Bugs that Matter: True Positives and False Negatives in CodeSonar ®

SATE 2010

Presented by:

Paul Anderson, VP of Engineering, GrammaTech Inc.

GrammaTech, Inc.
317 N Aurora St.
Ithaca, NY 14850
Tel: 607-273-7340
E-mail: info@grammatech.com

Agenda

- About CodeSonar
- Dovecot results
- Wireshark results
- Conclusions

About CodeSonar ®

- Languages: C, C++
- 93 warning classes, mostly serious bugs, some coding standards checking
  - › API provided for domain-specific checks
- Web-client hosted UI
- Precise parse
- Whole Program
- Path Sensitive
- Object Sensitive

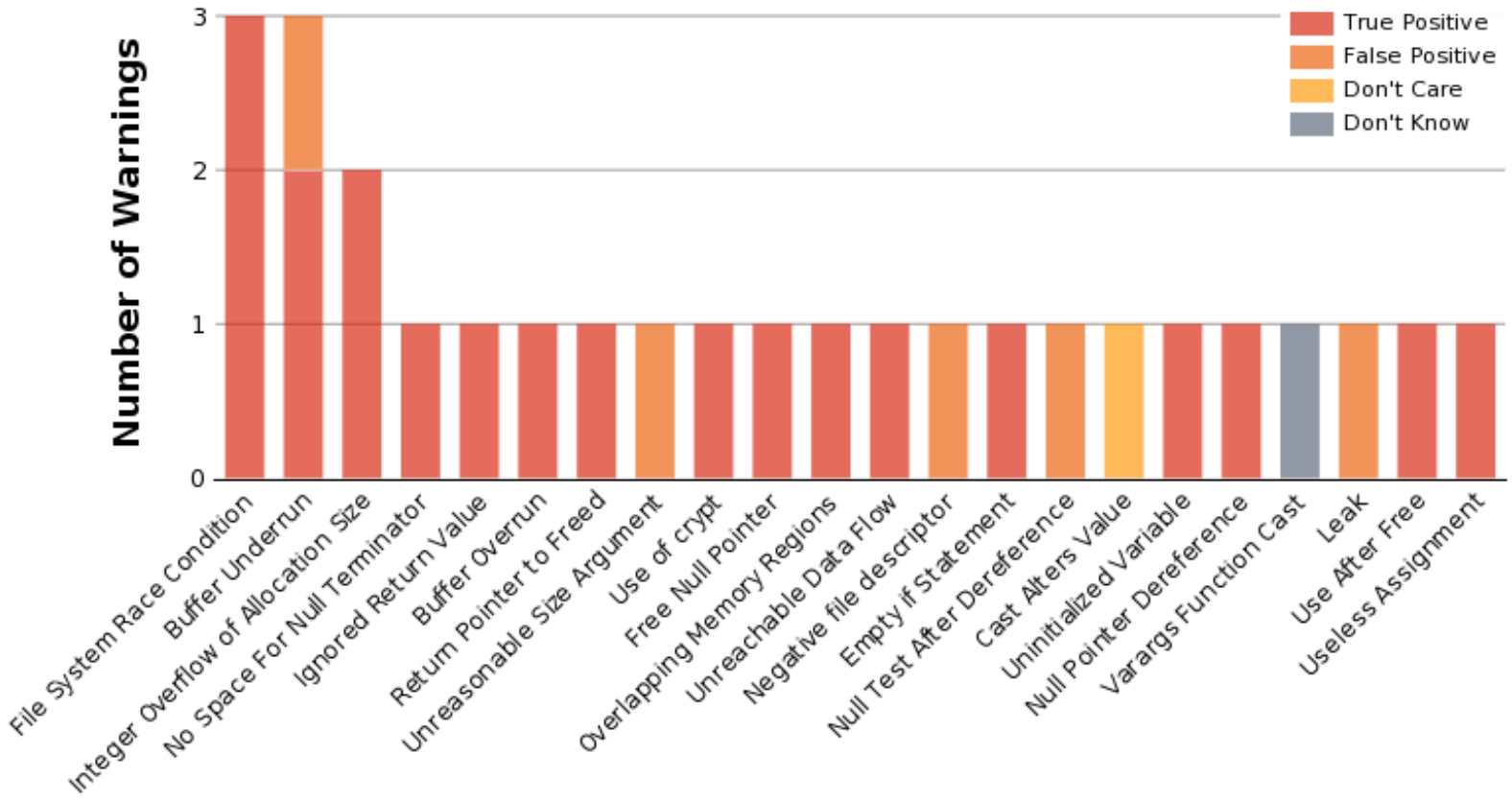
Recommended Usage for CodeSonar ®

- Run with default options
- Sample the results
  - › Look for parse errors, sources of false positives, undefined functions, etc.
- Tweak configuration parameters
  - › Maybe write models for undefined functions
- Rinse and repeat

Dovecot results

- Initial run yielded false positives, caused by:
  - CodeSonar did not know that these functions terminate execution:
    - i_fatal(), i_fatal_status(), i_panic(), test_assert_failed()
  - Memory allocation models needed:
    - i_malloc(), i_realloc(), etc.
  - Undefined functions
    - E.g., PAM, xdr, BZ2.
- Final run
  - SATE team chose 30 results
  - We judged: 8 False Positives, 1 Don't Care, 1 Don't Know, 22 True Positives

Dovecot Sample Summary



Are these “Bugs that Matter?”

