SATE IV CVE-selected

Procedure and Observations

Vadim Okun, NIST

vadim.okun@nist.gov

March 29, 2012

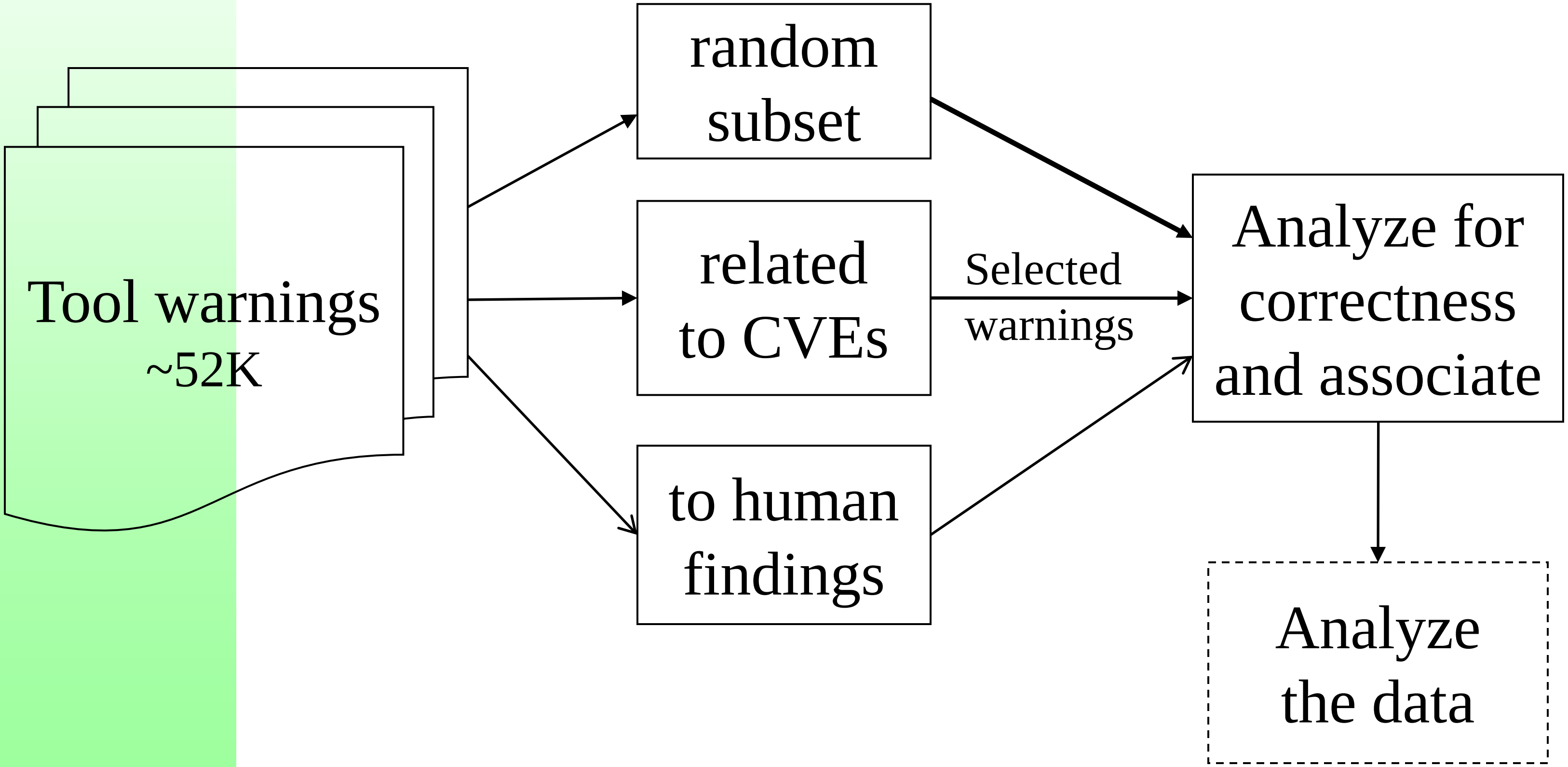
The SAMATE Project

http://samate.nist.gov/

NIST National Institute of Standards and Technology

Analysis procedure for CVE-selected test cases

Selection Methods:



