Pinpoint report

Project Name:	ton-2019-09-27-19-53-05
First Analysis Time:	2019-09-27 19:52:49
Latest Analysis Time:	2019-09-27 19:58:10
Range:	Single Analysis
Analysis Time:	2019-09-27 19:58:10
Issue Reviews:	Confirmed

Table of contents

1 Statistics

2 Issues by Types

2.0 Divide by zero

2.1 NULL pointer dereference

2.2 Return stack address

2.3 Incorrectly specified parameters in function calls

2.4 Weak Encrypt

2.5 Realloc buffer without clear

Overview

★ ★ ★

Level

441874

Actual lines of code

13456

Issues found this time

13456

Latest discovered issues

0

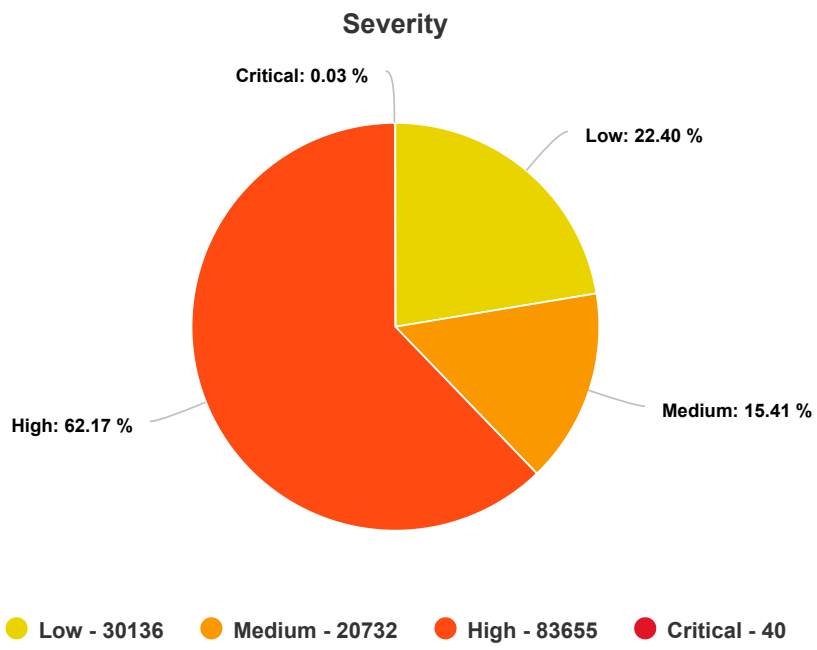
Latest fixed issues

3

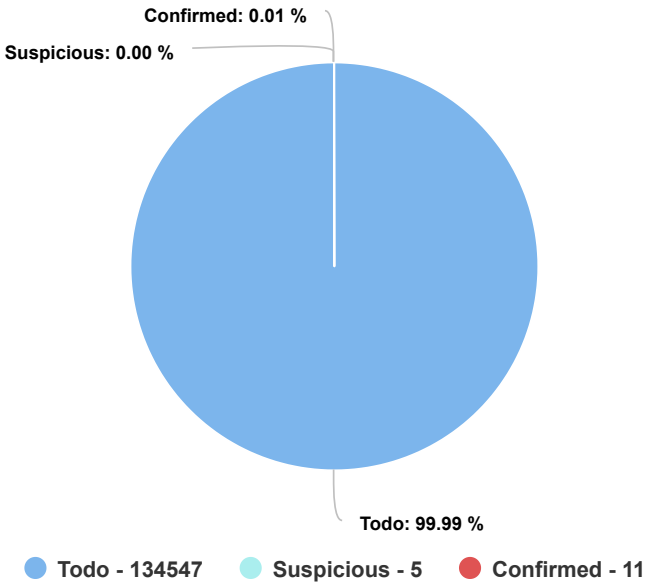
Issues found this time

3

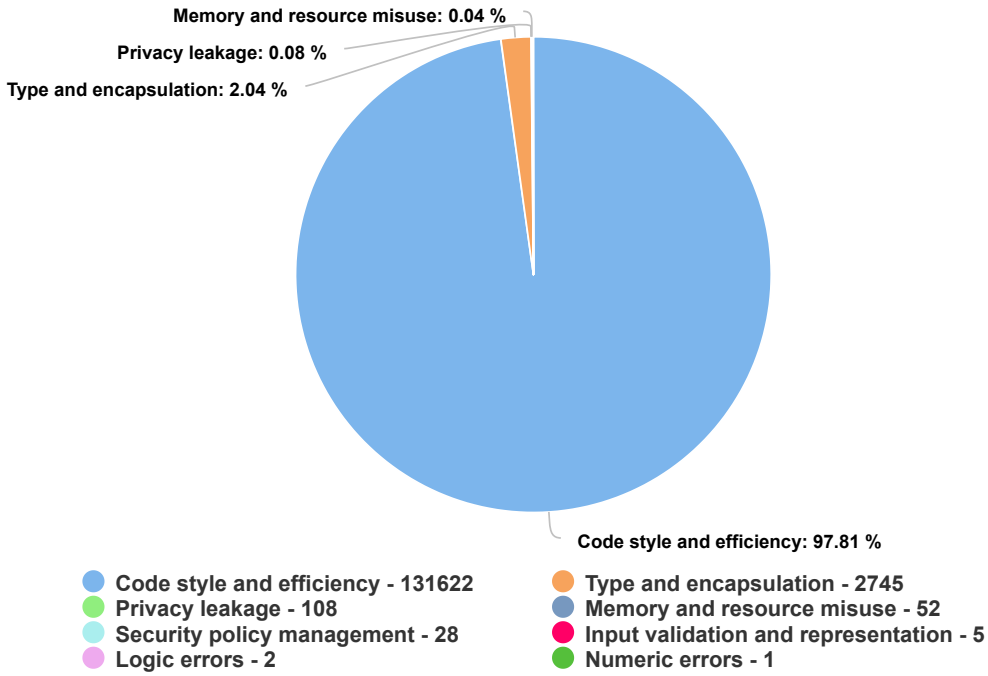
Latest discovered issues



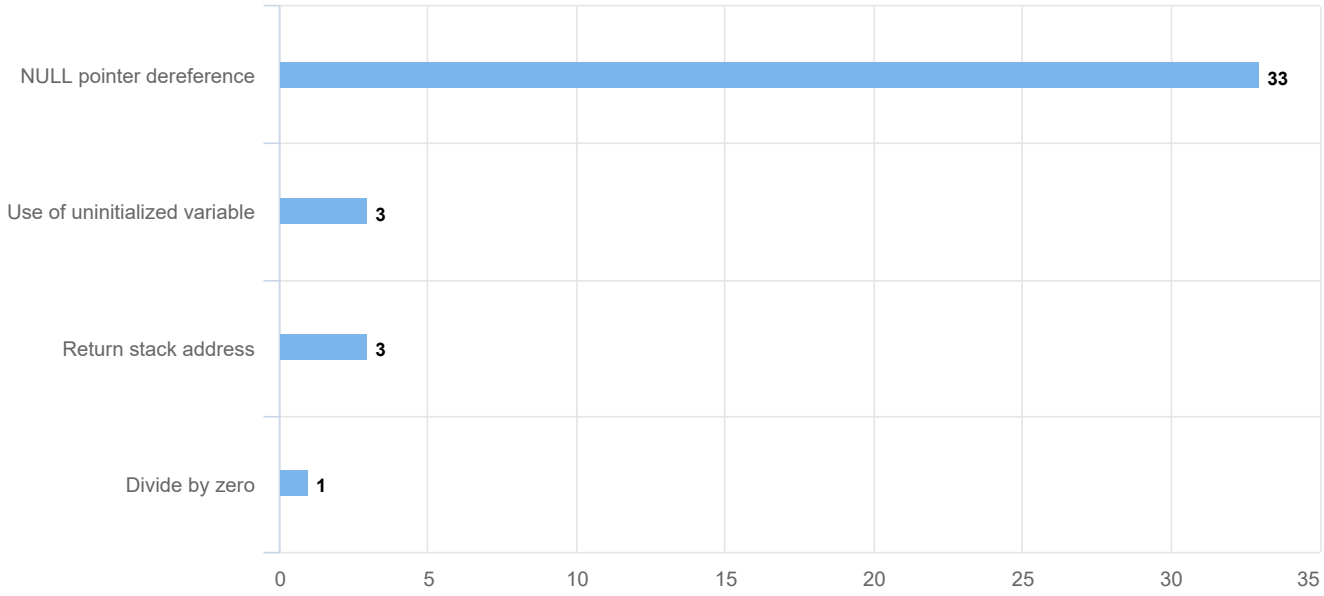
Review



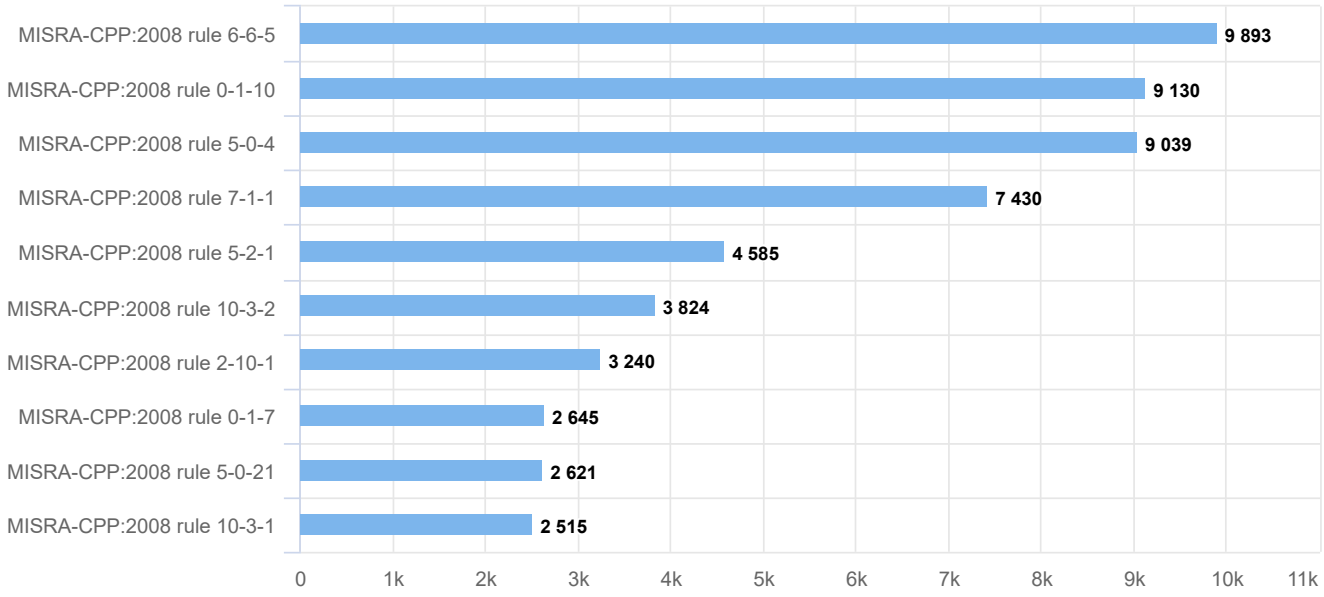
Category



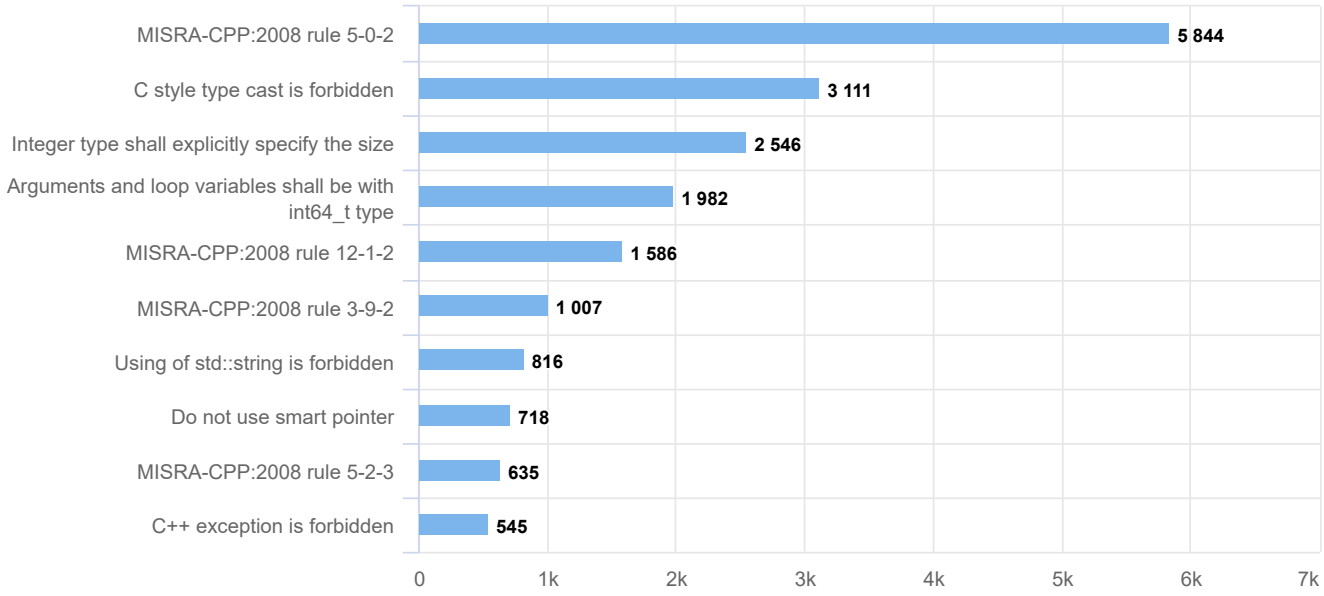
Top 10 critical issues



