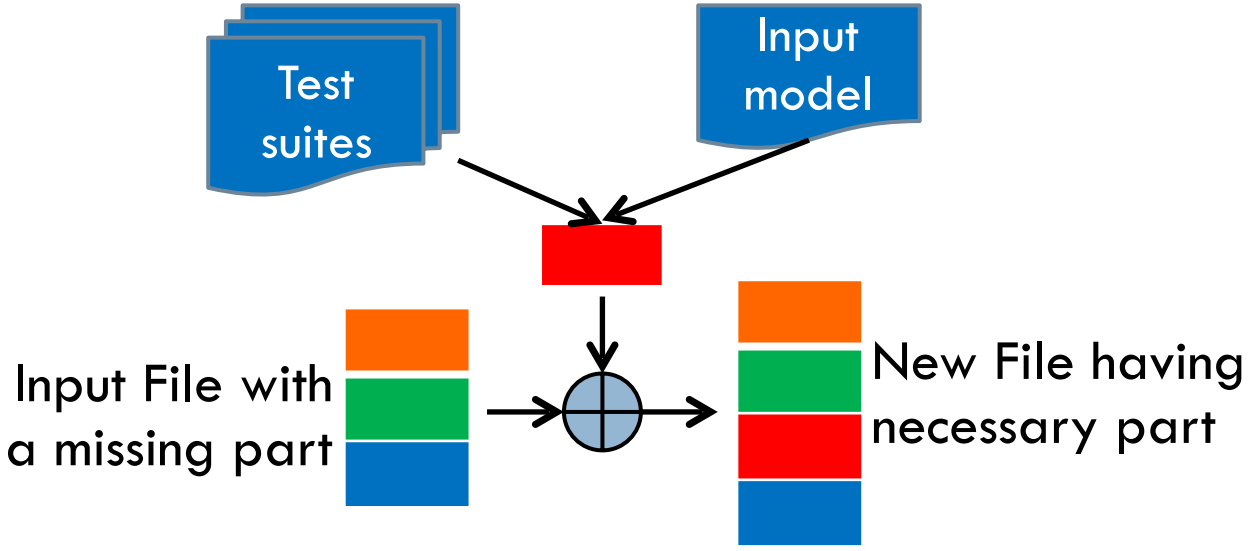
MOBF & WF are very unlikely to generate the crashing input IF the selected seed file does not have optional tRNS data chunk

Observation & Solution

- A missing data chunk can be obtained from other seed inputs in the test suite
- OR it can be directly instantiated from the ***input model***

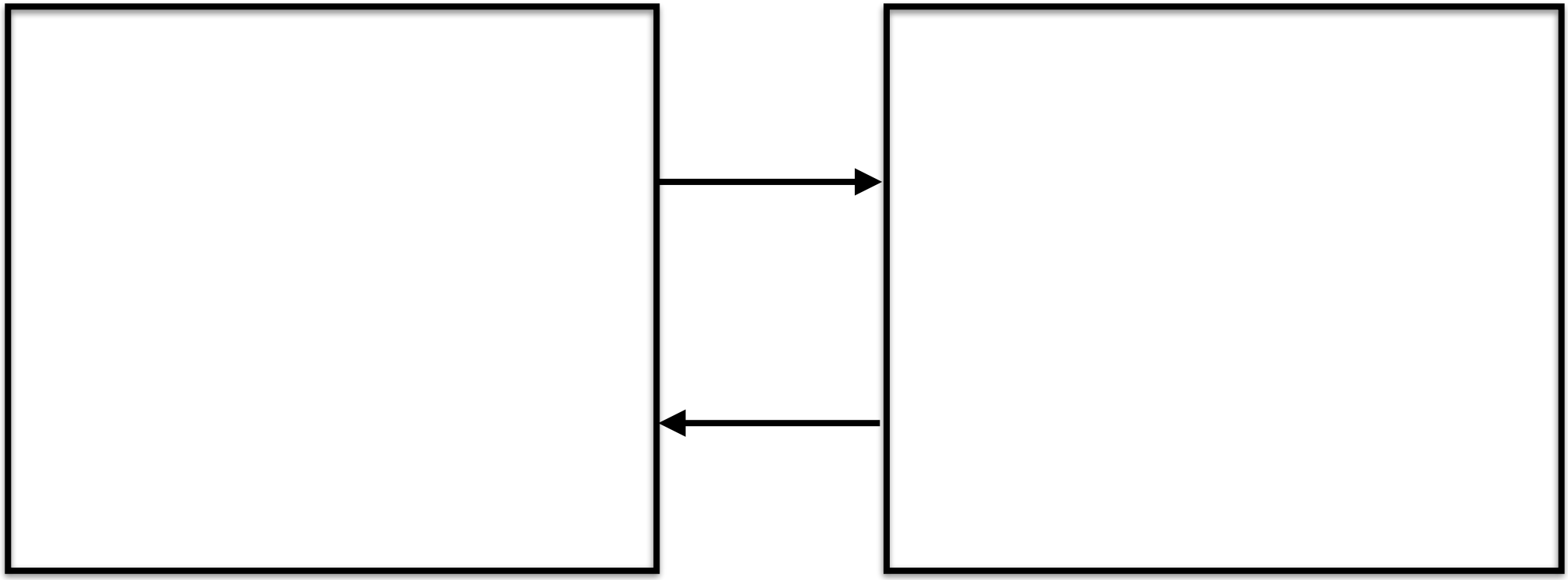
Observation & Solution

- A missing data chunk can be obtained from other seed inputs in the test suite
- OR it can be directly instantiated from the ***input model***