Wireshark Results

- Analyses for both versions completed but not submitted
  - › No time available to configure and tune the analysis ☹
- Missing source (> 700 functions):
  - › glib, gdk, gtk, gzip, pango, pcre
- CodeSonar found 4/17 CVEs

True Positive: CVE-2009-4376

- **Description:**
  Buffer overflow in the daintree_sna_read function in the Daintree SNA file parser in Wireshark 1.2.0 through 1.2.4 allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via a crafted packet.

- Fix:

```
-  if (sscanf(seekLine, "%*s %*u.%*u %*u %s", seekData) != 1) {

+  if (sscanf(seekLine, "%*s %*u.%*u %*u %"
+       SEEKDATA_MAX_FIELD_SIZE "s", seekData) != 1) {
```

True Positive: CVE-2009-3243

- **Description:**
  Unspecified vulnerability in the TLS dissector in Wireshark 1.2.0 and 1.2.1, when running on Windows, allows remote attackers to cause a denial of service (application crash) via unknown vectors related to TLS 1.2 conversations.
- Fix:

```
  const gchar* ssl_version_short_names[] = {
        "SSL",
        "SSLv2",
        "SSLv3",
        "TLSv1",
        "TLSv1.1",
        "DTLSv1.0",
        "PCT",
+       "TLSv1.2"
  };
```

- 8 buffer overruns found by CodeSonar related to this

True Positive: CVE-2009-4377

- **Description:**
  The (1) SMB and (2) SMB2 dissectors in Wireshark 0.9.0 through 1.2.4 allow remote attackers to cause a denial of service (crash) via a crafted packet that triggers a NULL pointer dereference, as demonstrated by fuzz-2009-12-07-11141.pcap.
- Fix:

```
-    nti=sip->extra_info;

+    if (sip->extra_info_type == SMB_EI_NTI) {
+                nti=sip->extra_info;
+           }
```

- CodeSonar reports null pointer dereference on nti in a different location

False Negative: CVE-2010-2285

- **Description:**
  The SMB PIPE dissector in Wireshark 0.8.20 through 1.0.13 and 1.2.0 through 1.2.8 allows remote attackers to cause a denial of service (NULL pointer dereference) via unknown vectors.
- Cause: A call to g_vsnprintf() is passed a NULL in a position where a string is expected.
- Fix:

```
-    string);
+    string ? string : "(null)");
```

- Conclusion: model for g_vsnprintf() would likely have caused this to be found.
  - Also applies to **CVE-2010-1455**

False Negative: CVE-2010-0304

- **Description:**
  Multiple buffer overflows in the LWRES dissector in Wireshark 0.9.15 through 1.0.10 and 1.2.0 through 1.2.5 allow remote attackers to cause a denial of service (crash) via a malformed packet, as demonstrated using a stack-based buffer overflow to the dissect_getaddrsbyname_request function.
- Buffer overrun involved an indirect function call (not modeled by CodeSonar)
- Fix was a multiple-line rewrite
- Conclusion: indirect function call handling may help

False Negative: CVE-2010-2284

- **Description:**
  Buffer overflow in the ASN.1 BER dissector in Wireshark 0.10.13 through 1.0.13 and 1.2.0 through 1.2.8 has unknown impact and remote attack vectors. J. Oquendo discovered that the ASN.1 BER dissector could overrun the stack.

- Fix: “Make get_ber_length() iterative instead of recursive so we don't overrun the stack.”

- Conclusion: out of scope for CodeSonar

False Negative: CVE-2010-2286

- **Description**:
  The SigComp Universal Decompressor Virtual Machine dissector in Wireshark 0.10.7 through 1.0.13 and 1.2.0 through 1.2.8 allows remote attackers to cause a denial of service (infinite loop) via unknown vectors.
- **Bug fix:**

```
+       used_udvm_cycles++;
        current_instruction = buff[current_address];
        switch ( current_instruction ) {
        case SIGCOMP_INSTR_DECOMPRESSION_FAILURE:
-               used_udvm_cycles++;
                if ( result_code == 0 )
                        result_code = 9;
                proto_tree_add_text(udvm_tree, bytecode_tvb, 0, -1,
```

  - Other increments of used_udvm_cycles removed from other cases

CVE-2010-2286 continued...

- Note: loop termination condition does not mention used_udvm_cycles
- Loop termination controlled as follows:

```
execute_next_instruction:
     if ( used_udvm_cycles > maximum_UDVM_cycles ){
             result_code = 15;
            goto decompression_failure;
     }
     current_instruction = buff[current_address];
     switch ( current_instruction ) {
```

- Conclusion: smarter infinite loop detection might help

False Negative: CVE-2009-3551

- **Description:**
  Off-by-one error in the dissect_negprot_response function in packet-smb.c in the SMB dissector in Wireshark 1.2.0 through 1.2.2 allows remote attackers to cause a denial of service (application crash) via a file that records a malformed packet trace.
- **Bug Fix:**

```
      if (si->sip && si->sip->extra_info_type==SMB_EI_DIALECTS) {
              dialects = si->sip->extra_info;
-             if (dialect <= dialects->num) {
+             if (dialect < dialects->num) {
                      dialect_name = dialects->name[dialect];
              }
      }
```

- Conclusion: unclear

SATE Experience

- I wish we had more time and resources
- Having real bugs that matter is extremely useful