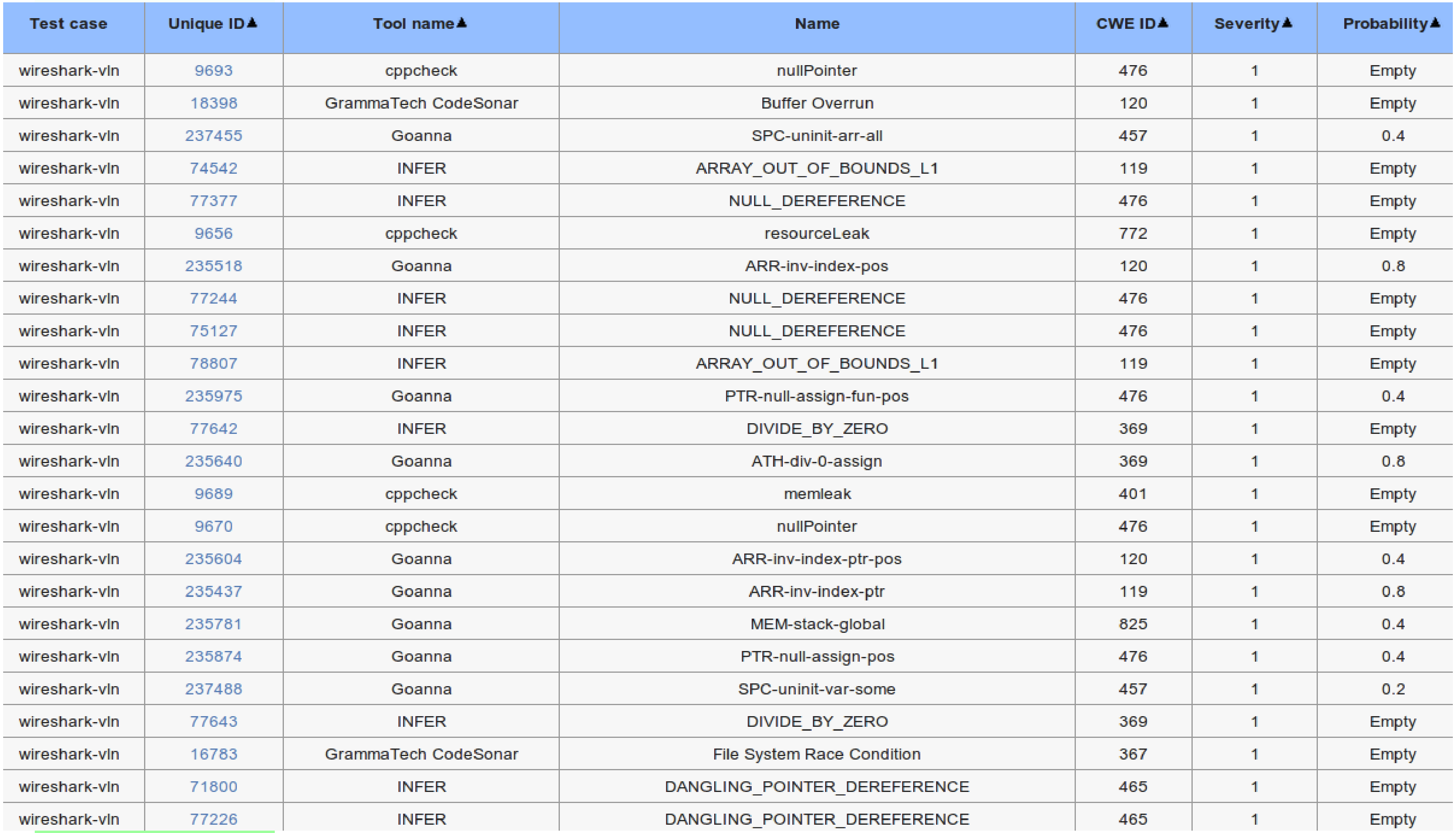
Outline

- Procedure for random subset analysis
- Observations from analysis
- Suggestions for tool improvement

Procedure for Subset Analysis

- A selected set of warnings were analyzed by experienced programmers
  - This year it was Aurelien, Vadim, and Paul