Data chunk Transplantation

Model-Based Whitebox Fuzzing



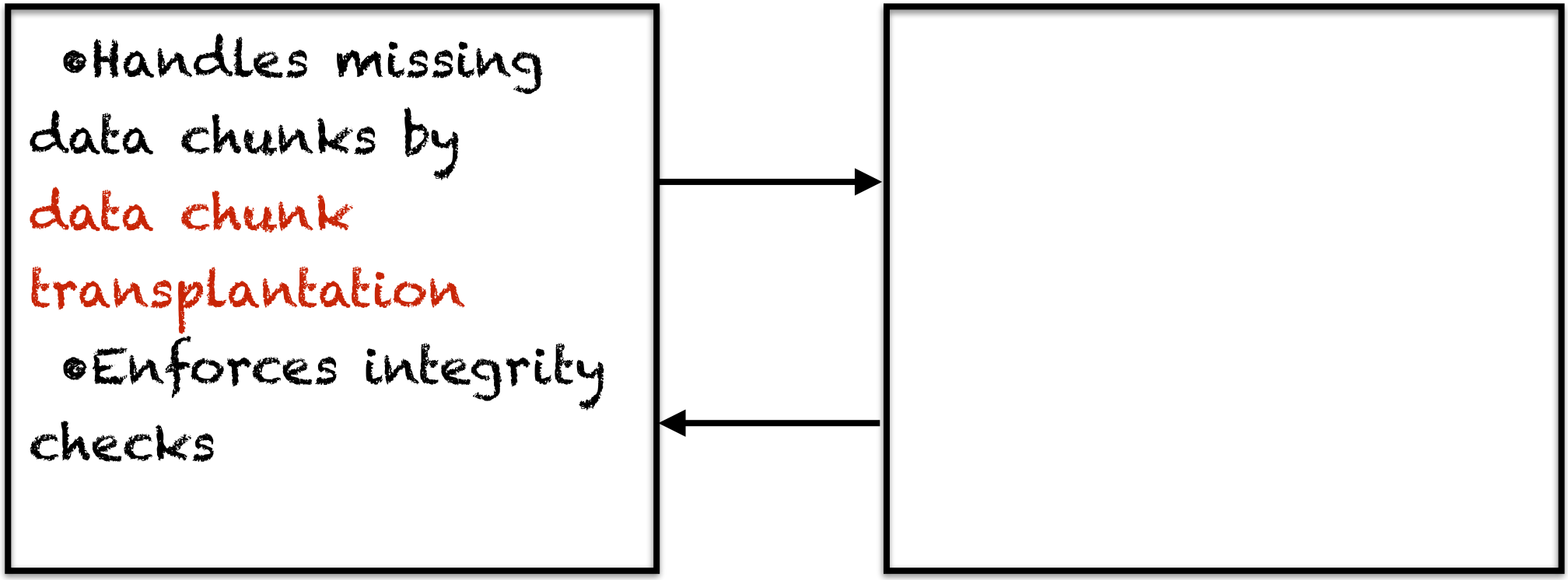
Augmented MoBF

MoBF + Transplantation

Selective and Targeted

Whitebox Fuzzing

Model-Based Whitebox Fuzzing



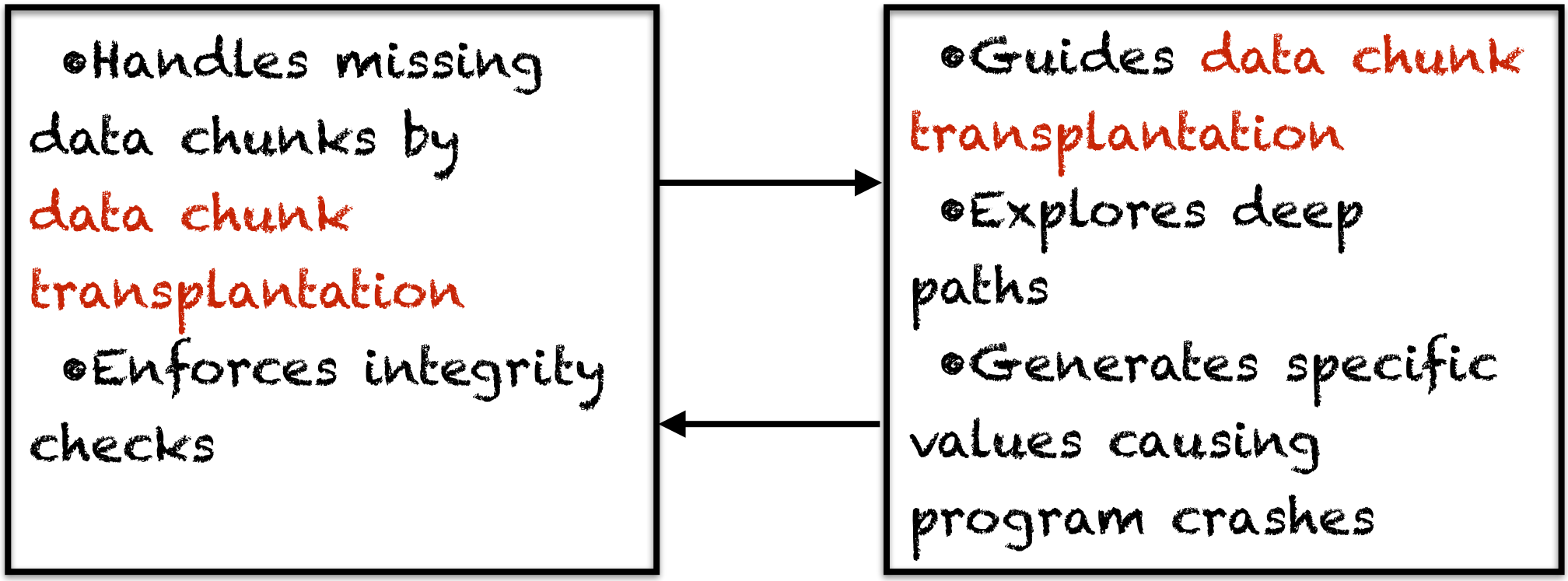
Augmented MoBF

MoBF + Transplantation

Selective and Targeted

Whitebox Fuzzing

Model-Based Whitebox Fuzzing



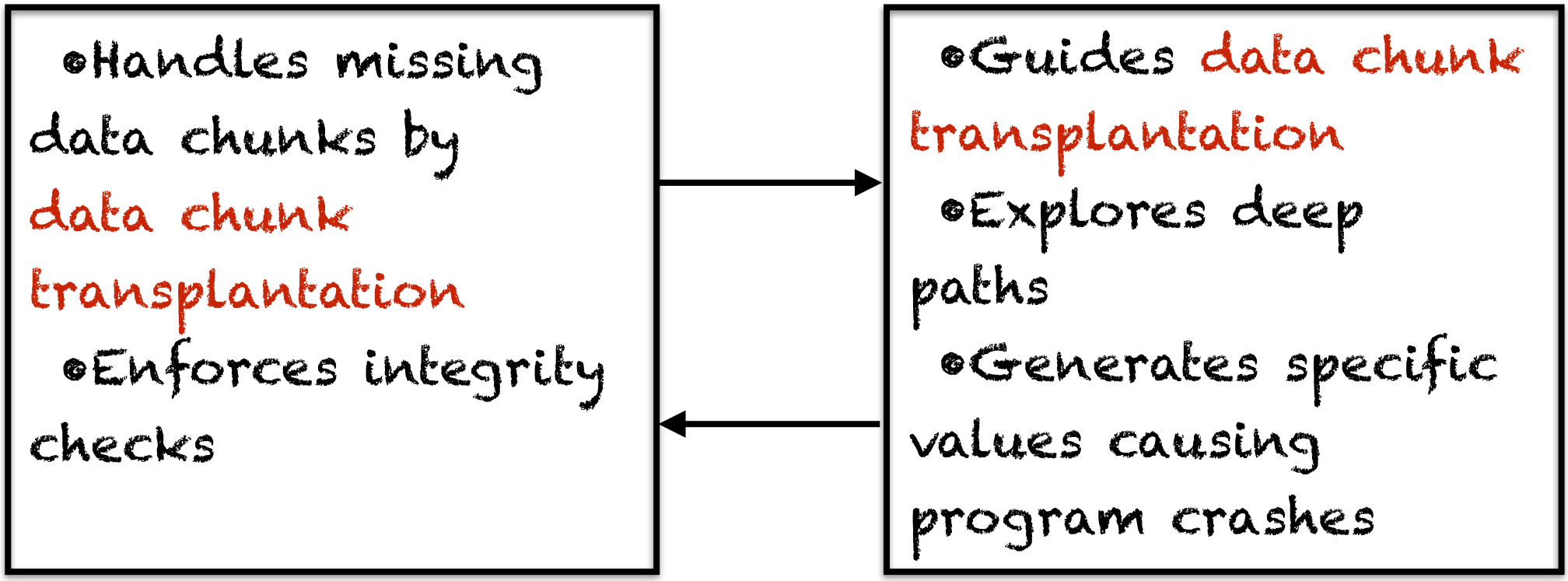
Augmented MoBF

MoBF + Transplantation

Selective and Targeted

Whitebox Fuzzing

Model-Based Whitebox Fuzzing



Augmented MoBF

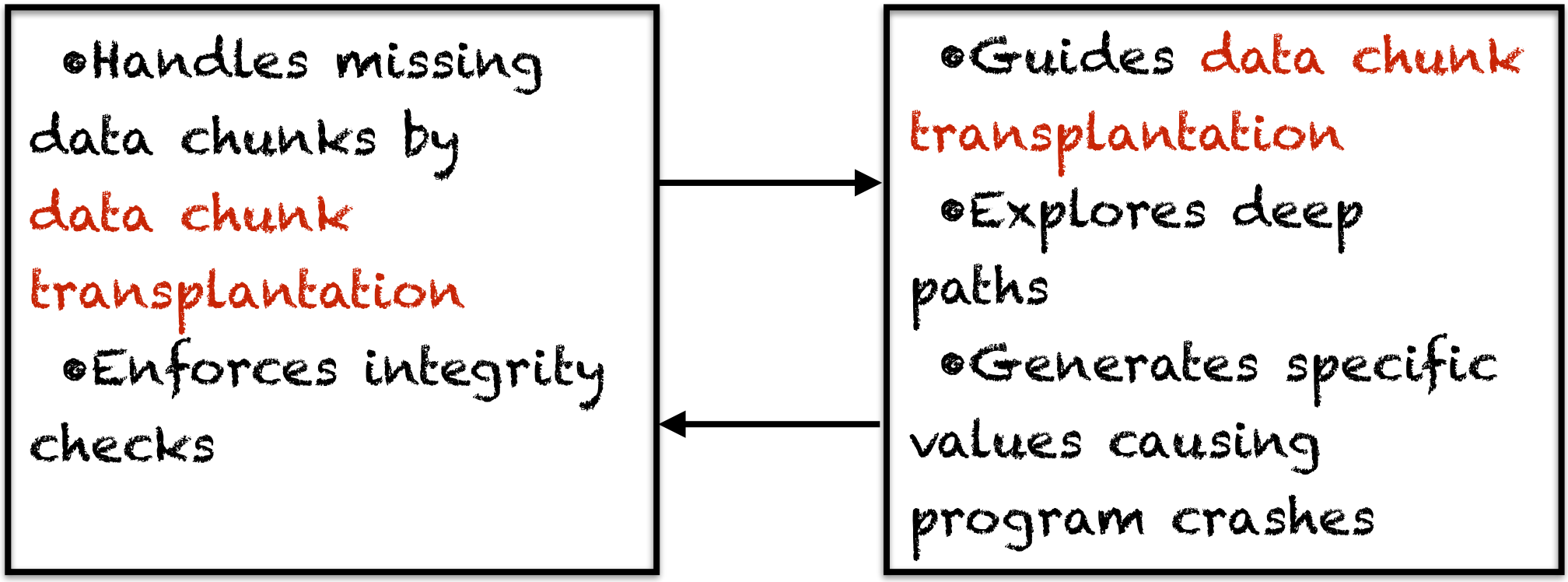
MoBF + Transplantation

Selective and Targeted

Whitebox Fuzzing

Peach Fuzzer
Production-quality MoBF

Model-Based Whitebox Fuzzing



Augmented MoBF

MoBF + Transplantation

Peach Fuzzer
Production-quality MoBF

Selective and Targeted

Whitebox Fuzzing

Hercules (ICSE'15)
Scale to WMP, Adobe Reader

What the input model looks like?

XML-based Input Model (Peach Fuzzer)

Data model for a generic data chunk

```
<DataModel name="Chunk">
    <Number name="Length" size="32" endian="big">
        <Relation type="size" of="Data" />
    </Number>
    <Block name="TypeData">
        <String name="Type" length="4" />
        <Blob name="Data" />
    </Block>
    <Number size="32" endian="big">
        <Fixup class="Crc32Fixup">
            <Param name="ref" value="TypeData"/>
        </Fixup>
    </Number>
</DataModel>
```

XML-based Input Model (Peach Fuzzer)

Data model for a generic data chunk

```
<DataModel name="Chunk">
    <Number name="Length" size="32" endian="big">
        <Relation type="size" of="Data" />
    </Number>
    <Block name="TypeData">
        <String name="Type" length="4" />
        <Blob name="Data" />
    </Block>
    <Number size="32" endian="big">
        <Fixup class="Crc32Fixup">
            <Param name="ref" value="TypeData"/>
        </Fixup>
    </Number>
</DataModel>
```

XML-based Input Model (Peach Fuzzer)

Data model for a generic data chunk

```
<DataModel name="Chunk">
    <Number name="Length" size="32" endian="big">
        <Relation type="size" of="Data" />
    </Number>
    <Block name="TypeData">
        <String name="Type" length="4" />
        <Blob name="Data" />
    </Block>
    <Number size="32" endian="big">
        <Fixup class="Crc32Fixup">
            <Param name="ref" value="TypeData"/>
        </Fixup>
    </Number>
</DataModel>
```

XML-based Input Model (Peach Fuzzer)

Data model for a generic data chunk

```
<DataModel name="Chunk">
    <Number name="Length" size="32" endian="big">
        <Relation type="size" of="Data" />
    </Number>
    <Block name="TypeData">
        <String name="Type" length="4" />
        <Blob name="Data" />
    </Block>
    <Number size="32" endian="big">
        <Fixup class="Crc32Fixup">
            <Param name="ref" value="TypeData"/>
        </Fixup>
    </Number>
</DataModel>
```

```
<DataModel name="Chunk_IHDR" ref="Chunk">
    <Block name="TypeData">
        <String name="Type" value="IHDR" length="4"
        token="true"/>
        <Block name="Data">
            <Number name="width" size="32" />
            <Number name="height" size="32" />
            <Number name="bits" size="8" />
            <Number name="color_type" size="8" />
            <Number name="compression" size="8" />
            <Number name="filter" size="8" />
            <Number name="interlace" size="8" />
        </Block>
    </Block>
</DataModel>
```