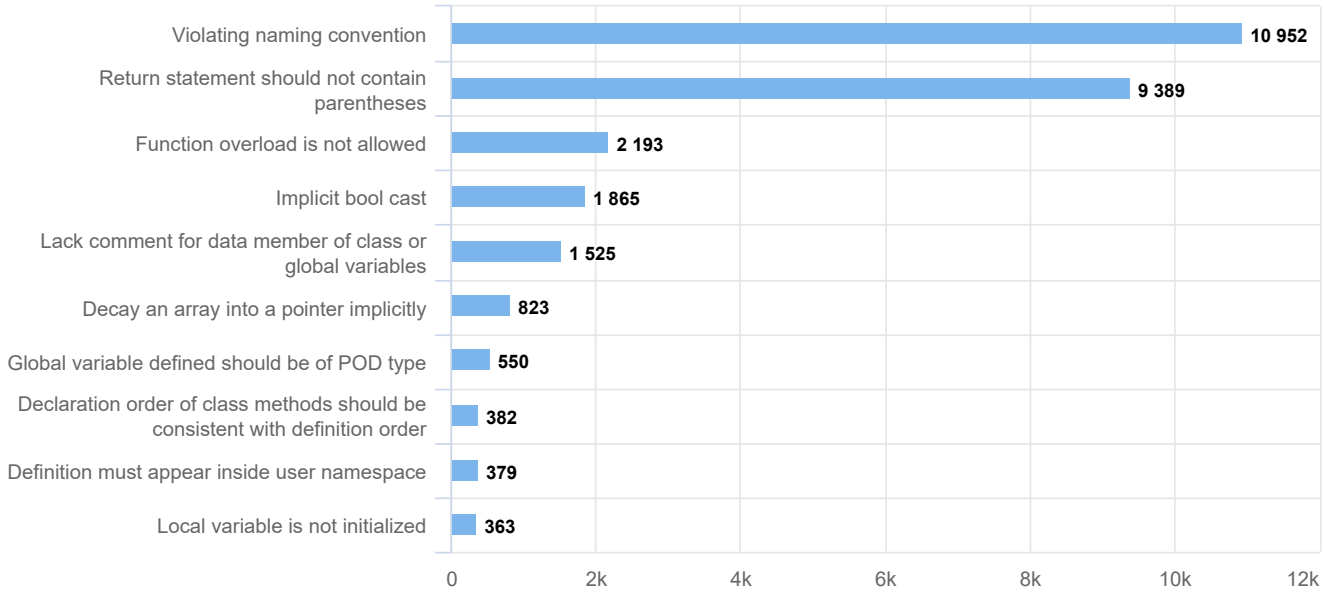
Top 10 high issues



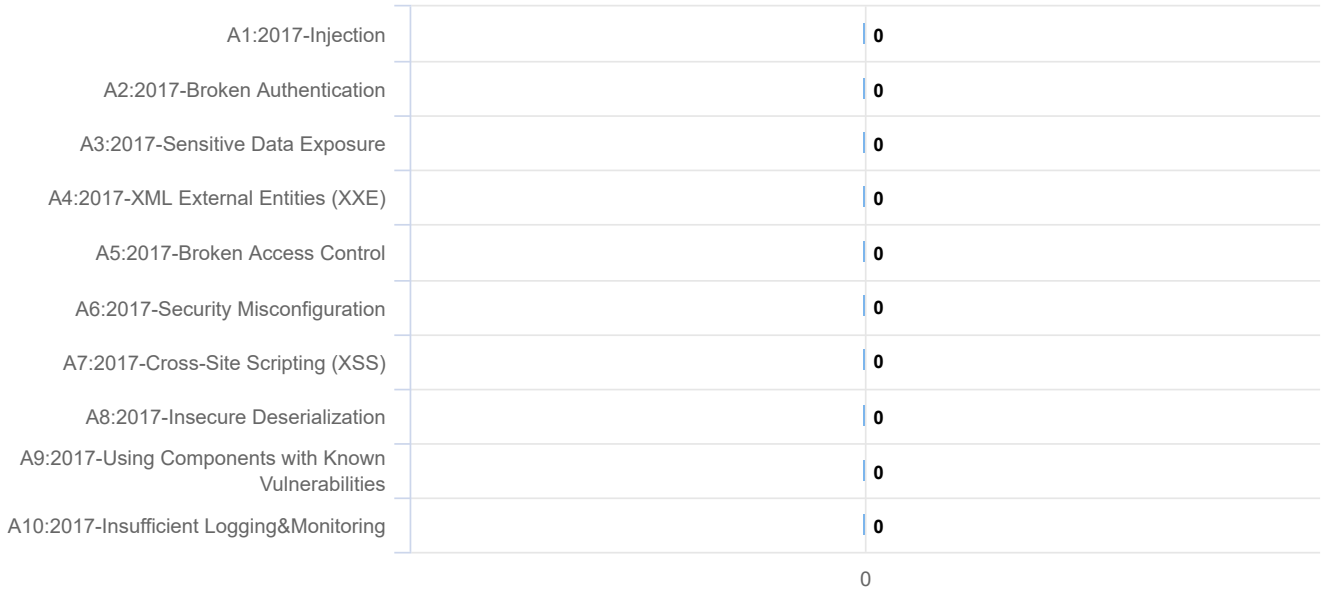
Top 10 medium issues



Top 10 low issues



OWASP Top 10


A1:2017-Injection
0
A2:2017-Broken Authentication
0
A3:2017-Sensitive Data Exposure
0
A4:2017-XML External Entities (XXE)
0
A5:2017-Broken Access Control
0
A6:2017-Security Misconfiguration
0
A7:2017-Cross-Site Scripting (XSS)
0
A8:2017-Insecure Deserialization
0
A9:2017-Using Components with Known Vulnerabilities
0
A10:2017-Insufficient Logging&Monitoring
0
0


CWE/SANS Top 25
TOP1: SQL Injection CWE-89
TOP2: OS Command Injection CWE-78
TOP3: Classic Buffer Overflow CWE-120
TOP4: Cross-site Scripting CWE-79
TOP5: Missing Authentication for Critical Fun…
TOP6: Missing Authorization CWE-862
TOP7: Use of Hard-coded Credentials CWE-…
TOP8: Missing Encryption of Sensitive Data …
TOP9: Unrestricted Upload of File with Dang…
TOP10: Reliance on Untrusted Inputs in a Se…
TOP11: Execution with Unnecessary Privileg…
TOP12: Cross-Site Request Forgery (CSRF)…
TOP13: Improper Limitation of a Pathname t…
TOP14: Download of Code Without Integrity …
TOP15: Incorrect Authorization CWE-863
TOP16: Inclusion of Functionality from Untru…
TOP17: Incorrect Permission Assignment for…
TOP18: Use of Potentially Dangerous Functi…
TOP19: Use of a Broken or Risky Cryptograp…
TOP20: Incorrect Calculation of Buffer Size …
TOP21: Improper Restriction of Excessive A…
TOP22: URL Redirection to Untrusted Site C…
TOP23: Uncontrolled Format String CWE-134
TOP24: Integer Overflow or Wraparound CW…
TOP25: Use of a One-Way Hash without a S…
0
0
0
0
0
0
108
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
20
40
60
80
100
120

[Critical] Divide by zero

- Description
- Vulnerability and risk
- Likelihood of Exploit
- Potential Mitigations
- Demonstrative Examples
  - Code example
  - Fixed code example

Description

When a value is divided by zero, a runtime error occurs, and the program usually crashes.