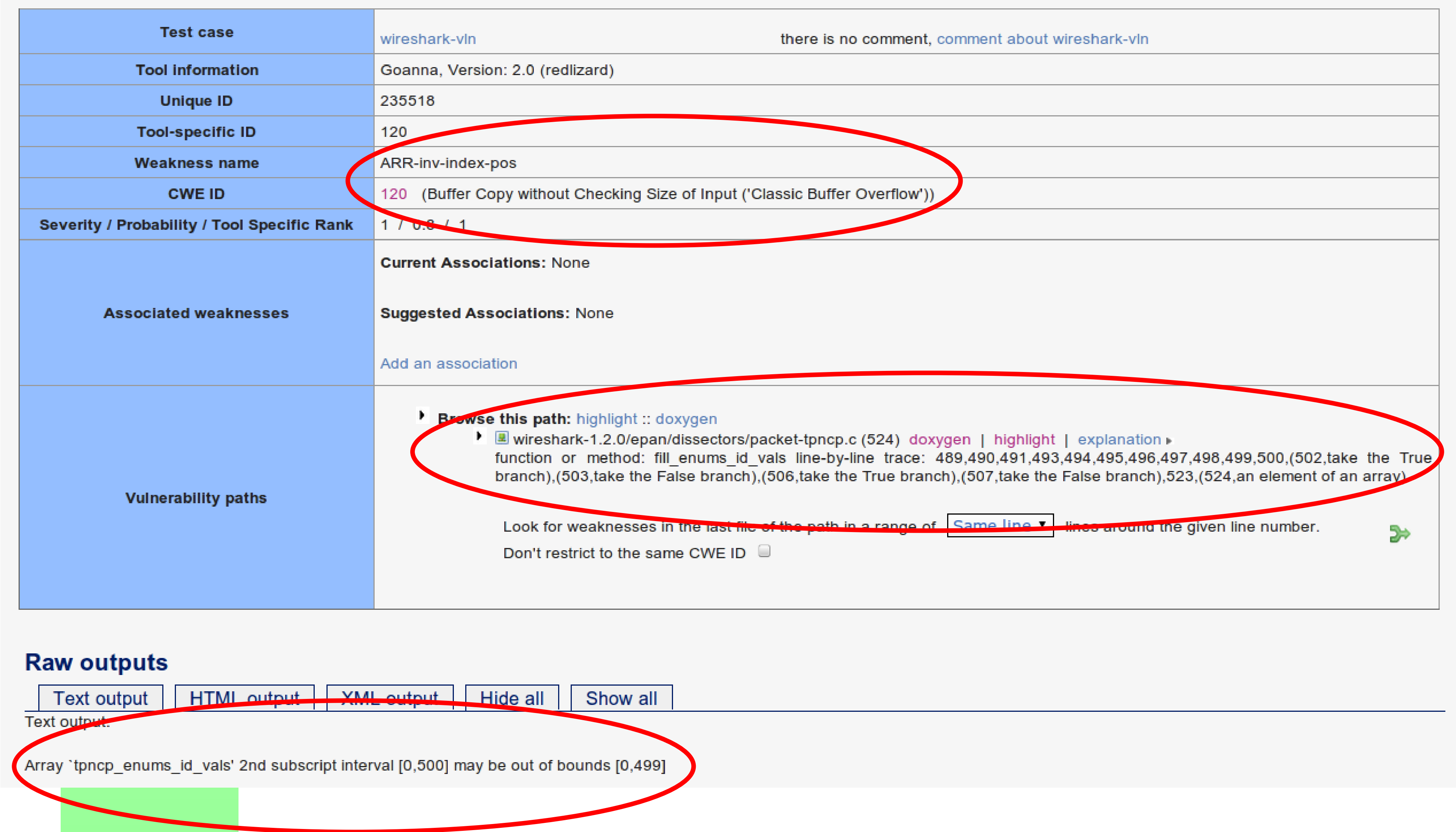
Step 1 – select a warning



Test case	Unique ID▲	Tool name▲	Name	CWE ID▲	Severity▲	Probability▲
wireshark-vln	9693	cppcheck	nullPointer	476	1	Empty
wireshark-vln	18398	GrammaTech CodeSonar	Buffer Overrun	120	1	Empty
wireshark-vln	237455	Goanna	SPC-uninit-arr-all	457	1	0.4
wireshark-vln	74542	INFER	ARRAY_OUT_OF_BOUNDS_L1	119	1	Empty
wireshark-vln	77377	INFER	NULL_DEREFERENCE	476	1	Empty
wireshark-vln	9656	cppcheck	resourceLeak	772	1	Empty
wireshark-vln	235518	Goanna	ARR-inv-index-pos	120	1	0.8
wireshark-vln	77244	INFER	NULL_DEREFERENCE	476	1	Empty
wireshark-vln	75127	INFER	NULL_DEREFERENCE	476	1	Empty
wireshark-vln	78807	INFER	ARRAY_OUT_OF_BOUNDS_L1	119	1	Empty
wireshark-vln	235975	Goanna	PTR-null-assign-fun-pos	476	1	0.4
wireshark-vln	77642	INFER	DIVIDE_BY_ZERO	369	1	Empty
wireshark-vln	235640	Goanna	ATH-div-0-assign	369	1	0.8
wireshark-vln	9689	cppcheck	memleak	401	1	Empty
wireshark-vln	9670	cppcheck	nullPointer	476	1	Empty
wireshark-vln	235604	Goanna	ARR-inv-index-ptr-pos	120	1	0.4
wireshark-vln	235437	Goanna	ARR-inv-index-ptr	119	1	0.8
wireshark-vln	235781	Goanna	MEM-stack-global	825	1	0.4
wireshark-vln	235874	Goanna	PTR-null-assign-pos	476	1	0.4
wireshark-vln	237488	Goanna	SPC-uninit-var-some	457	1	0.2
wireshark-vln	77643	INFER	DIVIDE_BY_ZERO	369	1	Empty
wireshark-vln	16783	GrammaTech CodeSonar	File System Race Condition	367	1	Empty
wireshark-vln	71800	INFER	DANGLING_POINTER_DEREFERENCE	465	1	Empty
wireshark-vln	77226	INFER	DANGLING_POINTER_DEREFERENCE	465	1	Empty

Step 2 – understand the warning

- What does it say about the code?



Test case	wireshark-vln there is no comment, comment about wireshark-vln
Tool information	Goanna, Version: 2.0 (redlizard)
Unique ID	235518
Tool-specific ID	120
Weakness name	ARR-inv-index-pos
CWE ID	120 (Buffer Copy without Checking Size of Input ('Classic Buffer Overflow'))
Severity / Probability / Tool Specific Rank	1 / 0.8 / 1
Associated weaknesses	**Current Associations:** None **Suggested Associations:** None Add an association
Vulnerability paths	▸ **Browse this path:** highlight :: doxygen ▸ wireshark-1.2.0/epan/dissectors/packet-tpncp.c (524) doxygen | highlight | explanation ▸ function or method: fill_enums_id_vals line-by-line trace: 489,490,491,493,494,495,496,497,498,499,500,(502,take the True branch),(503,take the False branch),(506,take the True branch),(507,take the False branch),523,(524,an element of an array) Look for weaknesses in the last file of the path in a range of Same line ▾ lines around the given line number. Don't restrict to the same CWE ID □

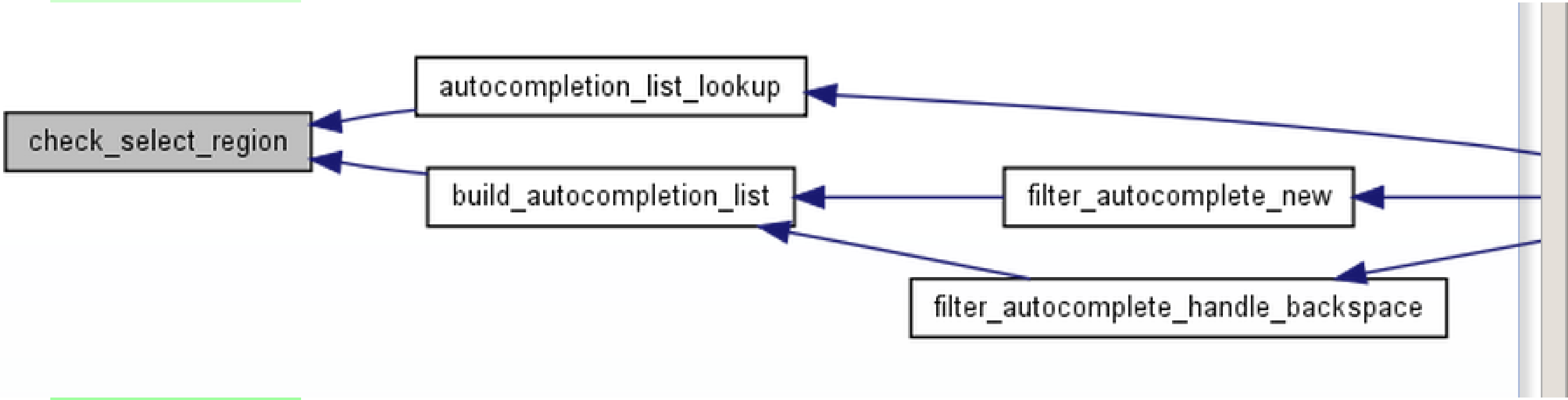
Raw outputs

Text output:

```
Array `tpncp_enums_id_vals' 2nd subscript interval [0,500] may be out of bounds [0,499]
```

