Public Blockchain Secure Audit Report

Numen Cyber Labs - Security Services

For Armonia

Table of Contents

Numen Cyber

Confidential

1. Project Information

Project Name: Armonia

Source Code Link: https://github.com/armoniax/amax.eva.chain

Commit Hash: 6545aaa48609368736de1a24fb1d14a44f950a0c

Audit Time: 2023/4/1- 2023/5/4

Language: C++

1.1 public blockchain

name	Armonia Meta Chain
Symbol	AMAX
Decimals	8
Total supply	1,000,000,000
Block interval	1 second
Consensus algorithm	APOS
TPS	5000+

Numen Cyber

Confidential

2. Audit Scope and Result

2.1 Audit Scope

No	Categories	items
1	P2P Communication Security	Connection Number Occupation Audit
		Eclipse Attack
		Packet Size Limit
		Node Communication Protocol Security
2	RPC Interface Security	RPC Sensitive Interface Permissions
		Traditional Web Security
		RPC Interface Security
3	Consensus Mechanism Security	Design Of Consensus Mechanism
		Implementation Of Consensus Verification
		Incentive Mechanism Audit
4	Transaction processing Security	Transaction Signature Logic
		Transaction Verification Logic
		Transaction Processing Logic
		Transaction Fee Setting
		Transaction Replay

Numen Cyber

Confidential

5	Cryptography Security	Random Number Range And Probability Distribution
		Cryptographic Algorithm Lmplementation/Use
6	Wallet Module & Account Security Audit	Private Key / Mnemonic Word Storage Security
		Private Key / Mnemonic Word Usage Security
		Private key/mnemonic generation algorithm
7	Others Security Audit	Database Security
		Thread Security
		File Permission Security
		Historical Vulnerability Security

Numen Cyber

2.1 Test Method

- White-box Testing

Conduct a security audit of the project's source code and SDK, as well as dynamic debugging and vulnerability discovery of the p2p and RPC nodes.

- Gray-box Testing

Utilize security scanning and auditing tools to conduct a security assessment of the project's source code, identifying potential vulnerability points that could lead to anomalous behavior.

- Black-box Testing

Simulate security testing attacks against the nodes to check whether they respond properly.

Confidential

2.2 Audit Result

Numen Cyber Labs conducted a comprehensive audit of xxx blockchain's code security and business logic security using black-box, white-box, and grey-box testing methods.

- Critical code files or functions：1
- Major-risk code files or functions：6
- Medium-risk code files or functions：1
- Low-risk code files or functions：3

The comprehensive evaluation is considered as a PASSED.