Vulnerability and risk

If the divisor is some uncerntain value including zero, the program will crash during execution.

Likelihood of Exploit

High

Potential Mitigations

Check whether the divisor is zero or not before running any division, especially it is divided by other uncerntain value, i.e., some calculation result.

Demonstrative Examples

Code example

```
1 int bad(int a){
2   int b = 0;
3   return a / b;
4 }
```

In the above program, b is zero, when it executes "a/b", the divisor is zero, program crashes.

Fixed code example

```
int good(int a){
  int b = 0;
  if(b != 0)
    return a / b;
  return a;
}
```

In the above program, b is checked before running "a/b", which guarantees no divisor is zero.

Issue report (1 - 1)

Issue Location:
tdutils/td/utils/benchmark.h: 107

Review:	Confirmed
Time:	2019-09-27 19:58:10
Issue ID:	cc26a733d46f0dfd6cd050e0b14ac9c9

Trigger steps:

tdutils/td/utils/benchmark.h

```
 96   return bench_n(b, n);
 97 }
 98
 99 inline void bench(Benchmark &b, double max_time = 1.0) {
100   int n = 1;
      1 Store 0.000000e+00 to pass_time
101   double pass_time = 0;
102   double total_pass_time = 0;
103   while (pass_time < max_time && total_pass_time < max_time * 3 && n < (1
    << 30)) {
104     n *= 2;
105     std::tie(pass_time, total_pass_time) = bench_n(b, n);
```

tdutils/td/utils/benchmark.h

```
102   double total_pass_time = 0;
103   while (pass_time < max_time && total_pass_time < max_time * 3 && n < (1
    << 30)) {
104     n *= 2;
```

```
105     std::tie(pass_time, total_pass_time) = bench_n(b, n);
106   }
      2 The divisor pass_time could be 0 in the calculation pass_time
107   pass_time = n / pass_time;
108
109   int pass_cnt = 2;
110   double sum = pass_time;
111   double square sum = pass time * pass time;
```

[Critical] NULL pointer dereference

- Description
- Vulnerability and risk
  - DoS: crash / exit / restart
  - Execute unauthorized code or commands
- Likelihood of Exploit
- Potential Mitigations
- Demonstrative Examples
  - Vulnerable code Example 1
  - Fixed Code Example 1
  - Example 2
  - Example 3
  - Fixed code example 3
- References

Description

An attempt to access data using a null pointer causes a runtime error. When a program dereferences a pointer that is expected to be valid but turns out to be null, a null pointer dereference occurs. Null-pointer dereference defects often occur due to ineffective error handling or race conditions, and typically cause abnormal program termination. Before a pointer is dereferenced in C/C++ code, it must be checked to confirm that it is not equal to null.

Vulnerability and risk

DoS: crash / exit / restart

NULL pointer dereferences usually result in the failure of the process unless exception handling (on some platforms) is available and implemented. Even when exception handling is being used, it can still be very difficult to return the software to a safe state of operation.

Execute unauthorized code or commands

In very rare circumstances and environments, code execution is possible.

Likelihood of Exploit

Critical

Potential Mitigations

- Check for a null value in the results of all functions that return values
- Make sure all external inputs are validated
- Explicitly initialize variables
- Make sure that unusual exceptions are handled correctly

Demonstrative Examples

Vulnerable code Example 1

```
 1 void reassign(int *argument, int *p) {
 2   if (goodEnough(argument)) return;
 3   *argument = *p;
 4 }
 5
 6 void npd_check_call_must(int *argument) {
 7   int *p = getValue();
 8   if (p != 0) {
 9     *p = 1;
10   }
11   reassign(argument, p);
12 }
```

Although *p is checked for null at line 8, it's then passed to function reassign, in which it is dereferenced without being checked for null. This type of vulnerability can produce unexpected and unintended results.

Fixed Code Example 1

```
 1 void reassign(int *argument, int *p) {
 2   if (goodEnough(argument)) return;
 3   *argument = *p;
 4 }
 5
 6 void npd_check_call_must(int *argument) {
 7   int *p = getValue();
 8   if (p != 0) {
 9     *p = 1;
10   }
11   if (p != 0) reassign(argument, p);
12 }
```

In the fixed version of the code, a second check for null has been put in line 11.

Example 2

This example takes an IP address from a user, verifies that it is well formed and then looks up the hostname and copies it into a buffer.

```
void host_lookup(char *user_supplied_addr){
struct hostent *hp;
in_addr_t *addr;
char hostname[64];
in_addr_t inet_addr(const char *cp);

// routine that ensures user_supplied_addr is in the right format for conversion
validate_addr_form(user_supplied_addr);
addr = inet_addr(user_supplied_addr);
hp = gethostbyaddr(addr, sizeof(struct in_addr), AF_INET);
strcpy(hostname, hp->h_name);
}
```

If an attacker provides an address that appears to be well-formed, but the address does not resolve to a hostname, then the call to gethostbyaddr() will return NULL. Since the code does not check the return value from gethostbyaddr, a NULL pointer dereference would then occur in the call to strcpy().

Example 3

This java example illustrates missing check of returned value might lead to crash of a program.

```
public static String readConfigFromFile(File file) {
  try {
    return com.google.common.io.Files.toString(file);
  } catch (java.io.IOException e) {
    return null;
  }
}

public static void main(String[] args) {
  if (args.length < 0) {
    return;
  }
  String config = readConfigFromFile(args[0]);
  // config might be null. If so, main exits.
  String[] lines = config.split("\n");
```

```
16   ...
17 }
```

Fixed code example 3

```
 1 public static String readConfigFromFile(File file) {
 2   try {
 3     return com.google.common.io.Files.toString(file);
 4   } catch (java.io.IOException e) {
 5     return null;
 6   }
 7 }
 8
 9 public static void main(String[] args) {
10   if (args.length < 0) {
11     return;
12   }
13   String config = readConfigFromFile(args[0]);
14   if (config == null) {
15     System.err.println("Can't read config from " + args[0]);
16     System.exit(1);
17   }
18   String[] lines = config.split("\n");
19   ...
20 }
```

References

1. https://cwe.mitre.org/data/definitions/476.html ↩

Issue report

Issue Location:
catchain/catchain-receiver.cpp: 329

Review:	Confirmed
Time:	2019-09-27 19:58:10
Issue ID:	a1adaba60a2ae5a78bfac99a5574076e