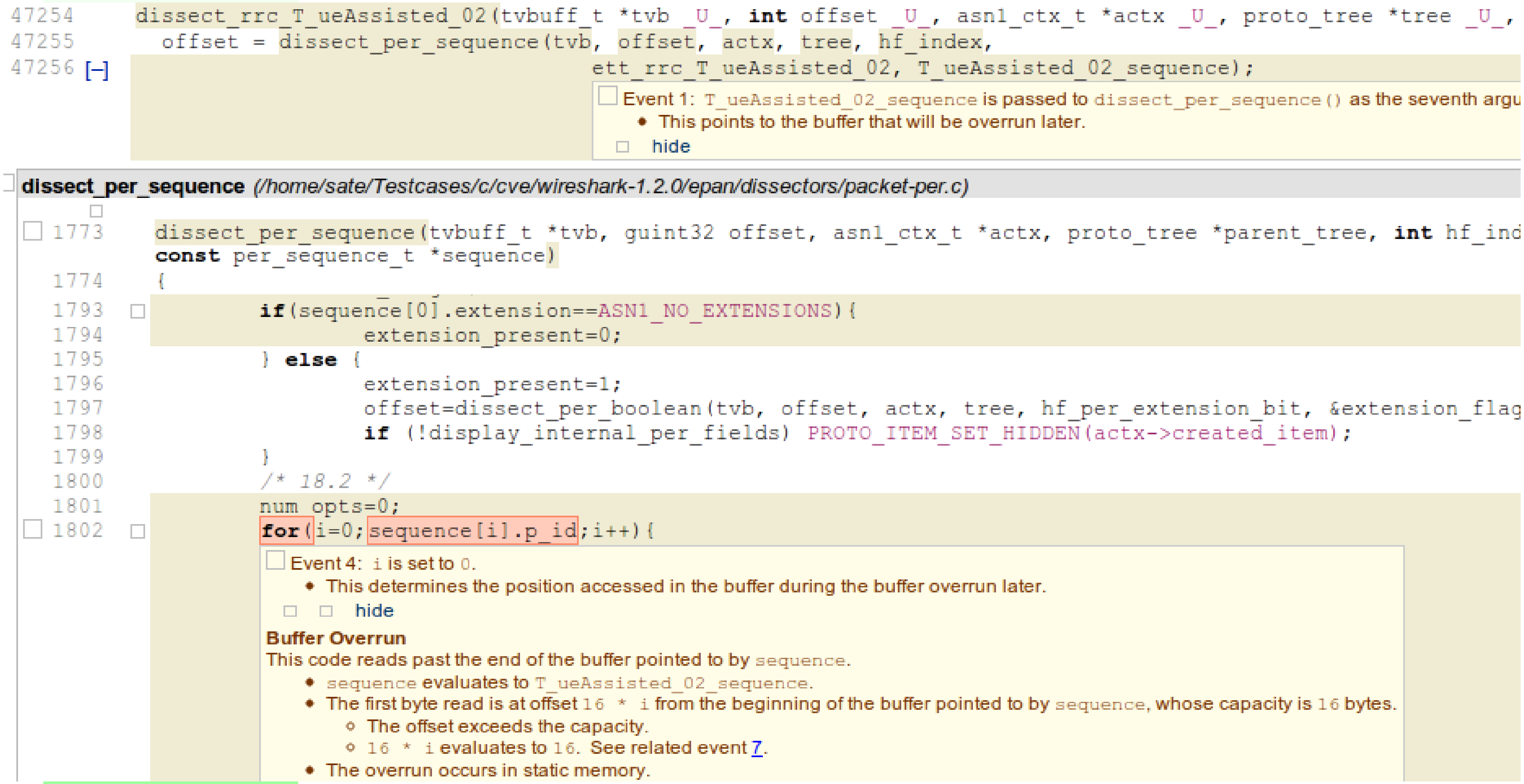
Step 3 – understand the code

- Does this happen? Could it cause problems?
- Doxygen provides call graphs and hyperlinks to functions and definitions.


check_select_region
autocompletion_list_lookup
build_autocompletion_list
filter_autocomplete_new
filter_autocomplete_handle_backspace

Step 3 – understand the code

- Original tool output has a lot of information and splices code to show control flow.



```
dissect_rrc_T_ueAssisted_02(tvbuff_t *tvb _U_, int offset _U_, asn1_ctx_t *actx _U_, proto_tree *tree _U_,
 offset = dissect_per_sequence(tvb, offset, actx, tree, hf_index,
                                                     ett_rrc_T_ueAssisted_02, T_ueAssisted_02_sequence);
```

□ Event 1: `T_ueAssisted_02_sequence` is passed to `dissect_per_sequence()` as the seventh argu

- This points to the buffer that will be overrun later.

□ hide

dissect_per_sequence *(/home/sate/Testcases/c/cve/wireshark-1.2.0/epan/dissectors/packet-per.c)*

```
dissect_per_sequence(tvbuff_t *tvb, guint32 offset, asn1_ctx_t *actx, proto_tree *parent_tree, int hf_ind
const per_sequence_t *sequence)
{
        if(sequence[0].extension==ASN1_NO_EXTENSIONS){
                extension_present=0;
        } else {
                extension_present=1;
                offset=dissect_per_boolean(tvb, offset, actx, tree, hf_per_extension_bit, &extension_flag
                if (!display_internal_per_fields) PROTO_ITEM_SET_HIDDEN(actx->created_item);
        }
        /* 18.2 */
        num_opts=0;
        for(i=0;sequence[i].p_id;i++){
```

□ Event 4: `i` is set to `0`.

- This determines the position accessed in the buffer during the buffer overrun later.