Numen Cyber

Confidential

3.Vulnerability Summary

3.1 Vulnerability level distribution

Table of vulnerability risk level counts			
Critical	Major	Medium	Low
1	6	1	3

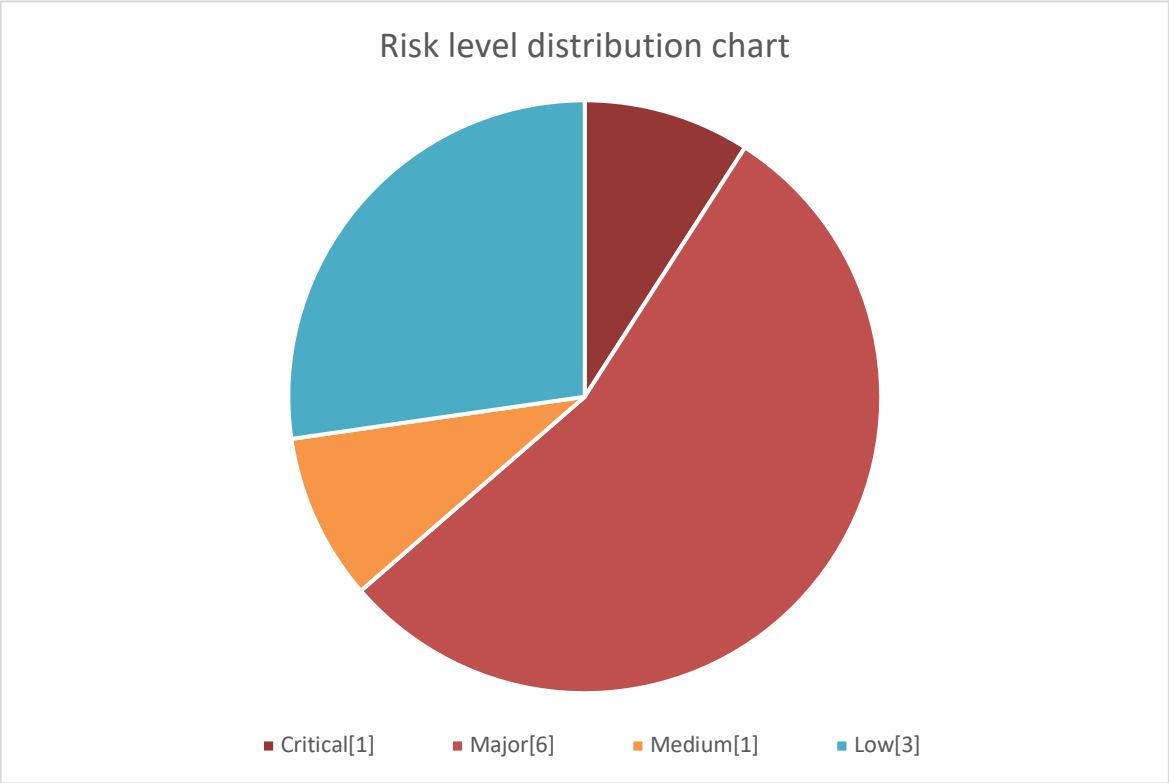

Table 3.1: Vulnerability level distribution

Confidential

3.2 Key Findings

ID	Severity	Findings Title	Confirm	STATUS
NVE-001	Critical	Integer overflows leading to out-of-bounds access can lead to remote code execution	Confirmed	Fixed
NVE-002	Major	Null pointer access can cause node DOS	Confirmed	Fixed
NVE-003	Major	Node denial of service due to null pointer access	Confirmed	Fixed
NVE-004	Major	Out-of-bounds accesses cause node memory corruption	Confirmed	Fixed
NVE-005	Major	Null pointer access can cause node DOS	Confirmed	Fixed
NVE-006	Major	Null pointer access can cause node DOS	Confirmed	Fixed
NVE-007	Major	The find method may return nullptr	Confirmed	Fixed
NVE-008	Medium	Possible illegal characters	Confirmed	Fixed
NVE-009	Low	Design issue - block_num type should be set to uint64_t	Confirmed	Ingored
NVE-010	Low	Unreasonable use of space	Confirmed	Ingored
NVE-011	Low	Possible performance problems caused by too much data	Confirmed	Ingored

Table 3.2: Key Findings

Confidential

4.Detailed Explanation of the Audit Results

4.1 Integer overflows leading to out-of-bounds access can lead to remote code execution

Code path:libraries/eos-vm/include/eosio/vm/execution_context.hpp

Risk description:

```
class execution_context : public execution_context_base<execution_contex
t<Host>, Host> {
  using base_type=execution_context_base<execution_context<Host>, Host>;
    public:
      using base_type::_mod;
      using base_type::_rhf;
      using base_type::_linear_memory;
      using base_type::_error_code;
      using base_type::handle_signal;
      ...
      inline operand_stack_elem& peek_operand(size_t i = 0) { return _o
s.peek(i); }
```

Numen Cyber

peek_operand returns the data at index i of the vector _store of the current operator stack

ElemT& peek(size_t i) { return _store[_index - 1 - i]; }

When the user constructs a malicious wasm file, _index=0 i>0 will cause an integer overflow.

In the type_check function const auto& op = peek_operand((ft.param_types.size() - 1) - i); after that op is an illegal address.

```
inline void type_check(const func_type& ft) {
    for (uint32_t i = 0; i < ft.param_types.size(); i++) {
       const auto& op = peek_operand((ft.param_types.size() - 1) - i);
       visit(overloaded{ [&](const i32_const_t&) {
           EOS_VM_ASSERT(ft.param_types[i] == types::i32, wasm_interpret
er_exception,"function param type mismatch");
       },
       [&](const f32_const_t&) {
           EOS_VM_ASSERT(ft.param_types[i] == types::f32, wasm_interpret
er_exception,"function param type mismatch");
       },
       [&](const i64_const_t&) {
           EOS_VM_ASSERT(ft.param_types[i] == types::i64, wasm_interpret
er_exception,"function param type mismatch");
       },
```