Trigger steps:

catchain/catchain-receiver.cpp

```
236   to_run_.push_back(block);
237 }
238
239 CatChainReceivedBlock *CatChainReceiverImpl::get_block(CatChainBlockHash
    hash) const {
240   auto it = blocks_.find(hash);
      1 Select the true branch at this point (it->operator==(ref) is
      true)
241   if (it == blocks_.end()) {
242     return nullptr;
243   } else {
244     return it->second.get();
245   }
```

catchain/catchain-receiver.cpp

```
241   if (it == blocks_.end()) {
242     return nullptr;
243   } else {
244     return it->second.get();
245   }
      2 Return null to caller
246 }
247
248 void
    CatChainReceiverImpl::add_block_cont_3(tl_object_ptr<ton_api::catchain_blo
    ck> block, td::BufferSlice payload) {
249   last_sent_block_ = create_block(std::move(block),
    td::SharedSlice{payload.as_slice()});
250   last_sent_block_->written();
```

catchain/catchain-receiver.cpp

```
319   auto prev = last_sent_block_->export_tl_dep();
320
321   std::vector<tl_object_ptr<ton_api::catchain_block_dep>> deps_arr;
322   deps_arr.resize(deps.size());
323   for (size_t i = 0; i < deps.size(); i++) {
      3 Function get_block executes and stores the return value to B (B
      can be null)
324     auto B = get_block(deps[i]);
```

```
325     LOG_CHECK(B != nullptr) << this << ": cannot find block with hash " <<
deps[i];
326     if (!intentional_fork_) {
327       CHECK(B->get_source_id() != local_idx_);
328     }
```

catchain/catchain-receiver.cpp

```
324     auto B = get_block(deps[i]);
325     LOG_CHECK(B != nullptr) << this << ": cannot find block with hash " <<
deps[i];
326     if (!intentional_fork_) {
327       CHECK(B->get_source_id() != local_idx_);
328     }
      4 Call a virtual function on B
329     deps_arr[i] = B->export_tl_dep();
330   }
331
332   auto height = prev->height_ + 1;
333   auto block_data = create_tl_object<ton_api::catchain_block_data>
(std::move(prev), std::move(deps_arr));
```

Issue Location:
catchain/catchain-receiver.cpp: 394

Review: Confirmed

Time: 2019-09-27 19:58:10

Issue ID: d9e9680aecbbd7a4574c4a2f8a095ffd

Trigger steps:

catchain/catchain-receiver.cpp

```
236   to_run_.push_back(block);
237 }
238
239 CatChainReceivedBlock *CatChainReceiverImpl::get_block(CatChainBlockHash
hash) const {
240   auto it = blocks_.find(hash);
      1 Select the true branch at this point (it->operator==(ref) is
      true)
241   if (it == blocks_.end()) {
242     return nullptr;
243   } else {
244     return it->second.get();
```

```
245   }
```

catchain/catchain-receiver.cpp

```
241   if (it == blocks_.end()) {
242     return nullptr;
243   } else {
244     return it->second.get();
245   }
```

2 Return **null** to caller

```
246 }
247
248 void
    CatChainReceiverImpl::add_block_cont_3(tl_object_ptr<ton_api::catchain_blo
    ck> block, td::BufferSlice payload) {
249   last_sent_block_ = create_block(std::move(block),
    td::SharedSlice{payload.as_slice()});
250   last sent block ->written();
```

catchain/catchain-receiver.cpp

```
387   }
388
389   std::vector<tl_object_ptr<ton_api::catchain_block_dep>> deps_arr;
390   deps_arr.resize(deps.size());
391   for (size_t i = 0; i < deps.size(); i++) {
```

3 Function **get_block** executes and stores the return value to **B** (**B** can be null)

```
392     auto B = get_block(deps[i]);
393     LOG_CHECK(B != nullptr) << this << ": cannot find block with hash " <<
    deps[i];
394     CHECK(B->get_source_id() != local_idx_);
395     deps_arr[i] = B->export_tl_dep();
396   }
```

catchain/catchain-receiver.cpp

```
389   std::vector<tl_object_ptr<ton_api::catchain_block_dep>> deps_arr;
390   deps_arr.resize(deps.size());
391   for (size_t i = 0; i < deps.size(); i++) {
392     auto B = get_block(deps[i]);
393     LOG_CHECK(B != nullptr) << this << ": cannot find block with hash " <<
    deps[i];
```

4 Call a virtual function on **B**

```
394     CHECK(B->get_source_id() != local_idx_);
395     deps_arr[i] = B->export_tl_dep();
```

```
396   }
397
398   auto block_data = create_tl_object<ton_api::catchain_block_data>(prev-
>export_tl_dep(), std::move(deps_arr));
```

Issue Location:
catchain/catchain-receiver.cpp: 938

Review: Confirmed

Time: 2019-09-27 19:58:10

Issue ID: 2d179368203812ce9825d9c701f8521d

Trigger steps:

catchain/catchain-receiver.cpp

```
605   return std::move(A);
606 }
607
608 CatChainReceiverSource
    *CatChainReceiverImpl::get_source_by_hash(PublicKeyHash source_hash) const
    {
609   auto it = sources_hashes_.find(source_hash);
      1 Select the true branch at this point (it->operator==(ref) is
      true)
610   if (it == sources_hashes_.end()) {
611     return nullptr;
612   }
613   return get_source(it->second);
614 }
```

catchain/catchain-receiver.cpp

```
609   auto it = sources_hashes_.find(source_hash);
610   if (it == sources_hashes_.end()) {
611     return nullptr;
612   }
613   return get_source(it->second);
      2 Return null to caller
614 }
615
616 CatChainReceiverSource
    *CatChainReceiverImpl::get_source_by_adnl_id(adnl::AdnlNodeIdShort
    source_hash) const {
617   auto it = sources_adnl_addrs_.find(source_hash);
```

```
618   if (it == sources_adnl_addrs_.end()) {
```

catchain/catchain-receiver.cpp

```
932                           get_source(local_idx_)->get_adnl_id(),
    overlay_id_, local_id_, 0, std::move(data));
933 }
934 void CatChainReceiverImpl::send_custom_query_data(PublicKeyHash dst,
    std::string name,
935
    td::Promise<td::BufferSlice> promise, td::Timestamp timeout,
936                                                   td::BufferSlice query) {
```

3 Function **get_source_by_hash** executes and stores the return value to **S** (**S** can be null)

```
937   auto S = get_source_by_hash(dst);
938   td::actor::send_closure(overlay_manager_,
    &overlay::Overlays::send_query, S->get_adnl_id(),
939                           get_source(local_idx_)->get_adnl_id(),
    overlay_id_, std::move(name), std::move(promise),
940                           timeout, std::move(query));
941 }
```

catchain/catchain-receiver.cpp

```
933 }
934 void CatChainReceiverImpl::send_custom_query_data(PublicKeyHash dst,
    std::string name,
935
    td::Promise<td::BufferSlice> promise, td::Timestamp timeout,
936                                                   td::BufferSlice query) {
937   auto S = get_source_by_hash(dst);
```

4 Call a virtual function on **S**

```
938   td::actor::send_closure(overlay_manager_,
    &overlay::Overlays::send_query, S->get_adnl_id(),
939                           get_source(local_idx_)->get_adnl_id(),
    overlay_id_, std::move(name), std::move(promise),
940                           timeout, std::move(query));
941 }
942
```