XML-based Input Model (Peach Fuzzer)

Data model for a generic data chunk

```
<DataModel name="Chunk">
    <Number name="Length" size="32" endian="big">
        <Relation type="size" of="Data" />
    </Number>
    <Block name="TypeData">
        <String name="Type" length="4" />
        <Blob name="Data" />
    </Block>
    <Number size="32" endian="big">
        <Fixup class="Crc32Fixup">
            <Param name="ref" value="TypeData"/>
        </Fixup>
    </Number>
</DataModel>
```

inherits common data fields & relationships

```
<DataModel name="Chunk_IHDR" ref="Chunk">
    <Block name="TypeData">
        <String name="Type" value="IHDR" length="4"
        token="true"/>
        <Block name="Data">
            <Number name="width" size="32" />
            <Number name="height" size="32" />
            <Number name="bits" size="8" />
            <Number name="color_type" size="8" />
            <Number name="compression" size="8" />
            <Number name="filter" size="8" />
            <Number name="interlace" size="8" />
        </Block>
    </Block>
</DataModel>
```

XML-based Input Model (Peach Fuzzer)

Data model for a generic data chunk

```
<DataModel name="Chunk">
    <Number name="Length" size="32" endian="big">
        <Relation type="size" of="Data" />
    </Number>
    <Block name="TypeData">
        <String name="Type" length="4" />
        <Blob name="Data" />
    </Block>
    <Number size="32" endian="big">
        <Fixup class="Crc32Fixup">
            <Param name="ref" value="TypeData"/>
        </Fixup>
    </Number>
</DataModel>
```

inherits common data fields & relationships

```
<DataModel name="Chunk_IHDR" ref="Chunk">
    <Block name="TypeData">
        <String name="Type" value="IHDR" length="4"
        token="true"/>
        <Block name="Data">
            <Number name="width" size="32" />
            <Number name="height" size="32" />
            <Number name="bits" size="8" />
            <Number name="color_type" size="8" />
            <Number name="compression" size="8" />
            <Number name="filter" size="8" />
            <Number name="interlace" size="8" />
        </Block>
    </Block>
</DataModel>
```

XML-based Input Model (Peach Fuzzer)

Data model for a generic data chunk

```
<DataModel name="Chunk">
    <Number name="Length" size="32" endian="big">
        <Relation type="size" of="Data" />
    </Number>
    <Block name="TypeData">
        <String name="Type" length="4" />
        <Blob name="Data" />
    </Block>
    <Number size="32" endian="big">
        <Fixup class="Crc32Fixup">
            <Param name="ref" value="TypeData"/>
        </Fixup>
    </Number>
</DataModel>
```

inherits common data fields & relationships

```
<DataModel name="Chunk_IHDR" ref="Chunk">
    <Block name="TypeData">
        <String name="Type" value="IHDR" length="4"
        token="true"/>
        <Block name="Data">
            <Number name="width" size="32" />
            <Number name="height" size="32" />
            <Number name="bits" size="8" />
            <Number name="color_type" size="8" />
            <Number name="compression" size="8" />
            <Number name="filter" size="8" />
            <Number name="interlace" size="8" />
        </Block>
    </Block>
</DataModel>
```

Data model for PNG image files

```
<DataModel name="Png">
    <Number name="Signature" valueType="hex"
    value="89504e470d0a1a0a" size="64"
    token="true" />
    <Choice maxOccurs="30000">
        <Block ref="Chunk_IHDR"/>
        <Block ref="Chunk_PLTE"/>
        <Block ref="Chunk_IDAT"/>
        ...
        <Block ref="Chunk_IEND"/>
        <Block ref="Chunk"/>
    </Choice>
</DataModel>
```

XML-based Input Model (Peach Fuzzer)

Data model for a generic data chunk

```
<DataModel name="Chunk">
    <Number name="Length" size="32" endian="big">
        <Relation type="size" of="Data" />
    </Number>
    <Block name="TypeData">
        <String name="Type" length="4" />
        <Blob name="Data" />
    </Block>
    <Number size="32" endian="big">
        <Fixup class="Crc32Fixup">
            <Param name="ref" value="TypeData"/>
        </Fixup>
    </Number>
</DataModel>
```

inherits common data fields & relationships

```
<DataModel name="Chunk_IHDR" ref="Chunk">
    <Block name="TypeData">
        <String name="Type" value="IHDR" length="4"
        token="true"/>
        <Block name="Data">
            <Number name="width" size="32" />
            <Number name="height" size="32" />
            <Number name="bits" size="8" />
            <Number name="color_type" size="8" />
            <Number name="compression" size="8" />
            <Number name="filter" size="8" />
            <Number name="interlace" size="8" />
        </Block>
    </Block>
</DataModel>
```

Data model for PNG image files

```
<DataModel name="Png">
    <Number name="Signature" valueType="hex"
    value="89504e470d0a1a0a" size="64"
    token="true" />
    <Choice maxOccurs="30000">
        <Block ref="Chunk_IHDR"/>
        <Block ref="Chunk_PLTE"/>
        <Block ref="Chunk_IDAT"/>
        ...
        <Block ref="Chunk_IEND"/>
        <Block ref="Chunk"/>
    </Choice>
</DataModel>
```

XML-based Input Model (Peach Fuzzer)

Data model for a generic data chunk

```
<DataModel name="Chunk">
    <Number name="Length" size="32" endian="big">
        <Relation type="size" of="Data" />
    </Number>
    <Block name="TypeData">
        <String name="Type" length="4" />
        <Blob name="Data" />
    </Block>
    <Number size="32" endian="big">
        <Fixup class="Crc32Fixup">
            <Param name="ref" value="TypeData"/>
        </Fixup>
    </Number>
</DataModel>
```

inherits common data fields & relationships

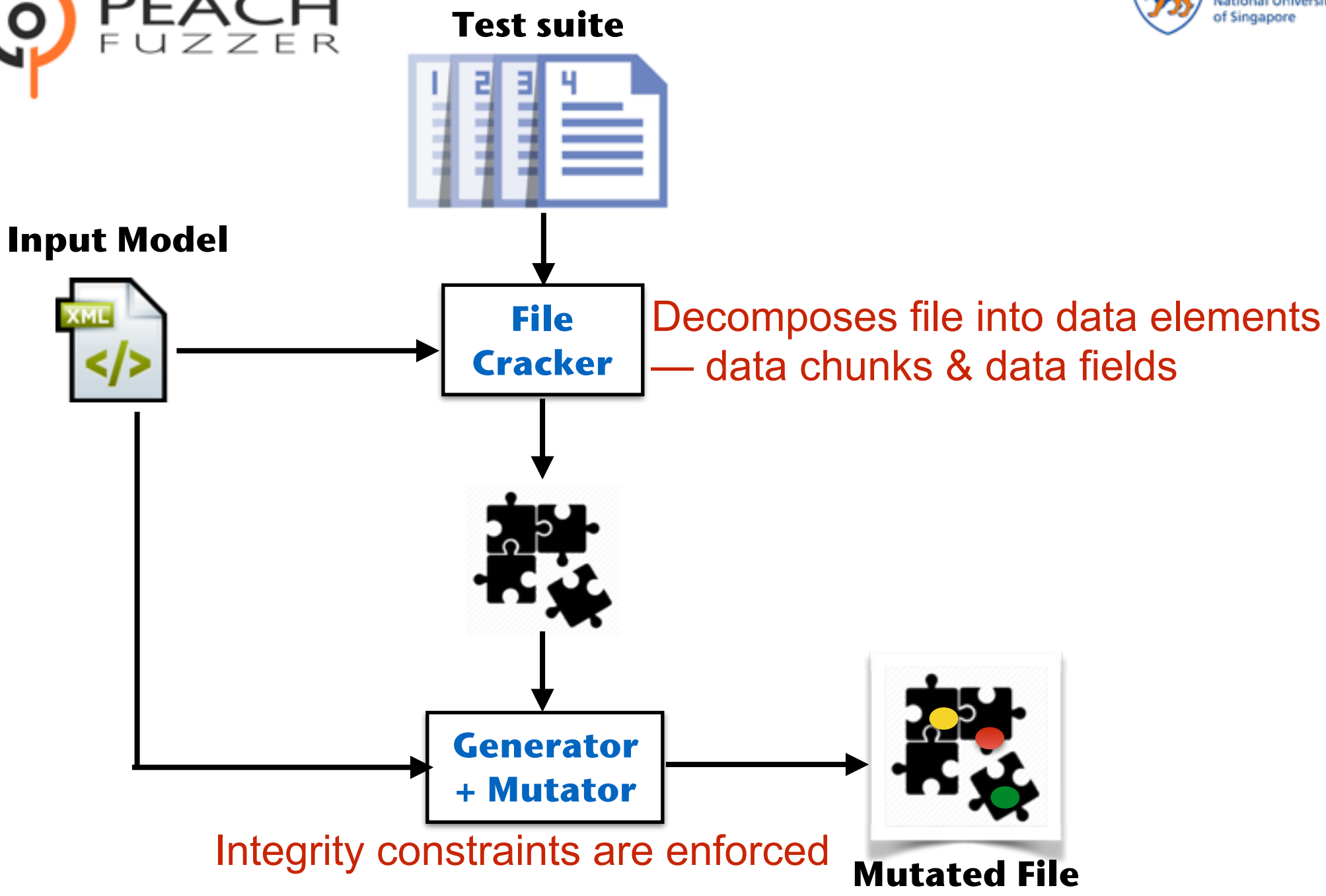
```
<DataModel name="Chunk_IHDR" ref="Chunk">
    <Block name="TypeData">
        <String name="Type" value="IHDR" length="4"
        token="true"/>
        <Block name="Data">
            <Number name="width" size="32" />
            <Number name="height" size="32" />
            <Number name="bits" size="8" />
            <Number name="color_type" size="8" />
            <Number name="compression" size="8" />
            <Number name="filter" size="8" />
            <Number name="interlace" size="8" />
        </Block>
    </Block>
</DataModel>
```