□ □ hide

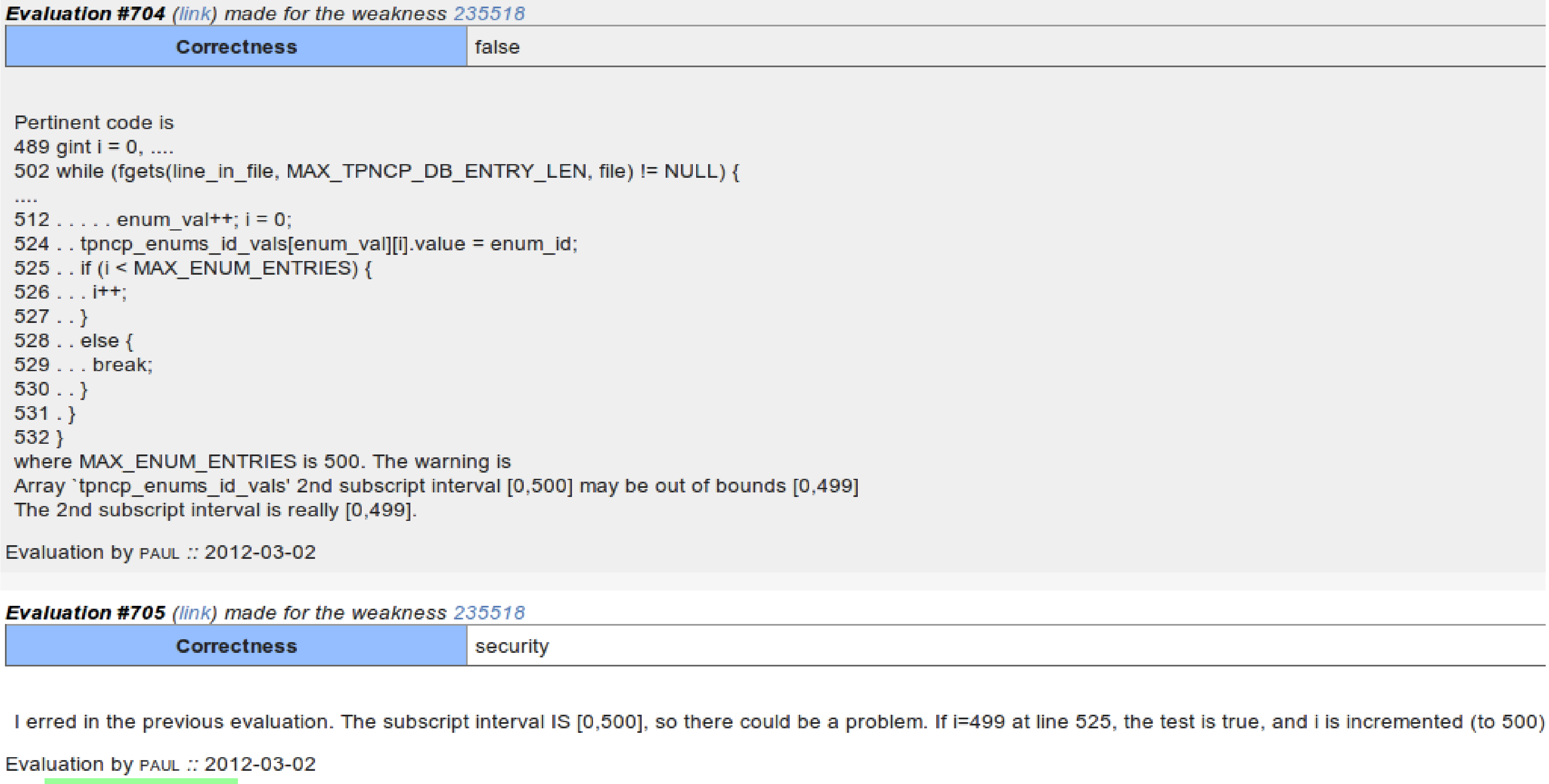
Buffer Overrun

This code reads past the end of the buffer pointed to by `sequence`.

- `sequence` evaluates to `T_ueAssisted_02_sequence`.
- The first byte read is at offset `16 * i` from the beginning of the buffer pointed to by `sequence`, whose capacity is `16` bytes.
  - The offset exceeds the capacity.
  - `16 * i` evaluates to `16`. See related event 7.
- The overrun occurs in static memory.

Step 4 – write an evaluation

- Include code snippets and reasoning so others can critique it



Evaluation #704 *(link) made for the weakness 235518*

Correctness	false

Pertinent code is

```
489 gint i = 0, ....
502 while (fgets(line_in_file, MAX_TPNCP_DB_ENTRY_LEN, file) != NULL) {
....
512 . . . . . enum_val++; i = 0;
524 . . tpncp_enums_id_vals[enum_val][i].value = enum_id;
525 . . if (i < MAX_ENUM_ENTRIES) {
526 . . . i++;
527 . . }
528 . . else {
529 . . . break;
530 . . }
531 . }
532 }
```

where MAX_ENUM_ENTRIES is 500. The warning is
Array `tpncp_enums_id_vals' 2nd subscript interval [0,500] may be out of bounds [0,499]
The 2nd subscript interval is really [0,499].

Evaluation by PAUL :: 2012-03-02

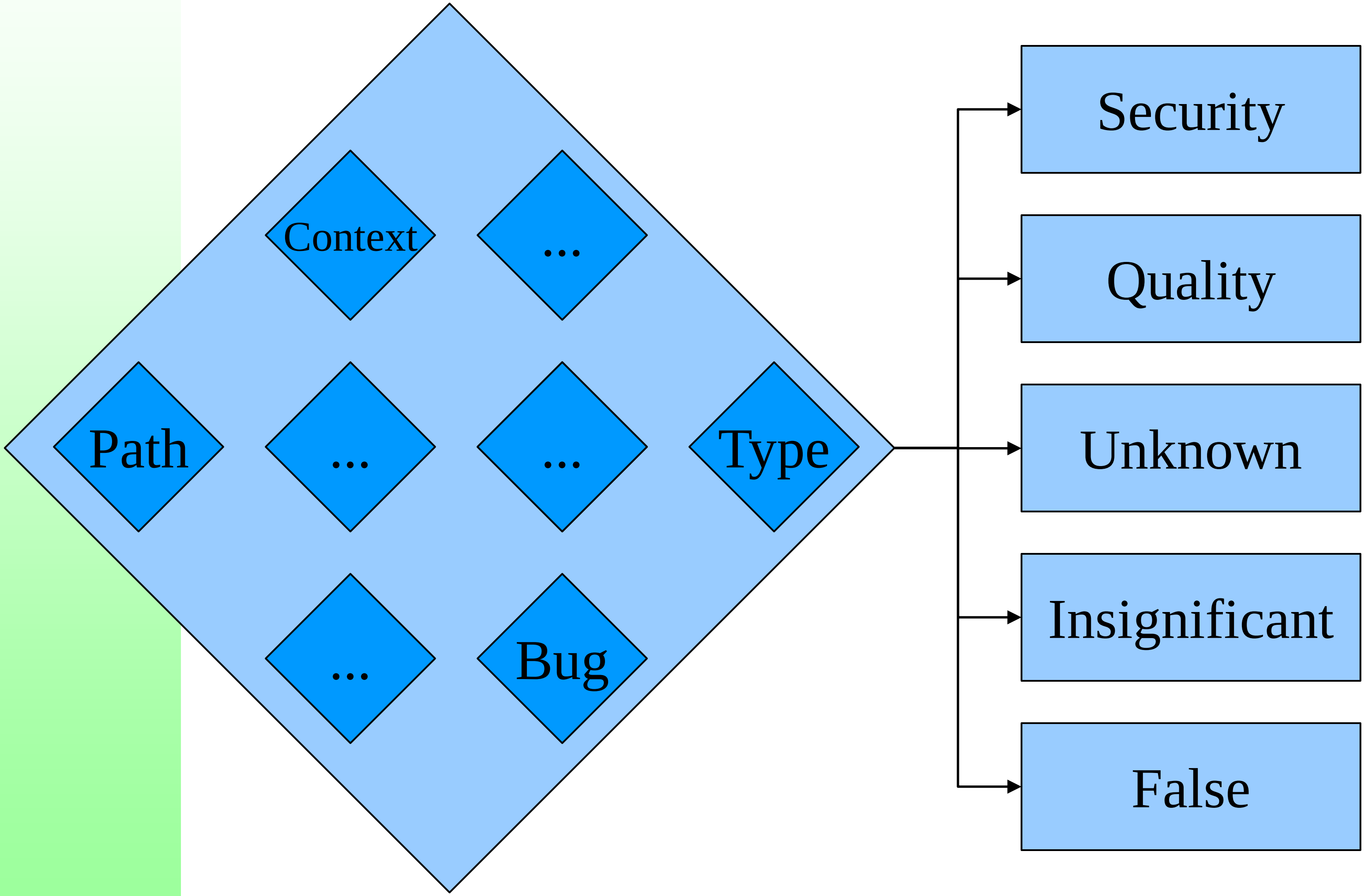
Evaluation #705 *(link) made for the weakness 235518*