Issue Location:
catchain/catchain-receiver.cpp: 955

Review: Confirmed

Time: 2019-09-27 19:58:10

Issue ID: 01b2773e4556adc3426a95d9c7f35b26

Trigger steps:

catchain/catchain-receiver.cpp

```
605   return std::move(A);
606 }
607
608 CatChainReceiverSource
    *CatChainReceiverImpl::get_source_by_hash(PublicKeyHash source_hash) const
    {
609   auto it = sources_hashes_.find(source_hash);
      1 Select the true branch at this point (it->operator==(ref) is
      true)
610   if (it == sources_hashes_.end()) {
611     return nullptr;
612   }
613   return get_source(it->second);
614 }
```

catchain/catchain-receiver.cpp

```
609   auto it = sources_hashes_.find(source_hash);
610   if (it == sources_hashes_.end()) {
611     return nullptr;
612   }
613   return get_source(it->second);
      2 Return null to caller
614 }
615
616 CatChainReceiverSource
    *CatChainReceiverImpl::get_source_by_adnl_id(adnl::AdnlNodeIdShort
    source_hash) const {
617   auto it = sources_adnl_addrs_.find(source_hash);
618   if (it == sources_adnl_addrs_.end()) {
```

catchain/catchain-receiver.cpp

```
949                           get_source(local_idx_)->get_adnl_id(),
    overlay_id_, std::move(name), std::move(promise),
950                           timeout, std::move(query), max_answer_size,
    via);
951 }
952
953 void CatChainReceiverImpl::send_custom_message_data(PublicKeyHash dst,
    td::BufferSlice data) {
```

```
    3 Function get_source_by_hash executes and stores the return value to S (S can be null)
954   auto S = get_source_by_hash(dst);
955   td::actor::send_closure(overlay_manager_, &overlay::Overlays::send_message, S->get_adnl_id(),
956                           get_source(local_idx_)->get_adnl_id(), overlay_id_, std::move(data));
957 }
958
```

catchain/catchain-receiver.cpp

```
950                           timeout, std::move(query), max_answer_size, via);
951 }
952
953 void CatChainReceiverImpl::send_custom_message_data(PublicKeyHash dst, td::BufferSlice data) {
954   auto S = get_source_by_hash(dst);
    4 Call a virtual function on S
955   td::actor::send_closure(overlay_manager_, &overlay::Overlays::send_message, S->get_adnl_id(),
956                           get_source(local_idx_)->get_adnl_id(), overlay_id_, std::move(data));
957 }
958
959 void CatChainReceiverImpl::block_written_to_db(CatChainBlockHash hash) {
```

[Critical] Return stack address

- Description
- Vulnerability and risk
- Likelihood of Exploit
- Potential mitigations
- Demonstrative Examples
  - Code example
  - Fixed code example
- References

Description

Because local variables are allocated on the stack, when a program returns a pointer to a local variable, it is returning a stack address. A subsequent function call is likely to re-use this same stack address, thereby overwriting the value of the pointer, which no longer corresponds to the same variable since a function's stack frame is invalidated when it returns. At best this will cause the value of the pointer to change unexpectedly. In many cases it causes the program to crash the next time the pointer is dereferenced.

Vulnerability and risk

DoS: crash / exit / restart

Likelihood of Exploit

High

Potential mitigations

Use malloc() dynamically to get the real memory address.

Demonstrative Examples

Code example

```
1 char* getName() {
2   char name[STR_MAX];
3   fillInName(name);
4   return name;
5 }
```

Fixed code example

```
char* getName() {
  char* name = (char*)malloc(STR_MAX * sizeof(char));
  fillInName(name);
  return name;
}
```

References

1. https://cwe.mitre.org/data/definitions/562.html ↩

Issue report

Issue Location:
tdfec/td/fec/online/Rfc.cpp: 42

Review:	Confirmed
Time:	2019-09-27 19:58:10
Issue ID:	c4884bd55f7776b557663b9d3abd8d17

Trigger steps:

tdfec/td/fec/online/Rfc.cpp

```
40    return size_t(static_cast<double>(source_blocks_count() +
outer_encoding_blocks_count()) * (1 + epsilon_));
41  }
42  Span<uint32> Rfc::Parameters::get_inner_encoding_row(size_t row_id) const {
43    Random r(static_cast<uint32>(row_id));
44    uint32 degree = degree_distribution_.get_degree(static_cast<uint32>(r() %
(1 << 20)) * 1.0 / (1 << 20));
      1 r is used as the 1st parameter in function set_random
45    inner_distribution_.set_random(&r);
46    MutableSpan<uint32> res(row_buffer_.data(), degree);
47    for (auto &x : res) {
48      x = inner_distribution_();
49    }
```

tdfec/td/fec/online/Rfc.h

```
30 class UniformDistributionSimple {
31  public:
32   UniformDistributionSimple(uint32 L, uint32 R) : L_(L), R_(R) {
33   }
34   void set_random(Random *random) {
     2 Store random to this->random_
35     random_ = random;
36   }
37   uint32 operator()() {
38     return static_cast<uint32>((*random_)() % (R_ - L_ + 1)) + L_;
39   }
```

tdfec/td/fec/online/Rfc.h

```
29 template <class Random>
30 class UniformDistributionSimple {
31  public:
32   UniformDistributionSimple(uint32 L, uint32 R) : L_(L), R_(R) {
33   }
     3 Return with value this->random_ modified
34   void set_random(Random *random) {
35     random_ = random;
36   }
37   uint32 operator()() {
38     return static_cast<uint32>((*random_)() % (R_ - L_ + 1)) + L_;
```

tdfec/td/fec/online/Rfc.cpp

```
40   return size_t(static_cast<double>(source_blocks_count() +
outer_encoding_blocks_count()) * (1 + epsilon_));
41 }
42 Span<uint32> Rfc::Parameters::get_inner_encoding_row(size_t row_id) const {
43   Random r(static_cast<uint32>(row_id));
44   uint32 degree = degree_distribution_.get_degree(static_cast<uint32>(r() %
(1 << 20)) * 1.0 / (1 << 20));
     4 Calling to function set_random modified &(this->inner_distribution_)->random_
45   inner_distribution_.set_random(&r);
46   MutableSpan<uint32> res(row_buffer_.data(), degree);
47   for (auto &x : res) {
48     x = inner_distribution_();
49   }
```

tdfec/td/fec/online/Rfc.cpp

```
37    return outer_encoding_blocks_count_;
38  }
39  size_t Rfc::Parameters::estimated_packets() const {
40    return size_t(static_cast<double>(source_blocks_count() +
    outer_encoding_blocks_count()) * (1 + epsilon_));
41  }
      5 Return with value this->inner_distribution_.random_ modified
42  Span<uint32> Rfc::Parameters::get_inner_encoding_row(size_t row_id) const
    {
43    Random r(static_cast<uint32>(row_id));
44    uint32 degree = degree_distribution_.get_degree(static_cast<uint32>(r() %
    (1 << 20)) * 1.0 / (1 << 20));
45    inner_distribution_.set_random(&r);
46    MutableSpan<uint32> res(row buffer .data(), degree);
```

Issue Location:
tdfec/td/fec/online/Rfc.h: 60

Review:	Confirmed
Time:	2019-09-27 19:58:10
Issue ID:	2e8f937b7f33b0cf3f19715dd7488f7b