Data model for PNG image files

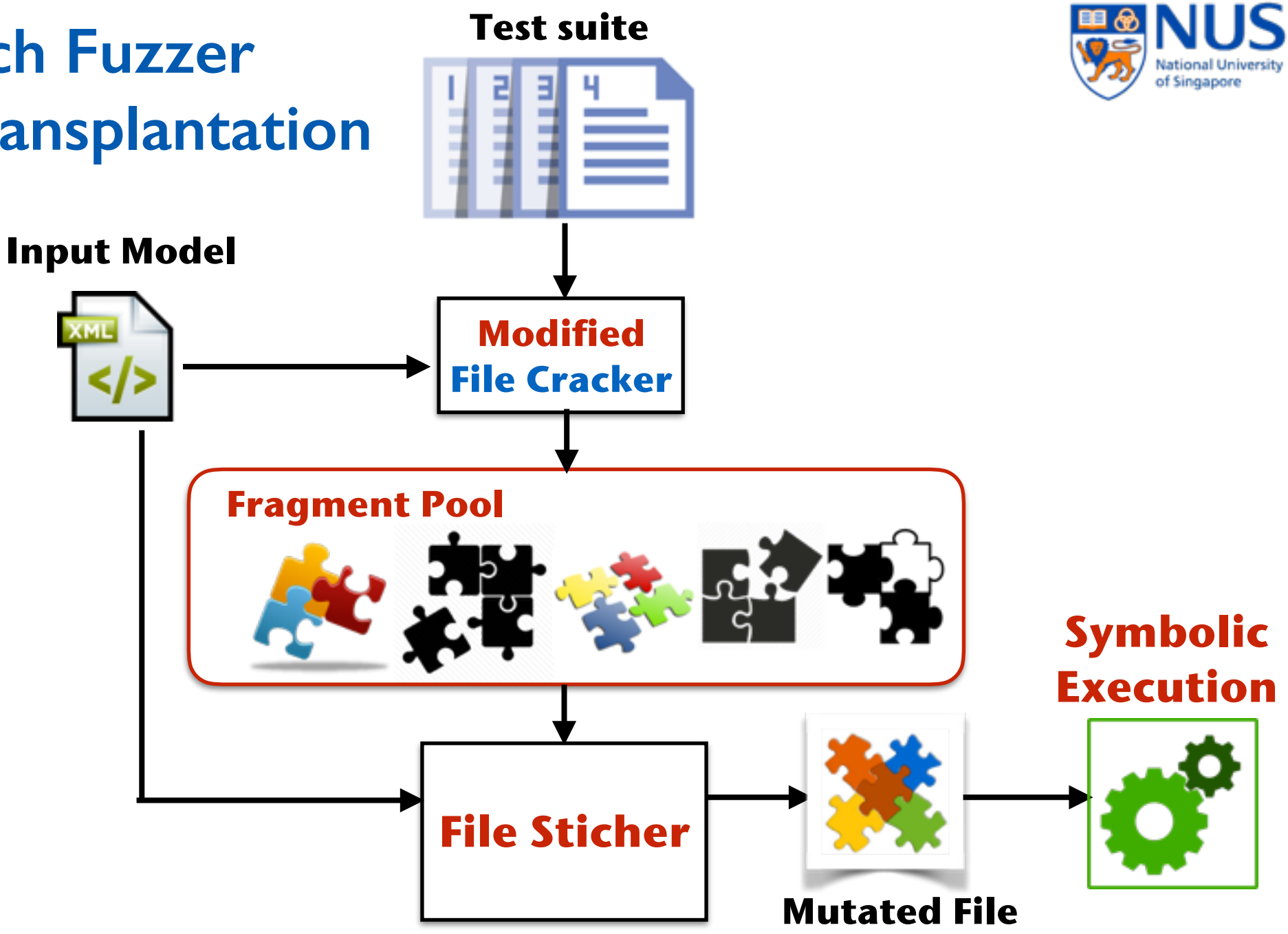
```
<DataModel name="Png">
    <Number name="Signature" valueType="hex"
    value="89504e470d0a1a0a" size="64"
    token="true" />
    <Choice maxOccurs="30000">
        <Block ref="Chunk_IHDR"/>
        <Block ref="Chunk_PLTE"/>
        <Block ref="Chunk_IDAT"/>
        ...
        <Block ref="Chunk_IEND"/>
        <Block ref="Chunk"/>
    </Choice>
</DataModel>
```

