Infoblox
CONTROL YOUR NETWORK

Security Architecture of BIND 9

神明達哉

May 20, 2016

Background

- ISC BIND
  - Berkeley Internet Name Domain
  - Most widely-used DNS server implementation to date
  - Open source, BSD-compatible license
  - Developed and maintained by Internet Systems Consortium (ISC)
    - http://www.isc.org/
  - BIND 9: Current major version, main topic of this talk
    - Prior versions: BIND 4, BIND 8
- About me
  - Previous: “researcher” at Toshiba
    - MPLS(-like), IPv6, DNS (for IPv6), DHCPv6
  - Ex-ISC employee
    - BIND 9 statistics, port randomization, performance enhancements; BIND 10
  - Current: engineer at Infoblox, Inc.
    - (but this talk is not about my current job)

Some History of BIND Security

- Lots of critical bugs in BIND 8
  - 16 buffer overflow/overrun bugs in changelog of BIND 8.4.7
  - Some could lead to arbitrary code execution
- BIND 9
  - Complete redesign and rewrite, no reuse of BIND 8 code
  - Improving security was one major goal

Security Considerations in BIND 9

- Avoid handling raw wire/text data as much as possible
  - Limit modules that touch raw data
  - Other modules access it through higher-level abstractions
- Avoid using error-prone language primitives
  - E.g. use buffer abstraction instead of direct manipulation of C-string/array
- Modularity
  - Per-file data hiding with opaque data structures
  - Minimize the risk of inter-module data corruption
- Design by contract (DBC)
  - Separate invalid conditions (caller's bug) from valid special cases
  - Treat the former with an assertion failure and core dumping
  - Help simplify implementation
  - Help avoid invalid operations
  - (Unfortunately) DoS attack vector

A Classic Example of Vulnerable Code

- Scenario: copy a string from a remote source to an internal buffer
  - (excluding a trailing nul for simplicity)
- Naively assume that the given string is short enough
  - Recipe for arbitrary code execution

```
void copy_text(const char *src, char *dst)
{
  while (*src != '\0')
    *dst++ = *src++;
}
```


Commonly Adopted Practice

- Pass size for any variable-length data
- Validate input, treat violation as an error
- Nothing wrong, but:
  - Tend to make code less understandable
    - Mix of caller's bug and actual run time error; lengthy error handling
  - A violation can result in an undefined behavior
    - Difficult to know how wrong it is

```
int copy_text(const char *src, size_t srclen,
              char *dst, size_t dstlen)
{
  if (srclen > dstlen)
    return -1; // intending to return an error
  while (*src != '\0')
    *dst++ = *src++;
  return 0; // intending to return success
}
```


DBC + Assertion: BIND 9-Way

- Clarify assumptions as a contract, handle violation with assert(3)
- Make code simpler and easier to understand/debug
- Yet safer
  - In that the worst case for assumption violation is an assertion failure
- But the “worst case” can be a DoS vector

```
void copy_text(const char *src, size_t srclen,
               char *dst, size_t dstlen)
{
  // caller is responsible for validation/pass valid data
  REQUIRE(dstlen >= srclen); // abort if this fails
  while (*src != '\0')
    *dst++ = *src++;
}
```


A Closer Look at BIND 9's Security Fixes

- 65 "security" fixes as of 9.10.3-P4

(example)

```
3861.[security]   Missing isc_buffer_availablelength check results
                  in a REQUIRE assertion when printing out a packet
                  (CVE-2014-3859).  [RT #36078]
```

- 28(+1) are assertion failure conditions
  - With varying remote exploitability
- 2 others are other types of DoS vulnerability (memory leak, inf loop)
- Others include: ACL bugs, DNSSEC validation bugs, Cache logic bugs, OpenSSL version bumps, improving random numbers, etc.
  - Some are not vulnerability
- Apparently no buffer overrun/code execution type of vulnerability
  - Except one in libbind (essentially a BIND 8 bug)

Hindsight: Things That Did not Go Well

- Monolithic design
  - Auth/Recursive/DDNS etc. in a single box
  - Fate Sharing
- Inter-module dependency
  - Making the code difficult to unit test
- Complicated architecture
  - Mix of thread/non-thread modes, etc
  - Hard to understand and maintain
- DBC/Assertion as a DoS vector
  - Should still be much better than code execution-kind, but certainly unacceptable in critical operation
  - Poorer testability left more contract violation cases open than expected

Aside: Dreams in BIND 10

- Aimed to achieve clarity, safety and robustness
  - Written in C++ instead of C
    - Much type safer, many efficient and high-level utilities, much easier to unit-test
  - Use exception instead of assertion
    - Keep code concise and clear, yet allow catch exceptions and recover gracefully
  - Automatic restart
    - Can't help in continued DoS, but still improve robustness for rare failures
  - Multi-process model
    - Avoid fate sharing
- Unfortunately project failed
  - See Shane Kerr's presentation at RIPE 68

Handling Security Issues at ISC

- Examine possible issues to determine if it's a security bug
  - Crash reports at users mailing list, individual bug reports, suspicious results from tests/static analysis, internal review, etc.
  - Quite strict: tend to treat it as a security matter for various levels of severity
- Communication via HTTPS or PGP-encrypted email
- Fix and test internally
- Phased disclosure
- Documentation
  - Description, Impact, Score, Workaround, Patch information
- Separate patch releases
  - BIND 9.x.y-Pn