Trigger steps:

tdfec/td/fec/online/Rfc.h

```
57      size_t estimated_packets() const;
58
59      template <class F>
60      void outer_encoding_for_each(F &&f) const {
61        Random r(static cast<uint32>(1));
      1 r is used as the 1st parameter in function set_random
62        outer_distribution_.set_random(&r);
63
64        for (uint32 j = 0; j < outer_encoding_blocks_count(); j++) {
65          f(j, uint32(source_blocks_count() + j));
66        }
```

tdfec/td/fec/online/Rfc.h

```
30  class UniformDistributionSimple {
31   public:
32    UniformDistributionSimple(uint32 L, uint32 R) : L_(L), R_(R) {
33    }
```

```
34   void set_random(Random *random) {
     2 Store random to this->random_
35     random_ = random;
36   }
37   uint32 operator()() {
38     return static_cast<uint32>((*random_)() % (R_ - L_ + 1)) + L_;
39   }
```

tdfec/td/fec/online/Rfc.h

```
29 template <class Random>
30 class UniformDistributionSimple {
31  public:
32   UniformDistributionSimple(uint32 L, uint32 R) : L_(L), R_(R) {
33   }
     3 Return with value this->random_ modified
34   void set_random(Random *random) {
35     random_ = random;
36   }
37   uint32 operator()() {
38     return static cast<uint32>((*random )() % (R  - L  + 1)) + L ;
```

tdfec/td/fec/online/Rfc.h

```
57     size_t estimated_packets() const;
58
59     template <class F>
60     void outer_encoding_for_each(F &&f) const {
61       Random r(static_cast<uint32>(1));
       4 Calling to function set_random modified &(this->outer_distribution_)->random_
62       outer_distribution_.set_random(&r);
63
64       for (uint32 j = 0; j < outer_encoding_blocks_count(); j++) {
65         f(j, uint32(source_blocks_count() + j));
66       }
```

tdfec/td/fec/online/Rfc.h

```
55     size_t source_blocks_count() const;
56     size_t outer_encoding_blocks_count() const;
57     size_t estimated_packets() const;
58
59     template <class F>
       5 Return with value this->outer_distribution_.random_ modified
```

```
60      void outer_encoding_for_each(F &&f) const {
61        Random r(static_cast<uint32>(1));
62        outer_distribution_.set_random(&r);
63
64        for (uint32 j = 0; j < outer_encoding_blocks_count(); j++) {
```

[High] Incorrectly specified parameters in function calls

- Description
- Vulnerability and risk
  - DoS: crash / exit / restart
  - Execute unauthorized code or commands
- Likelihood of Exploit
- Potential Mitigations
- Demonstrative Examples
  - Vulnerable code Example 1
  - Fixed Code Example 1
  - Example 2
  - Example 3
  - Fixed code example 3
- References

Description

An attempt to access data using a null pointer causes a runtime error. When a program dereferences a pointer that is expected to be valid but turns out to be null, a null pointer dereference occurs. Null-pointer dereference defects often occur due to ineffective error handling or race conditions, and typically cause abnormal program termination. Before a pointer is dereferenced in C/C++ code, it must be checked to confirm that it is not equal to null.

Vulnerability and risk

DoS: crash / exit / restart

NULL pointer dereferences usually result in the failure of the process unless exception handling (on some platforms) is available and implemented. Even when exception handling is being used, it can still be very difficult to return the software to a safe state of operation.

Execute unauthorized code or commands

In very rare circumstances and environments, code execution is possible.

Likelihood of Exploit

Critical

Potential Mitigations

- Check for a null value in the results of all functions that return values
- Make sure all external inputs are validated
- Explicitly initialize variables
- Make sure that unusual exceptions are handled correctly

Demonstrative Examples

Vulnerable code Example 1

```
 1 void reassign(int *argument, int *p) {
 2   if (goodEnough(argument)) return;
 3   *argument = *p;
 4 }
 5
 6 void npd_check_call_must(int *argument) {
 7   int *p = getValue();
 8   if (p != 0) {
 9     *p = 1;
10   }
11   reassign(argument, p);
12 }
```

Although *p is checked for null at line 8, it's then passed to function reassign, in which it is dereferenced without being checked for null. This type of vulnerability can produce unexpected and unintended results.

Fixed Code Example 1

```
 1 void reassign(int *argument, int *p) {
 2   if (goodEnough(argument)) return;
 3   *argument = *p;
 4 }
 5
 6 void npd_check_call_must(int *argument) {
 7   int *p = getValue();
 8   if (p != 0) {
 9     *p = 1;
10   }
11   if (p != 0) reassign(argument, p);
12 }
```

In the fixed version of the code, a second check for null has been put in line 11.

Example 2

This example takes an IP address from a user, verifies that it is well formed and then looks up the hostname and copies it into a buffer.

```
void host_lookup(char *user_supplied_addr){
struct hostent *hp;
in_addr_t *addr;
char hostname[64];
in_addr_t inet_addr(const char *cp);

// routine that ensures user_supplied_addr is in the right format for conversion
validate_addr_form(user_supplied_addr);
addr = inet_addr(user_supplied_addr);
hp = gethostbyaddr(addr, sizeof(struct in_addr), AF_INET);
strcpy(hostname, hp->h_name);
}
```

If an attacker provides an address that appears to be well-formed, but the address does not resolve to a hostname, then the call to gethostbyaddr() will return NULL. Since the code does not check the return value from gethostbyaddr, a NULL pointer dereference would then occur in the call to strcpy().

Example 3

This java example illustrates missing check of returned value might lead to crash of a program.

```
public static String readConfigFromFile(File file) {
  try {
    return com.google.common.io.Files.toString(file);
  } catch (java.io.IOException e) {
    return null;
  }
}

public static void main(String[] args) {
  if (args.length < 0) {
    return;
  }
  String config = readConfigFromFile(args[0]);
  // config might be null. If so, main exits.
  String[] lines = config.split("\n");
```

```
16    ...
17  }
```

Fixed code example 3

```
 1  public static String readConfigFromFile(File file) {
 2    try {
 3      return com.google.common.io.Files.toString(file);
 4    } catch (java.io.IOException e) {
 5      return null;
 6    }
 7  }
 8
 9  public static void main(String[] args) {
10    if (args.length < 0) {
11      return;
12    }
13    String config = readConfigFromFile(args[0]);
14    if (config == null) {
15      System.err.println("Can't read config from " + args[0]);
16      System.exit(1);
17    }
18    String[] lines = config.split("\n");
19    ...
20  }
```

References

1. https://cwe.mitre.org/data/definitions/476.html ↩

Issue report (1 - 1)

Issue Location: catchain/catchain-receiver.cpp: 948

Review:	Confirmed
Time:	2019-09-27 19:58:10
Issue ID:	a464d069d484533f7d89e05c95737ad3