PEACH FUZZER

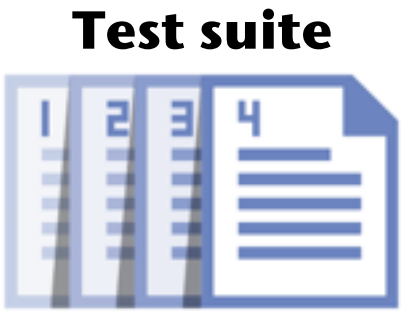
NUS National University of Singapore

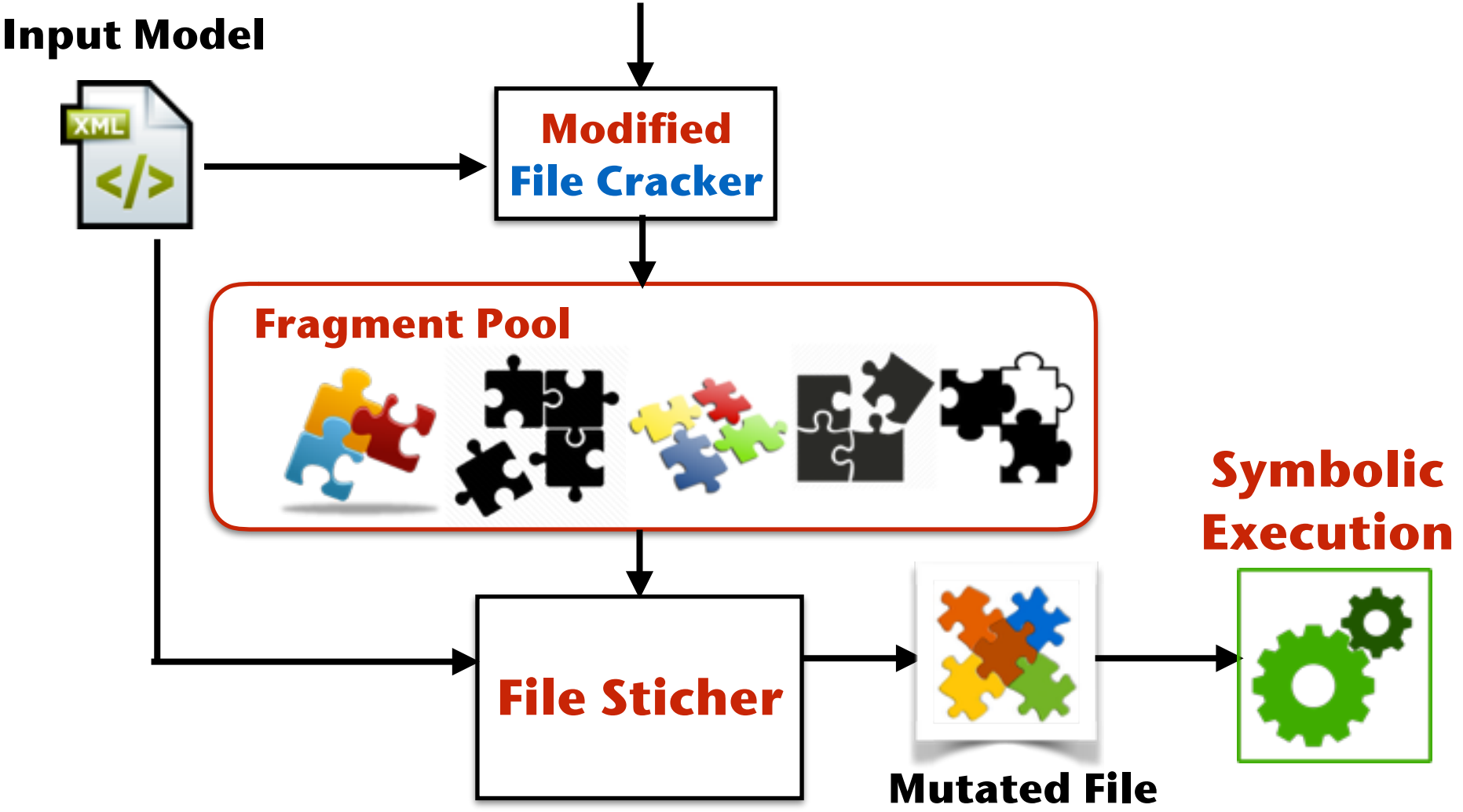


Peach Fuzzer + Transplantation

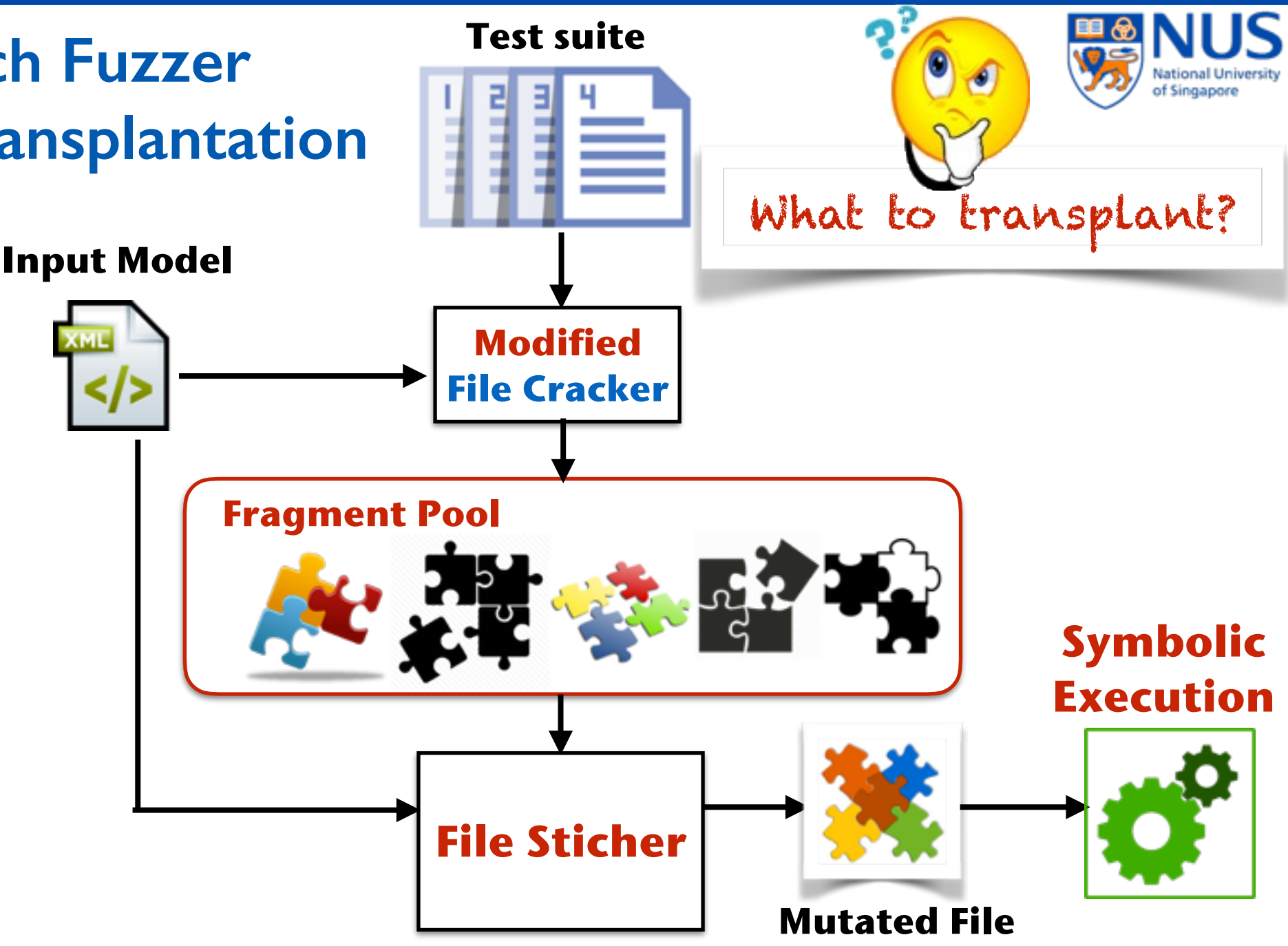


Peach Fuzzer + Transplantation

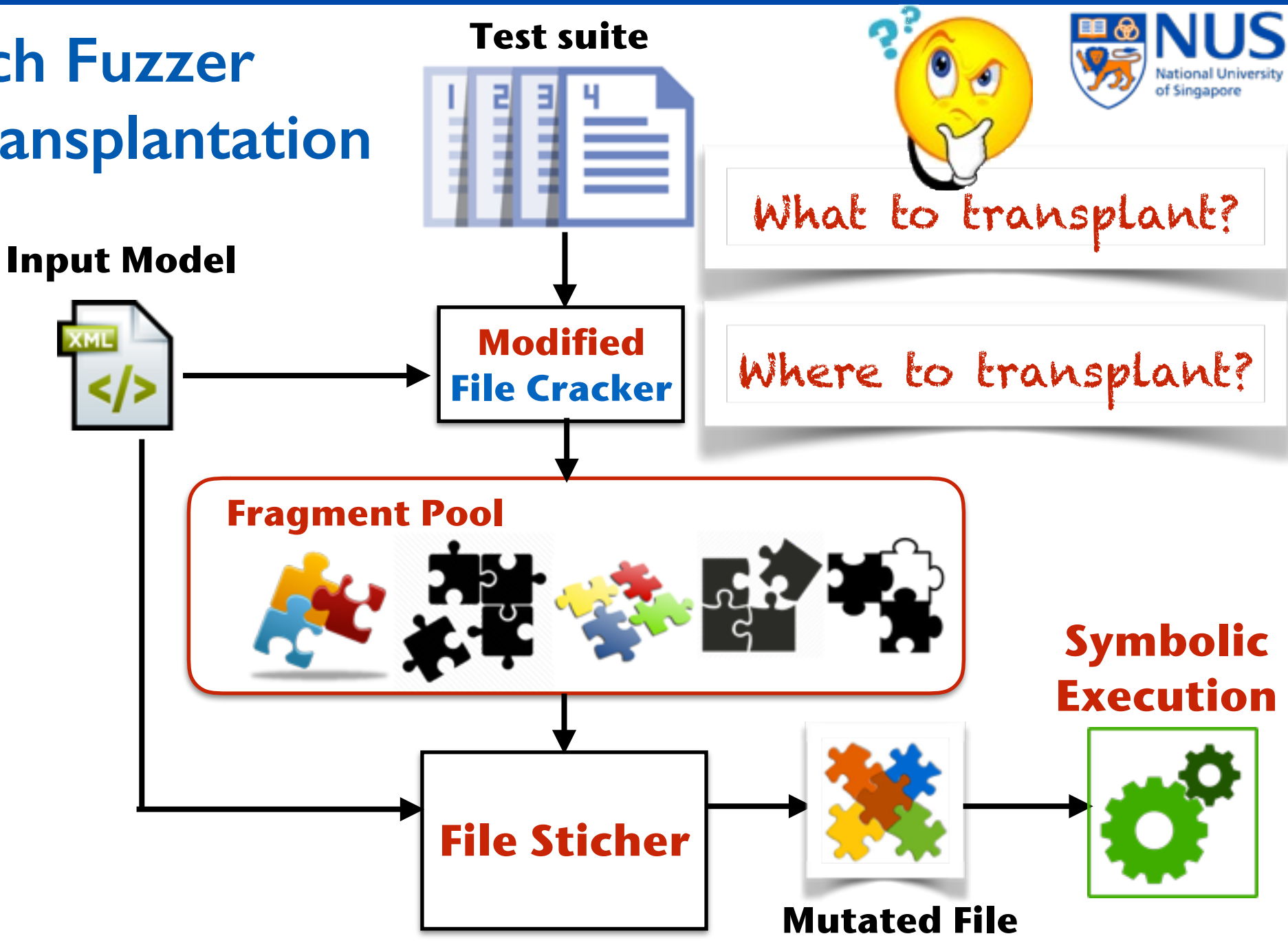




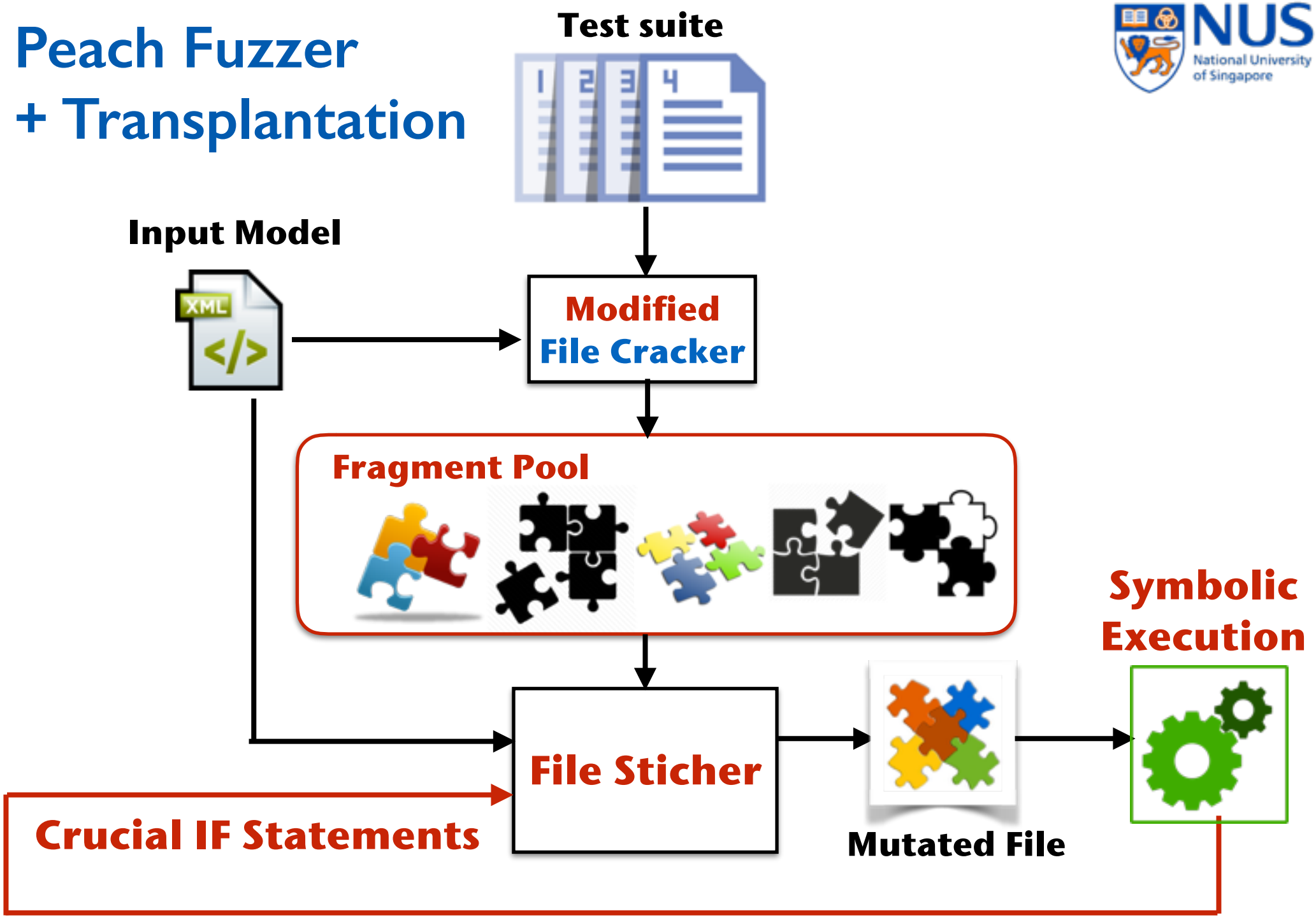
Peach Fuzzer + Transplantation



Peach Fuzzer + Transplantation



Peach Fuzzer + Transplantation



Crucial IF Statements

```
1  // read chunks' info before first IDAT chunk
2  void png_read_info(png_structp ptr)
3  {
4  // read and check the PNG file signature
5  read_sig(f);
6  for (;;)
7  {
8    // get current chunk's information
9    uint_32 length = read_chunk_header(ptr);
10   uint_32 chunk_name = ptr->chunk_name;
11   // mandatory chunks
12   if (chunk_name == png_IHDR)
13      handle_IHDR(ptr, length);
14   else if (chunk_name == png_IEND)
15      handle_IEND(ptr, length);
16   else if (chunk_name == png_PLTE)
17      handle_PLTE(ptr, length);
18   else if (chunk_name == png_IDAT)
19   {
20      ptr->idat_size = length;
21      break;
22   }
23   // optional chunks
24   else if ...
25   else if (chunk_name == png_tRNS)
26      handle_tRNS(ptr, length);
27   else if ...
28 }
29 }
30 // initialize row buffer for reading data from file
31 void png_read_start_row(png_structp ptr)
32 {
33    size_t buf_size;
34    ...
35    buf_size = calculateBufSize(ptr);
36    ptr->row_buf = png_malloc(ptr, buf_size);
37    png_memset(ptr->row_buf, 0, ptr->rowbytes);
38 }
```

Code extracted from LibPNG

Crucial IF Statements

```
1  // read chunks' info before first IDAT chunk
2  void png_read_info(png_structp ptr)
3  {
4  // read and check the PNG file signature
5  read_sig(f);
6  for (;;)
7  {
8    // get current chunk's information
9    uint_32 length = read_chunk_header(ptr);
10   uint_32 chunk_name = ptr->chunk_name;
11   // mandatory chunks
12   if (chunk_name == png_IHDR)
13      handle_IHDR(ptr, length);
14   else if (chunk_name == png_IEND)
15      handle_IEND(ptr, length);
16   else if (chunk_name == png_PLTE)
17      handle_PLTE(ptr, length);
18   else if (chunk_name == png_IDAT)
19   {
20      ptr->idat_size = length;
21      break;
22   }
23   // optional chunks
24   else if ...
25   else if (chunk_name == png_tRNS)
26      handle_tRNS(ptr, length);
27   else if ...
28 }
29 }
30 // initialize row buffer for reading data from file
31 void png_read_start_row(png_structp ptr)
32 {
33    size_t buf_size;
34    ...