Correctness	security

I erred in the previous evaluation. The subscript interval IS [0,500], so there could be a problem. If i=499 at line 525, the test is true, and i is incremented (to 500)

Evaluation by PAUL :: 2012-03-02

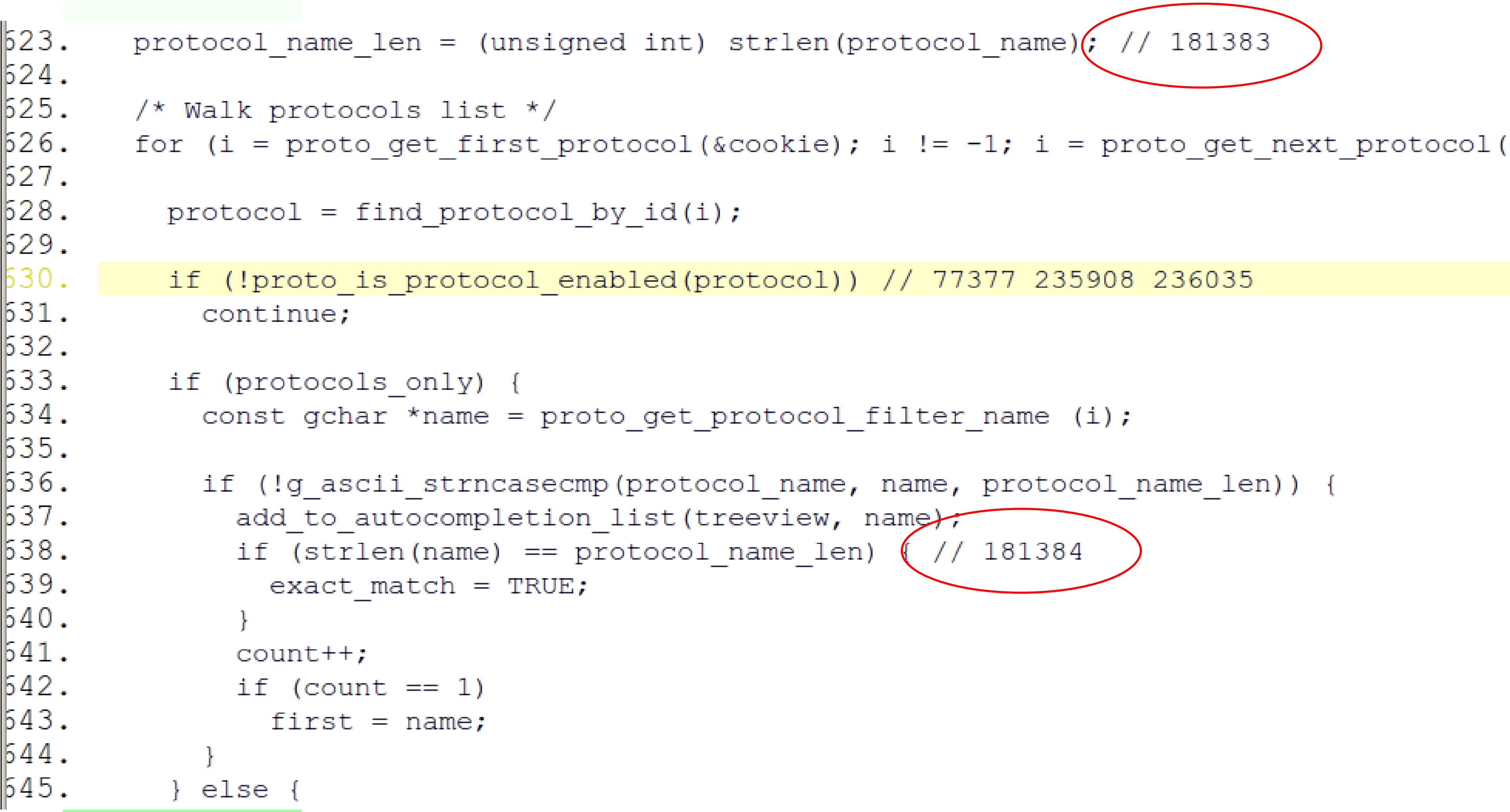
Decision process



Step 4b – alert developers

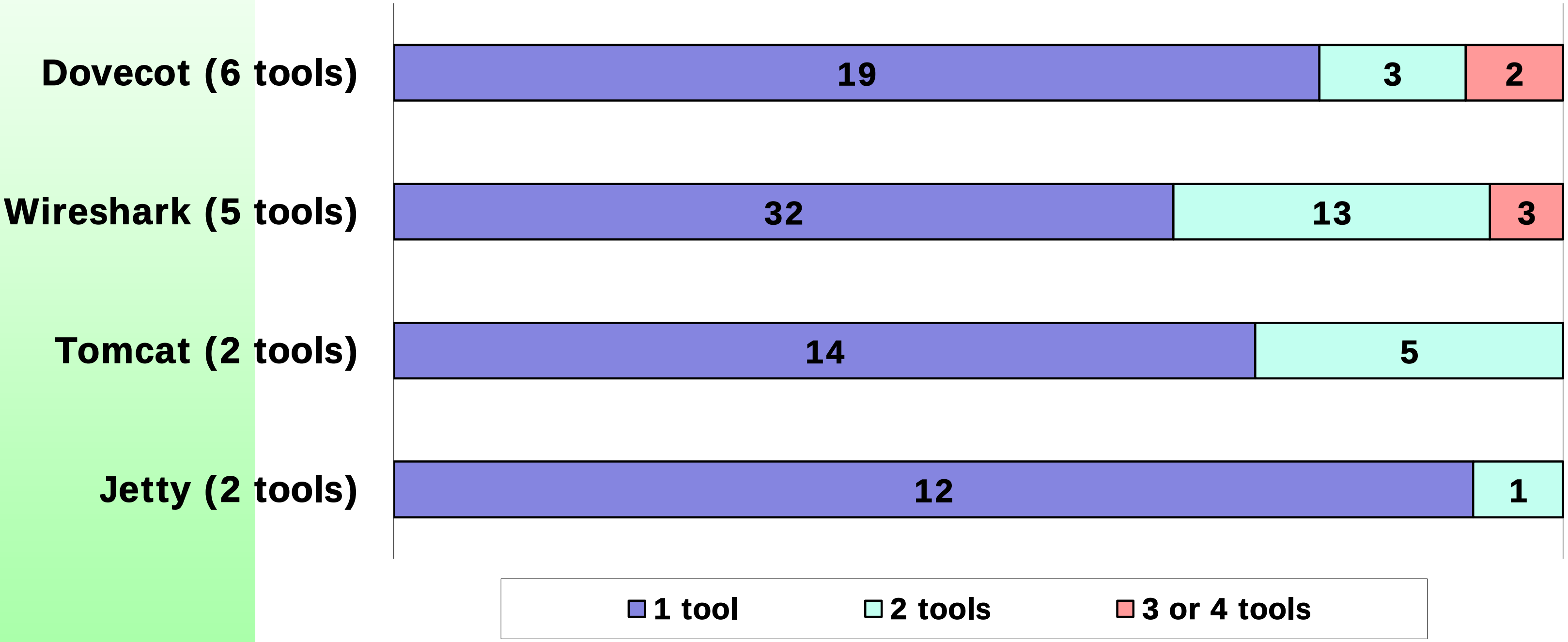
- If there is clearly an error
  - and it is easily fixed or high impact
  - and it exists in the current version,
- tell the developers

