The Bugs Framework (BF)
Agenda

● Terminology:
  ○ Bug, Weakness
  ○ Vulnerability
  ○ Failure
● Existing Repositories:
  ○ CWE
  ○ CVE
  ○ NVD

● The Bugs Framework (BF)
  ○ Goals
  ○ Features
● Example – Heartbleed
● Potential Impacts
Terminology
Bug, Weakness, Vulnerability, Failure

- **Software Bug:**
  - A coding error
  - Needs to be fixed

- **Software Weakness – difficult to define:**
  - Caused by a bug or ill-formed data
  - Weakness Type – a meaningful notion!

- **Software Vulnerability:**
  - An instance of a weakness type that leads to a security failure
  - May have several underlying weaknesses
Existing Repositories
Commonly Used Repositories

- Weaknesses:
  CWE – Common Weakness Enumeration

- Vulnerabilities:
  CVE – Common Vulnerabilities and Exposures
  → over 18,000 documented in 2020

- Linking weaknesses to vulnerabilities – CWEs to CVEs:
  NVD – National Vulnerabilities Database
1. Imprecise Descriptions – CWE & CVE
2. Unclear Causality – CWE & CVE
3. Gaps in Coverage – CWE
4. Overlaps in Coverage – CWE
Problem #1: Imprecise Descriptions

- Example:

  CWE-502: Deserialization of Untrusted Data:
The application deserializes untrusted data without *sufficiently verifying that the resulting data will be valid*.

  - Unclear what “*sufficiently*” means,
  - “*verifying that data is valid*” is also confusing
Problem #2: Unclear Causality

- Example:

**CVE-2018-5907**
Possible buffer overflow in `msm_adsp_stream_callback_put` due to lack of input validation of user-provided data that leads to integer overflow in all Android releases (Android for MSM, Firefox OS for MSM, QRD Android) from CAF using the Linux kernel.

→ the NVD label is **CWE-190**

While the CWEs chain is:
CWE-20 → CWE-190 → CWE-119
Problems #3, #4: Gaps/Overlaps in Coverage

- Example:

CWEs coverage of buffer overflow by:

✓ Read/ Write
✓ Over/ Under
✓ Stack/ Heap

<table>
<thead>
<tr>
<th></th>
<th>Over</th>
<th>Under</th>
<th>Either End</th>
<th>Stack</th>
<th>Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>CWE-127</td>
<td>CWE-126</td>
<td>CWE-125</td>
<td>✶</td>
<td>✶</td>
</tr>
<tr>
<td>Write</td>
<td>CWE-124</td>
<td>CWE-120</td>
<td>CWE-123</td>
<td>CWE-787</td>
<td>✶</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CWE-121</td>
<td>CWE-122</td>
</tr>
<tr>
<td>Read/ Write</td>
<td>CWE-786</td>
<td>CWE-788</td>
<td>✶</td>
<td>✶</td>
<td>✶</td>
</tr>
</tbody>
</table>
The Bugs Framework (BF)
BF Goals

1. Solve the problems of imprecise descriptions and unclear causality

2. Solve the problems of gaps and overlaps in coverage
BF Features – Clear Causal Descriptions

- BF describes a bug/weakness as:
  - An improper state
  - Its transition

- Improper State –
  a tuple \((\text{operation}, \text{operand}_1, \ldots, \text{operand}_n)\), where at least one element is improper

- Transition –
  the result of the operation over the operands

Initial State – caused by the Bug
- the operation is improper

Intermediate State – caused by ill-formed data
- at least one operand is improper

Final State – the Failure
- caused by a final error

Final Error
BF Features – Chaining Weaknesses

- BF describes a vulnerability as:
  - A chain of improper states and their transitions
  - States change until a failure is reached

Initial State – caused by the Bug
- the operation is improper

Intermediate State – caused by ill-formed data
- at least one operand is improper

Final State – the Failure
- caused by a final error
How to find the Bug?

- Go backwards by operand until an operation is a cause
BF Features – Converging Vulnerabilities

Improper State 1
(operation 1
operand 11 ... operand 1i ...
...)

... Improper State n
(operation n
operand n1p ...
operand nm ...
...)

Final Error

Final Error

Failure

Improper State 1'
(operation 1'
operand 1'1 ... operand 1'i ...
...)

... Improper State q'
(operation q'
... operand q'k ...
...)

Final Error'

Initial State – caused by the Bug
– the operation is improper

Intermediate State – caused by ill-formed data
– at least one operand is improper

Final State – the Failure
– caused by a final error
BF Class – a taxonomic category of a weakness type, defined by:

- A set of operations
- All valid cause → consequence relations
- A set of attributes
Example:

The BF Memory Bugs Model:

- Four phases, corresponding to the BF memory bugs classes: MAD, MAL, MUS, MDL
- Memory operations flow
BF – Defined

- BF is a ...
  - Structured
  - Complete
  - Orthogonal
  - Language independent

classification of software bugs and weaknesses
BF Example – Description of Heartbleed
Heartbleed (CVE-2014-0160)

**CVE-2014-0160** The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug.

```c
1448 dtls1_process_heartbeat(SSL *s)
1449 {
1450  unsigned char *p = &s->s3->rrec.data[0], *pl;
1451  unsigned short hbtype;
1452  unsigned int payload;
1453  unsigned int padding = 16; /* Use minimum padding */
1454  /* Read type and payload length first */
1455  hbtype = *p++;
1456  n2s(p, payload);
1457  pl = p;
1458  ...
1459  if (hbtype == TLS1_HB_REQUEST)
1460     {
1461       unsigned char *buffer, *bp;
1462     
1463       /* Allocate memory for the response, size is 1 byte
1464        * message type, plus 2 bytes payload, plus
1465        * payload, plus padding */
1466       buffer = OPENSSL_malloc(1 + 2 + payload + padding);
1467       bp = buffer;
1468      
1469      /* Enter response type, length and copy payload */
1470      *bp++ = TLS1_HB_RESPONSE;
1471      *bp++ = payload;
1472      s2n(payload, bp);
1473      memcpyp(bp, pl, payload);
1474      
1475      /* Naive implementation of memcpy */
1476      void *memcpy (void *dst, const void *src, size_t n)
1477         {
1478           size_t i;
1479           for (i=0; i<n; i++)
1480               *(char *) dst++ = *(char *) src++;
1481           return dst;
1482         }
```

**DVR (Missing Verify, Data – payload length)**

**MAD (Reposition, Pointer, Wrong Value /Size Used/ Object)**

**MUS (Read, Over Bounds Pointer, Object)**

Caused by the Bug

Caused by ill-formed data
Clear Causality in Heartbleed

- **DVR** (Missing Verify, Data – payload length)
  - Wrong Value (Size Used)

- **MAD** (Reposition, Pointer, Wrong Value /Size Used/ Object)
  - Over Bounds

- **MUS** (Read, Over Bounds Pointer, object)
  - Buffer Overflow
  - Not Cleared Object

- **Information Exposure**
  - Caused by the Bug
  - Caused by ill-formed data
  - The Failure – caused by final error(s)

The Failure – caused by final error(s)
BF – Potential Impact
BF – Potential Impacts

- Allow precise communication about software bugs and weaknesses
- Help identify exploit mitigation techniques
Questions
Questions

Irena Bojanova: irena.bojanova@nist.gov

https://samate.nist.gov/BF/