35    buf_size = calculateBufSize(ptr);
36    ptr->row_buf = png_malloc(ptr, buf_size);
37    png_memset(ptr->row_buf, 0, ptr->rowbytes);
38 }
```

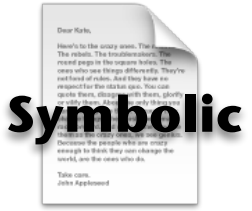
Code extracted from LibPNG

A Crucial IF Statement
- Only one branch has been taken
- depends on the presence of a data chunk in the input file

Detecting Crucial IF Statements

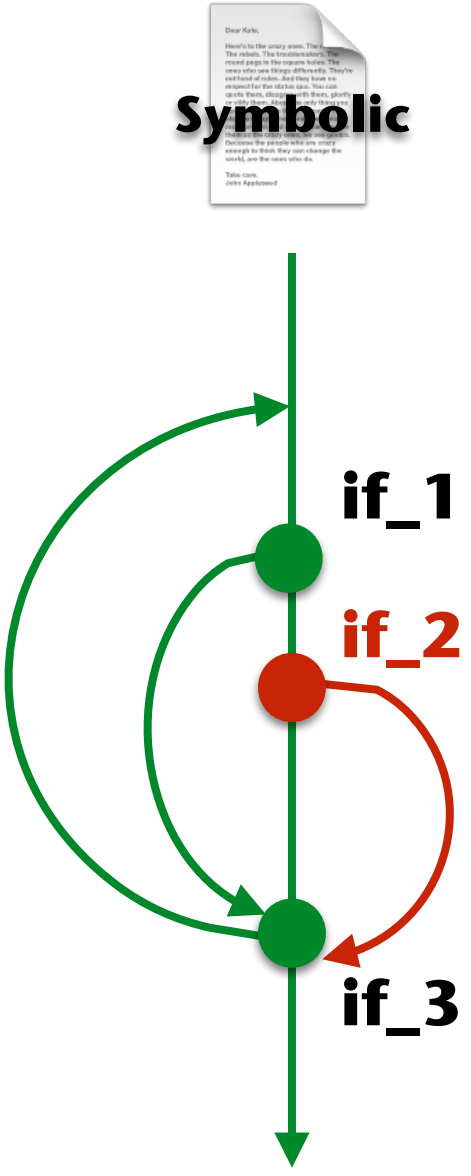
Detecting Crucial IF Statements

- **Step 1.** Mark input file (partially) symbolic



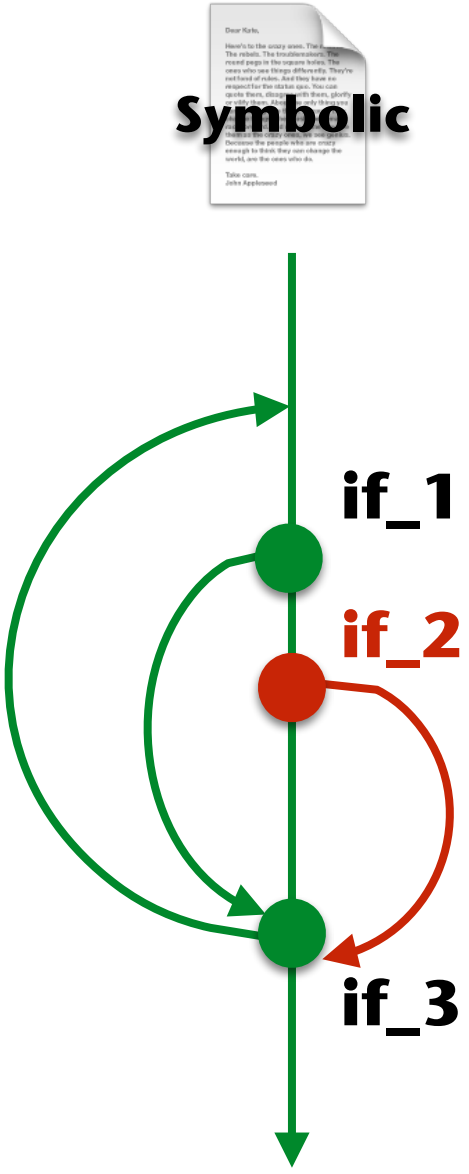
Detecting Crucial IF Statements

- **Step 1.** Mark input file (partially) symbolic
- **Step 2.** Concolically execute program in one path - *same path as concrete input*



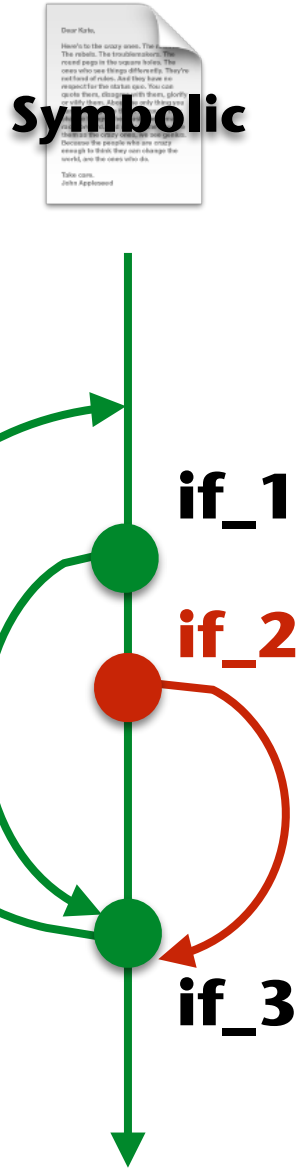
Detecting Crucial IF Statements

- **Step 1.** Mark input file (partially) symbolic
- **Step 2.** Concolically execute program in one path - *same path as concrete input*
- **Step 3.** Collect branch conditions of IF statements at which *only one branch has been taken (e.g., if_2)*



Detecting Crucial IF Statements

- **Step 1.** Mark input file (partially) symbolic
- **Step 2.** Concolically execute program in one path - *same path as concrete input*
- **Step 3.** Collect branch conditions of IF statements at which *only one branch has been taken (e.g., if_2)*
- **Step 4.** Use symbolic-execution-based *taint analysis & input model* to analyse branch conditions (at *if_2*) to validate crucial IFs statements



Evaluation - Subjects & Input Models

Evaluation - Subjects & Input Models

9 subject programs

Program	Version	Buggy module	Size	Errors
Video Lan Client	2.0.7	libpng.dll	184 KB	1
Video Lan Client	2.0.3	libpng.dll	182 KB	1
Libpng Test Program	1.5.4	libpng.dll	176 KB	1
XnView	1.98	XnView.exe	4.46 MB	0 + 3
Adobe Reader	9.2	cooltype.dll	2.32 MI	1
Windows Media Playe	9.0	quartz.dll	1.22 MI	2 + 1
Real Player SP	1.0	realplay.exe	60 KB	1
MIDI Player	0.35	mamplayer.ex	336 KB	1
Orbital Viewer	1.04	ov.exe	538 KB	1
			Total:	9 + 4

Evaluation - Subjects & Input Models

9 subject programs

Program	Version	Buggy module	Size	Errors
Video Lan Client	2.0.7	libpng.dll	184 KB	1
Video Lan Client	2.0.3	libpng.dll	182 KB	1
Libpng Test Program	1.5.4	libpng.dll	176 KB	1
XnView	1.98	XnView.exe	4.46 MB	0 + 3
Adobe Reader	9.2	cooltype.dll	2.32 MI	1
Windows Media Playe	9.0	quartz.dll	1.22 MI	2 + 1
Real Player SP	1.0	realplay.exe	60 KB	1
MIDI Player	0.35	mamplayer.ex	336 KB	1
Orbital Viewer	1.04	ov.exe	538 KB	1
			Total:	9 + 4