Step 5 – associate other warnings



```
protocol_name_len = (unsigned int) strlen(protocol_name); // 181383

/* Walk protocols list */
for (i = proto_get_first_protocol(&cookie); i != -1; i = proto_get_next_protocol(

  protocol = find_protocol_by_id(i);

  if (!proto_is_protocol_enabled(protocol)) // 77377 235908 236035
    continue;

  if (protocols_only) {
    const gchar *name = proto_get_protocol_filter_name (i);

    if (!g_ascii_strncasecmp(protocol_name, name, protocol_name_len)) {
      add_to_autocompletion_list(treeview, name);
      if (strlen(name) == protocol_name_len) { // 181384
        exact_match = TRUE;
      }
      count++;
      if (count == 1)
        first = name;
    }
  } else {
```

Overlap for true quality/security

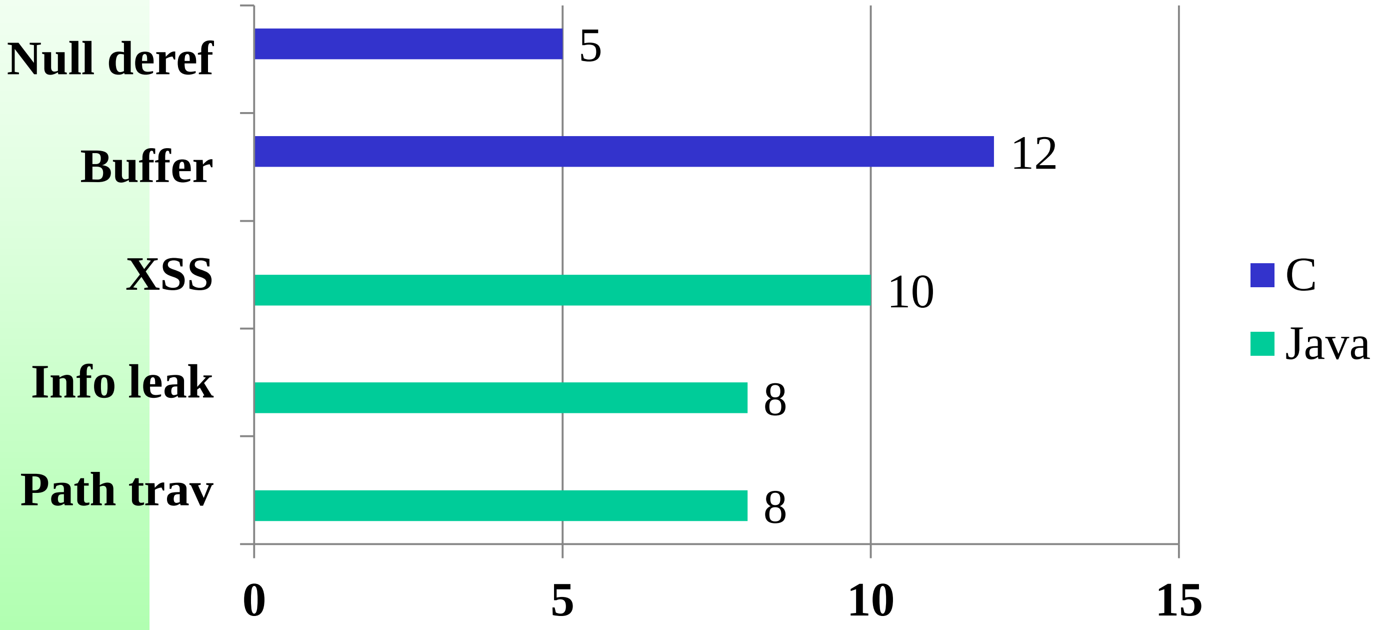


More overlap for some weakness categories

CVEs

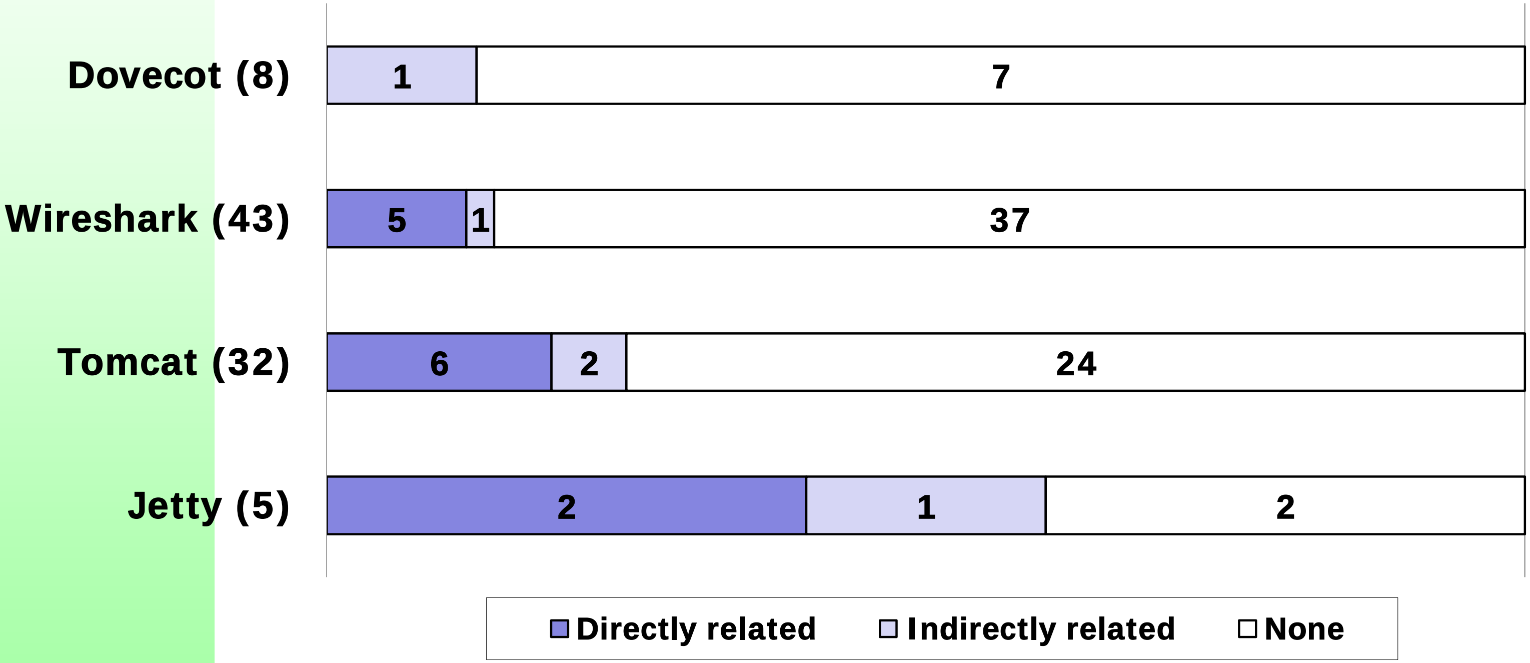
- Real-life vulnerabilities
- 88 CVEs in the 4 test cases
  - Identify source, sink or path locations
  - Match to tool warnings