Trigger steps:

catchain/catchain-receiver.cpp

```
942
943 void CatChainReceiverImpl::send_custom_query_data_via(PublicKeyHash dst,
    std::string name,
944
    td::Promise<td::BufferSlice> promise, td::Timestamp timeout,
945                                                       td::BufferSlice
    query, td::uint64 max_answer_size,
946
    td::actor::ActorId<adnl::AdnlSenderInterface> via) {
```

1 Calling **CatChainReceiverImpl::get_source_by_hash**

```
947   auto S = get_source_by_hash(dst);
948   td::actor::send_closure(overlay_manager_,
    &overlay::Overlays::send_query_via, S->get_adnl_id(),
949                           get_source(local_idx_)->get_adnl_id(),
    overlay_id_, std::move(name), std::move(promise),
950                           timeout, std::move(query), max_answer_size,
    via);
951 }
```

catchain/catchain-receiver.cpp

```
603                                                       keyring, adnl,
    overlay_manager, std::move(ids), local_id,
604                                                       unique_hash,
    db_root);
605   return std::move(A);
606 }
607
```

2 Entered call from **CatChainReceiverImpl::send_custom_query_data_via**

```
608 CatChainReceiverSource
    *CatChainReceiverImpl::get_source_by_hash(PublicKeyHash source_hash) const
    {
609   auto it = sources_hashes_.find(source_hash);
610   if (it == sources_hashes_.end()) {
611     return nullptr;
612   }
```

catchain/catchain-receiver.cpp

```
605   return std::move(A);
606 }
607
```

```
608 CatChainReceiverSource
    *CatChainReceiverImpl::get_source_by_hash(PublicKeyHash source_hash) const
    {
609   auto it = sources_hashes_.find(source_hash);
      3 Assuming the condition is true
610   if (it == sources_hashes_.end()) {
611     return nullptr;
612   }
613   return get_source(it->second);
614 }
```

catchain/catchain-receiver.cpp

```
606 }
607
608 CatChainReceiverSource
    *CatChainReceiverImpl::get_source_by_hash(PublicKeyHash source_hash) const
    {
609   auto it = sources_hashes_.find(source_hash);
610   if (it == sources_hashes_.end()) {
      4 Returning null pointer
611     return nullptr;
612   }
613   return get_source(it->second);
614 }
615
```

catchain/catchain-receiver.cpp

```
942
943 void CatChainReceiverImpl::send_custom_query_data_via(PublicKeyHash dst,
    std::string name,
944
    td::Promise<td::BufferSlice> promise, td::Timestamp timeout,
945                                                   td::BufferSlice
    query, td::uint64 max_answer_size,
946
    td::actor::ActorId<adnl::AdnlSenderInterface> via) {
      5 Returning from CatChainReceiverImpl::get_source_by_hash
947   auto S = get_source_by_hash(dst);
948   td::actor::send_closure(overlay_manager_,
    &overlay::Overlays::send_query_via, S->get_adnl_id(),
949                           get_source(local_idx_)->get_adnl_id(),
    overlay_id_, std::move(name), std::move(promise),
950                           timeout, std::move(query), max_answer_size,
    via);
951 }
```

catchain/catchain-receiver.cpp

```
942
943 void CatChainReceiverImpl::send_custom_query_data_via(PublicKeyHash dst,
    std::string name,
944
    td::Promise<td::BufferSlice> promise, td::Timestamp timeout,
945                                                   td::BufferSlice
    query, td::uint64 max_answer_size,
946
    td::actor::ActorId<adnl::AdnlSenderInterface> via) {
      6 S initialized to a null pointer value
947   auto S = get_source_by_hash(dst);
948   td::actor::send_closure(overlay_manager_,
    &overlay::Overlays::send_query_via, S->get_adnl_id(),
949                           get_source(local_idx_)->get_adnl_id(),
    overlay_id_, std::move(name), std::move(promise),
950                           timeout, std::move(query), max_answer_size,
    via);
951 }
```

catchain/catchain-receiver.cpp

```
943 void CatChainReceiverImpl::send_custom_query_data_via(PublicKeyHash dst,
    std::string name,
944
    td::Promise<td::BufferSlice> promise, td::Timestamp timeout,
945                                                   td::BufferSlice
    query, td::uint64 max_answer_size,
946
    td::actor::ActorId<adnl::AdnlSenderInterface> via) {
947   auto S = get_source_by_hash(dst);
      7 Called C++ object pointer is null
948   td::actor::send_closure(overlay_manager_,
    &overlay::Overlays::send_query_via, S->get_adnl_id(),
949                           get_source(local_idx_)->get_adnl_id(),
    overlay_id_, std::move(name), std::move(promise),
950                           timeout, std::move(query), max_answer_size,
    via);
951 }
952
```

[High] Weak Encrypt

Weak encrypt

Issue report

Issue Location:
tdutils/td/utils/crypto.cpp: 291

Review:	Confirmed
Time:	2019-09-27 19:58:10
Issue ID:	5f624dae47935e238474c4563fbd7be6

Trigger steps:

tdutils/td/utils/crypto.cpp

```
286   } else {
287     err = AES_set_decrypt_key(aes_key.ubegin(), 256, &key);
288   }
289   LOG_IF(FATAL, err != 0);
290   CHECK(from.size() <= to.size());
      1 Use of weak encrypt algorithm function AES_cbc_encrypt
291   AES_cbc_encrypt(from.ubegin(), to.ubegin(), from.size(), &key,
aes_iv.ubegin(), encrypt_flag);
292 }
293
294 void aes_cbc_encrypt(Slice aes_key, MutableSlice aes_iv, Slice from,
MutableSlice to) {
295   aes_cbc_xcrypt(aes_key, aes_iv, from, to, true);
```

Issue Location:
tdutils/td/utils/crypto.cpp: 457

Review:	Confirmed
Time:	2019-09-27 19:58:10
Issue ID:	9a53fc6ac8a21ee1594a01335fb3c70a

Trigger steps:

tdutils/td/utils/crypto.cpp

```
452   }
453 }
```

```
454
455 void md5(Slice input, MutableSlice output) {
456   CHECK(output.size() >= MD5_DIGEST_LENGTH);
    1 Use of weak encrypt algorithm function MD5
457   auto result = MD5(input.ubegin(), input.size(), output.ubegin());
458   CHECK(result == output.ubegin());
459 }
460
461 static void pbkdf2_impl(Slice password, Slice salt, int iteration_count,
    MutableSlice dest, const EVP_MD *evp_md) {
```

[Medium] Realloc buffer without clear

Realloc buffer without clear

Issue report (1 - 1)

Issue Location:
crypto/common/bitstring.cpp: 75

Review:	Confirmed
Time:	2019-09-27 19:58:10
Issue ID:	f874a4307cfcb968ed678daf7bd8c920

Trigger steps:

crypto/common/bitstring.cpp

```
70
71  BitString& BitString::reserve_bits(unsigned req_bits) {
72    req_bits += offs + len;
73    if (req_bits > bytes_alloc * 8) {
74      bytes_alloc = (req_bits + 7) >> 3;
      1 Realloc buffer without clean raw data.
75      ptr = (unsigned char*)std::realloc(ptr, bytes_alloc);
76      CHECK(ptr);
77    }
78    return *this;
79  }
```