6 Input models

Format	Size	Time spent	#Files	Average size
PDF	4.5 KB	12 hours	10	200 KB
PNG	8.3 KB	4 hours	10	55 KB
MIDI	13.9 KB	4 hours	10	20 KB
FLV	6.0 KB	4 hours	10	300 KB
ORB	6.0 KB	8 hours	10	4 KB
WAV*	7.5 KB	2 hours	10	260 KB

One-time effort

34 hrs

Evaluation - Effectiveness of MoWF

Time bound: 24hrs

Program	Advisory ID	Input Model	#Seed files	MoWF	Peach	Hercules
VLC 2.0.7	OSVDB-95632	PNG	10	✓	✗	✗
VLC 2.0.3	CVE-2012-5470	PNG	10	✓	✗	✗
LTP 1.5.4	CVE-2011-3328	PNG	10	✓	✗	✗
XNV1.98	Unknown-1	PNG	10	✓	✓	✗
XNV1.98	Unknown-2	PNG	10	✓	✓	✗
XNV1.98	Unknown-3	PNG	10	✓	✓	✗
WMP 9.0	Unknown-4	WAV	10	✓	✓	✗
WMP 9.0	CVE-2014-2671	WAV	10	✓	✗	✓
WMP 9.0	CVE-2010-0718	MIDI	10	✓	✗	✓
AR 9.2	CVE-2010-2204	PDF	10	✓	✗	✓
RP 1.0	CVE-2010-3000	FLV	10	✓	✗	✓
MP 0.35	CVE-2011-0502	MIDI	10	✓	✓	✓
OV 1.04	CVE-2010-0688	ORB	10	✓	✓	✓

Evaluation - Seed Input Dependence

Program	Advisory ID	Input Model	#Seed files	Hercules++
VLC 2.0.7	OSVDB-95632	PNG	0	✓
VLC 2.0.3	CVE-2012-5470	PNG	0	✓
LTP 1.5.4	CVE-2011-3328	PNG	0	✓
XNV1.98	Unknown-1	PNG	0	✓
XNV1.98	Unknown-2	PNG	0	✓
XNV1.98	Unknown-3	PNG	0	✓
WMP 9.0	Unknown-4	WAV	0	✗
WMP 9.0	CVE-2014-2671	WAV	0	✗
WMP 9.0	CVE-2010-0718	MIDI	0	✓
AR 9.2	CVE-2010-2204	PDF	0	✗
RP 1.0	CVE-2010-3000	FLV	0	✗
MP 0.35	CVE-2011-0502	MIDI	0	✓
OV 1.04	CVE-2010-0688	ORB	0	✓

Evaluation - Seed Input Dependence

Program	Advisory ID	Input Model	#Seed files	Hercules++
VLC 2.0.7	OSVDB-95632	PNG	0	✓
VLC 2.0.3	CVE-2012-5470	PNG	0	✓
LTP 1.5.4	CVE-2011-3328	PNG	0	✓
XNV1.98	Unknown-1	PNG	0	✓
XNV1.98	Unknown-2	PNG	0	✓
XNV1.98	Unknown-3	PNG	0	✓
WMP 9.0	Unknown-4	WAV	0	✗
WMP 9.0	CVE-2014-2671	WAV	0	✗
WMP 9.0	CVE-2010-0718	MIDI	0	✓
AR 9.2	CVE-2010-2204	PDF	0	✗
RP 1.0	CVE-2010-3000	FLV	0	✗
MP 0.35	CVE-2011-0502	MIDI	0	✓
OV 1.04	CVE-2010-0688	ORB	0	✓

70%
No seed file is needed

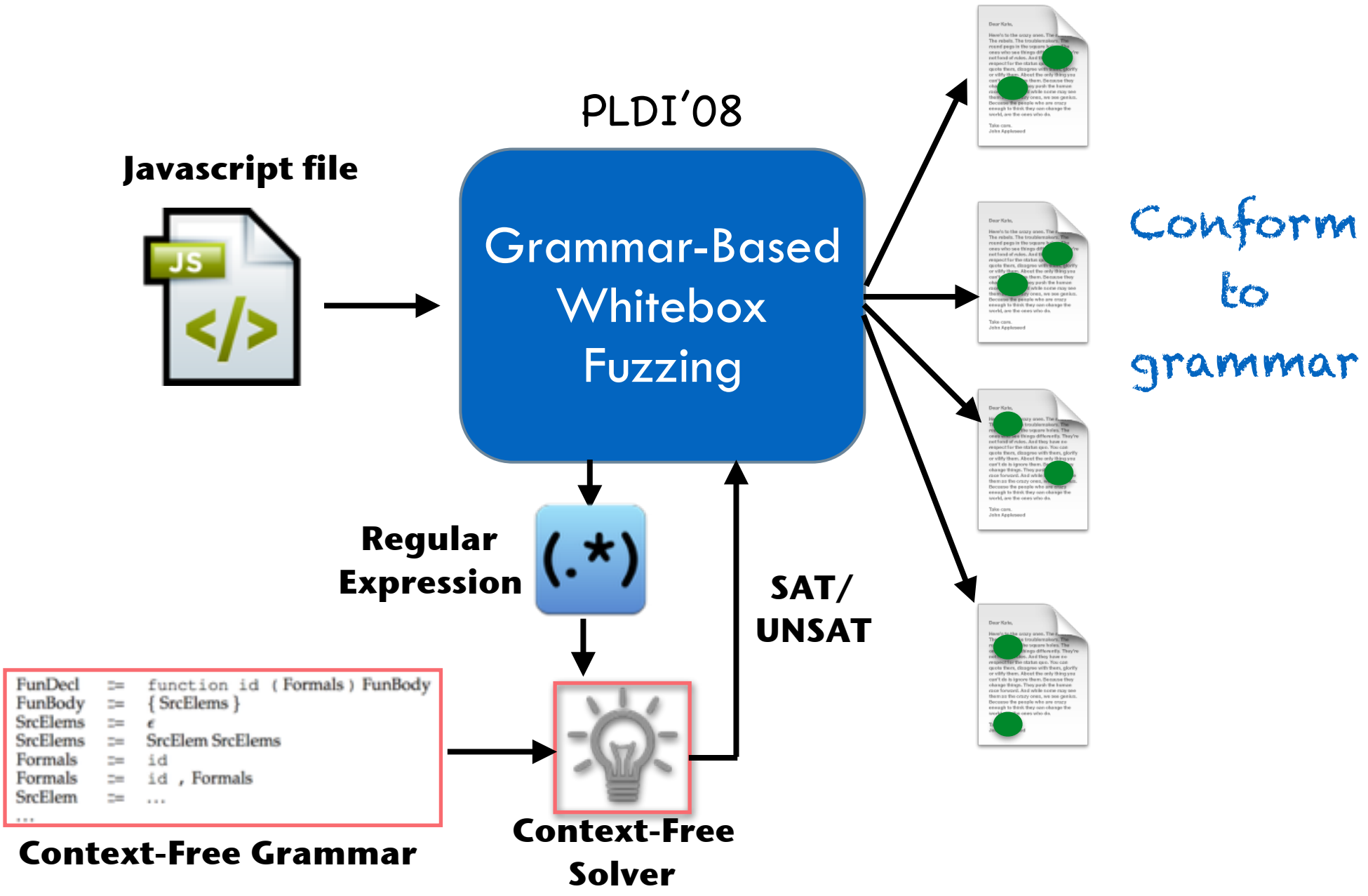
Related Work

NUS National University of Singapore

Related Work

Grammar-based whitebox fuzzing (PLDI'08)

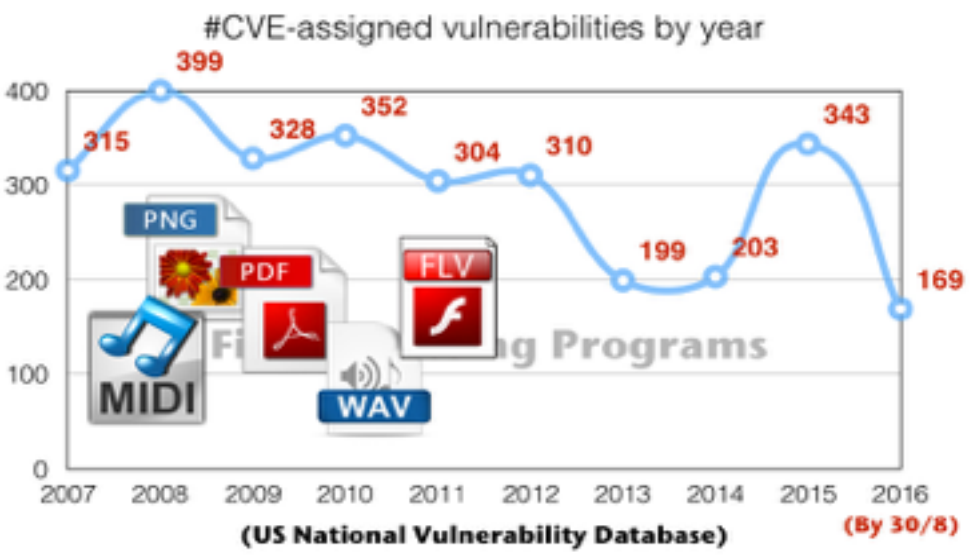
Grammar-based Whitebox Fuzzing (GWF)