Top 5 CWEs for CVEs



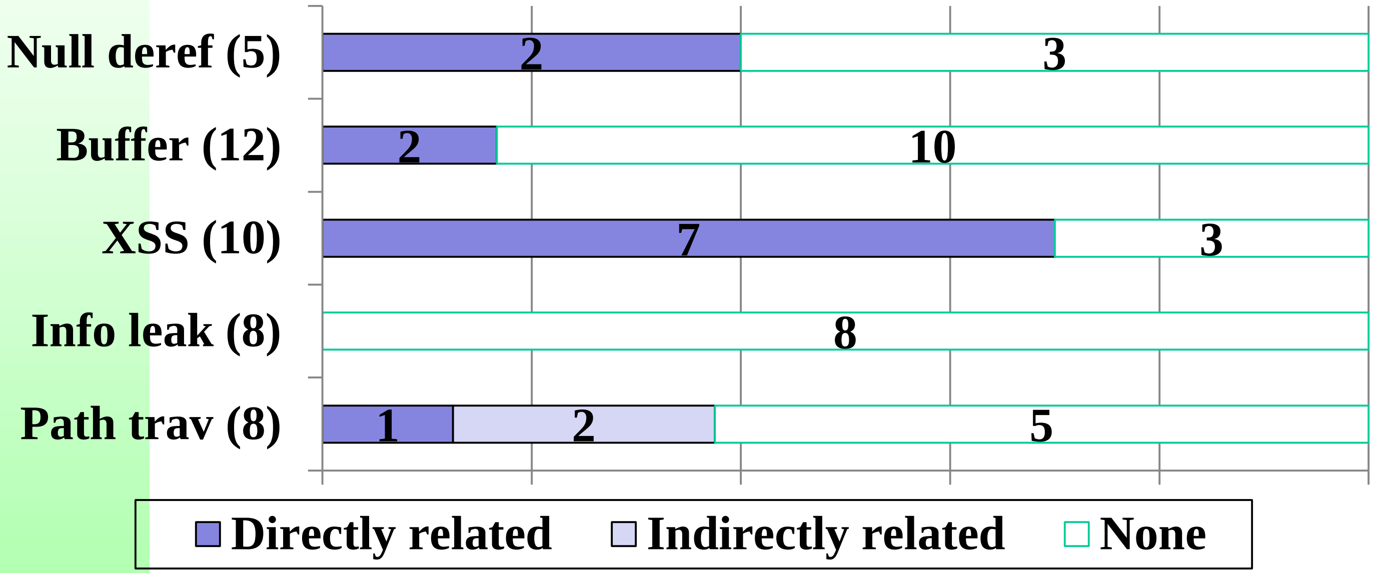
- Top CWEs cover 43 of 88 CVEs
- A total of 30 different CWE ids
- Many design flaws

Related warnings from tools



- CVEs described better than in SATE 2010

Related Warnings for Top 5 CWEs



- Related warnings from tools for 8 CWEs

CVE-2006-7195 Not Found

- JSP Standard Tag Library (JSTL)
  `<td>${header["host"]}</td>`
- Should understand popular libraries and frameworks

On discrimination

- Reporting a weakness when there is one
- Keeping quiet when there is none

- Varies a lot by tool and weakness category

CVE-2009-3550 Found

Vulnerable version:

```
1314 item = item -> parent;

1318     item = item -> parent;
```

Tool warning: pointer item last assigned on line 1314 could be NULL and is dereferenced at line 1318

CVE-2009-3550 Found

Fixed version:

```
#define GET_ITEM_PARENT(x) \
        ((x->parent!=NULL)?x->parent:x)

item = GET_ITEM_PARENT(item);

    item = GET_ITEM_PARENT(item);
```

No tool warning here. Perfect!

CVE-2006-7196 / 2009-0781

Vulnerable version:

```
String role = request.getParameter(“role”);
…
<%= role %>
```

Reported

CVE-2006-7196 / 2009-0781
Not discriminated

Fixed version:

```
String role = request.getParameter(“role”);
…
<%= filter(role) %>
```

Reported anyway

- Plenty of much more complex cases

Human Analysis

- Wireshark dissectors are protocol decoders
- Chose Intelligent Platform Management Interface (IPMI) dissector for analysis
  - Fuzzing
  - Manual source code review

Human Analysis Results

- Buffer overrun in vulnerable version
- Corrected in fixed version
- Corresponds to CVE-2009-2559

CVE-2009-2559 Not Found

tsel declared with size 4

```
static const int *tsel[] = { &ett_ipmi_se_XX_b1,
&ett_ipmi_se_XX_b2, &ett_ipmi_se_XX_b3, &ett_ipmi_se_XX_b4 };

for (i = 0; offs < len; i++, offs++) {

       s_tree = proto_item_add_subtree(ti, *tsel[i]);
```

i is not checked and goes out of bounds

- Tools routinely find such weaknesses. Why not here?
- Did tools find/analyze the code?

Summary

- Find and analyze more code
- Better discrimination
- Better understand libraries and frameworks
- Participate in future SATEs ☺