MoWF vs GWF

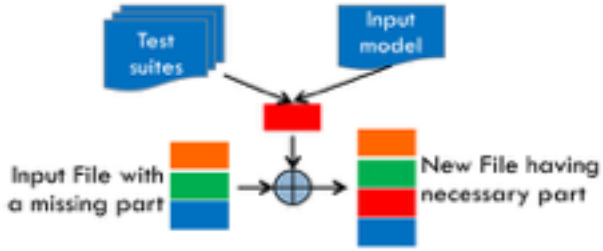
- Regular Expression (GWF) is much weaker than full Path Condition - it cannot encode simple arithmetic constraint like "x<y"
- MoWF maintains full Path Condition and has no impact on the soundness and completeness of Whitebox Fuzzing technique
- MoWF leverages File format input model - more expressive yet simple than Context-Free Grammar. It can comfortably handle integrity constraints like length-of, offset-of and checksums

Vulnerabilities in file-processing programs



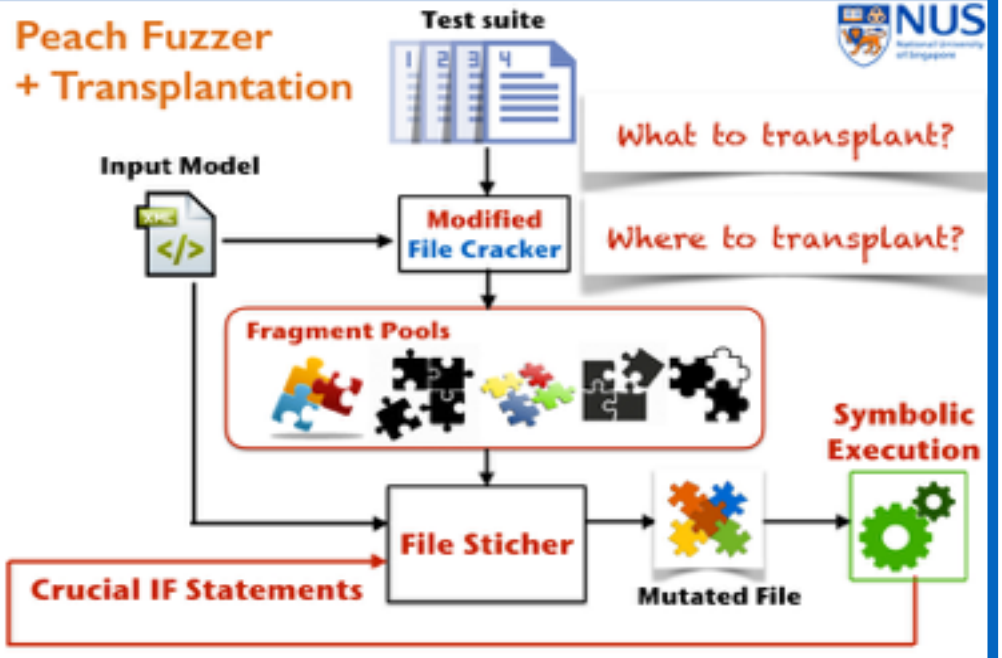
Observation & Solution

- A missing data chunk can be obtained from other seed inputs in the test suite
- OR it can be directly instantiated from an ***input model***



Data chunk Transplantation

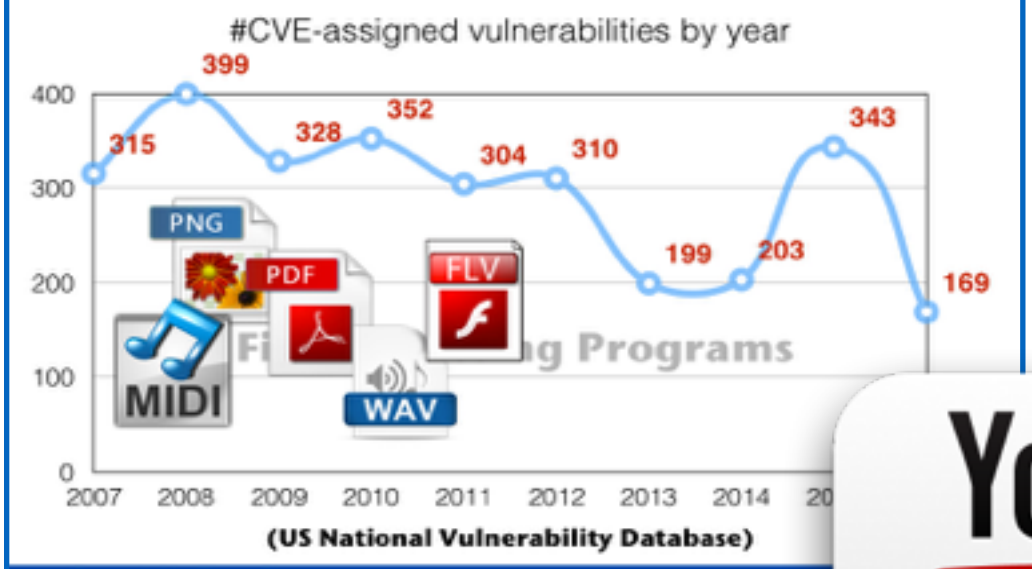
Peach Fuzzer + Transplantation



Evaluation - Effectiveness of MoWF

Program	Advisory ID	Input Model	#Seed Files	Hercules++	Peach	Hercules
VLC 2.0.7	OSVDB-95632	PNG	10	✓	✗	✗
VLC 2.0.3	CVE-2012-5470	PNG	10	✓	✗	✗
LTP 1.5.4	CVE-2011-3328	PNG	10	✓	✗	✗
XNV1.98	Unknown-1	PNG	10	✓	✓	✗
XNV1.98	Unknown-2	PNG	10	✓	✓	✗
XNV1.98	Unknown-3	PNG	10	✓	✓	✗
WMP 9.0	Unknown-4	WAV	10	✓	✓	✗
WMP 9.0	CVE-2014-2671	WAV	10	✓	✗	✓
WMP 9.0	CVE-2010-0718	MIDI	10	✓	✗	✓
AR 9.2	CVE-2010-2204	PDF	10	✓	✗	✓
RP 1.0	CVE-2010-3000	FLV	10	✓	✗	✓
MP 0.35	CVE-2011-0502	MIDI	10	✓	✓	✓
OV 1.04	CVE-2010-0688	ORB	10	✓	✓	✓

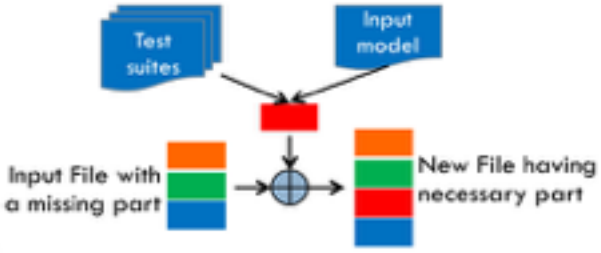
Vulnerabilities in file-processing programs



Presented by Thuan Pham

Observation & Solution

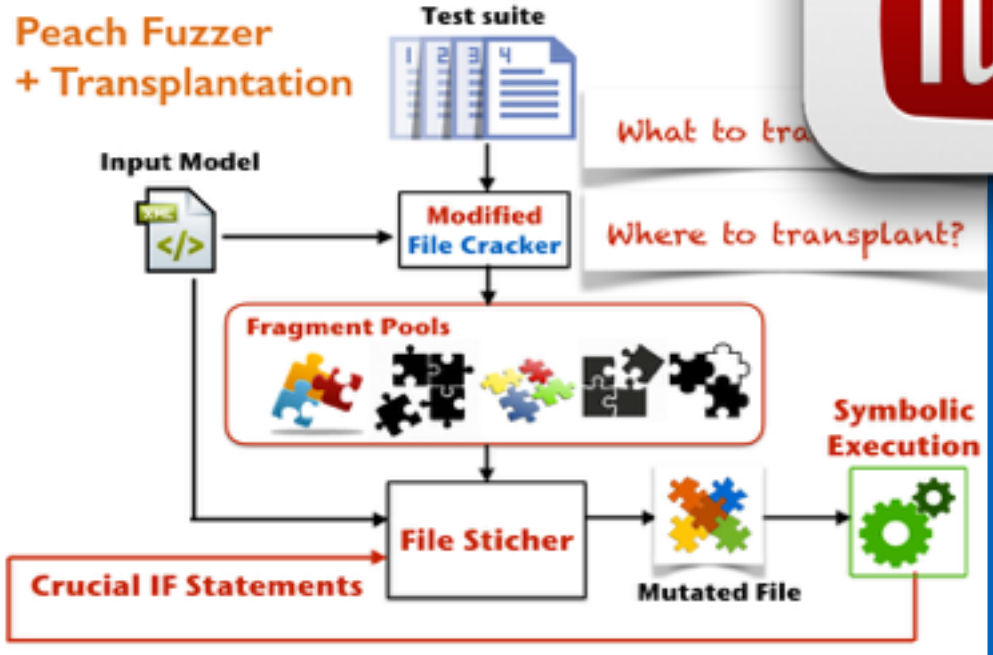
- A missing data chunk can be obtained from other seed inputs in the test suite
- OR it can be directly instantiated from an ***input model***



Data chunk Transplantation

Thuan Pham 11

Peach Fuzzer + Transplantation



Presented by Thuan Pham 15

on - Effectiveness of MoWF

	Advisory ID	Input Model	#Seed Files	Hercules++	Peach	Hercules
VLC 2.0.7	OSVDB-95632	PNG	10	✓	✗	✗
VLC 2.0.3	CVE-2012-5470	PNG	10	✓	✗	✗
LTP 1.5.4	CVE-2011-3328	PNG	10	✓	✗	✗
XNV1.98	Unknown-1	PNG	10	✓	✓	✗
XNV1.98	Unknown-2	PNG	10	✓	✓	✗
XNV1.98	Unknown-3	PNG	10	✓	✓	✗
WMP 9.0	Unknown-4	WAV	10	✓	✓	✗
WMP 9.0	CVE-2014-2671	WAV	10	✓	✗	✓
WMP 9.0	CVE-2010-0718	MIDI	10	✓	✗	✓
AR 9.2	CVE-2010-2204	PDF	10	✓	✗	✓
RP 1.0	CVE-2010-3000	FLV	10	✓	✗	✓
MP 0.35	CVE-2011-0502	MIDI	10	✓	✓	✓
OV 1.04	CVE-2010-0688	ORB	10	✓	✓	✓

Presented by Thuan Pham 21