Confidential

```
       [&](const f64_const_t&) {
          EOS_VM_ASSERT(ft.param_types[i] == types::f64, wasm_interpret
er_exception "function param type mismatch");
       },
        [&](auto) { throw wasm_interpreter_exception{ "function param in
valid type" }; } },op);}
```

STATUS:fixed

Fixed Git Commit:
https://github.com/armoniax/eos-vm/commit/755648758472bce5c146f671e900a4c53bc386c0

4.2 Null pointer access can cause node DOS

Code path:libraries/chain/apply_context.cpp

Risk description:

The function get_context_free_data, needs to determine if the buffer pointer is empty, otherwise it will cause a node DOS.

```
int apply_context::get_context_free_data( uint32_t index, char* buffer,
size_t buffer_size )const
{
   const auto& trx = trx_context.trx;

   if( index >= trx.context_free_data.size() ) return -1;

   auto s = trx.context_free_data[index].size();
   if( buffer_size == 0 ) return s;

   auto copy_size = std::min( buffer_size, s );
   memcpy( buffer, trx.context_free_data[index].data(), copy_size );

   return copy_size;
}
```

STATUS:fixed

Fixed Git Commit:
https://github.com/armoniax/amax.meta.chain/commit/fcd4461da49d6ecdd0d467fe926700dc42920b4b

4.3 Node denial of service due to null pointer access

Code path:libraries/fc/secp256k1/upstream/src/secp256k1.c

Risk description:

Confidential

Lines 6-10 all use DEBUG_CHECK, which in the RELEASE version does not check for data legitimacy. If ctx is empty at this point, ctx->ecmult_ctx will cause a null pointer access, which in turn will cause the node to deny service.

```
int secp256k1_ecdsa_verify(const secp256k1_context_t* ctx, const unsigne
d char *msg32, const unsigned char *sig, int siglen, const unsigned char
 *pubkey, int pubkeylen) {
    secp256k1_ge_t q;
    secp256k1_ecdsa_sig_t s;
    secp256k1_scalar_t m;
    int ret = -3;
    DEBUG_CHECK(ctx != NULL);
    DEBUG_CHECK(secp256k1_ecmult_context_is_built(&ctx->ecmult_ctx));
    DEBUG_CHECK(msg32 != NULL);
    DEBUG_CHECK(sig != NULL);
    DEBUG_CHECK(pubkey != NULL);
```

STATUS:fixed

Fixed Git Commit:
https://github.com/armoniax/secp256k1-zkp/commit/2e35e5d58638b26d6a550be870d20bf8f7a31c5c

Numen Cyber

4.4 Out-of-bounds accesses cause node memory corruption

Code path:libraries/fc/secp256k1/upstream/src/rangeproof_impl.h

Risk description:

When VERIFY_CHECK is not turned on.

```
/* Like DEBUG_CHECK(), but when VERIFY is defined instead of NDEBUG not
defined. */
#ifdef VERIFY
#define VERIFY_CHECK CHECK
#else
#define VERIFY_CHECK(cond) do { (void)(cond); } while(0)
#endif
```

rings can be 0, since VERIFY_CHECK(*rings > 0); the execution will pass without throwing an exception.

```
    /* If the masked number isn't precise, compute the public offset. */
        *min_value = value - v2;
        /* How many bits do we need to represent our value? */
        *mantissa = *v ? 64 - secp256k1_clz64_var(*v) : 1;
        if (*min_bits > *mantissa) {
            /* If the user asked for more precision, give it to them. */
            *mantissa = *min_bits;
        }
        /* Digits in radix-4, except for the last digit if our mantissa
```


Confidential

```
length is odd. */
        *rings = (*mantissa + 1) >> 1;
        for (i = 0; i < *rings; i++) {
            rsizes[i] = ((i < *rings - 1) | (!(*mantissa&1))) ? 4 : 2;
            *npub += rsizes[i];
            secidx[i] = (*v >> (i*2)) & 3;
        }
```

When the following lines 2 and 6 are executed, out-of-bounds accesses are caused, resulting in node memory corruption.

```
  memset(prep, 0, 4096);
  /*Note, the data corresponding to the blinding factors must be zero.*/
    if (rsizes[rings - 1] > 1) {
        int idx;
        /* Value encoding sidechannel. */
        idx = rsizes[rings - 1] - 1;
        idx -= secidx[rings - 1] == idx;
        idx = ((rings - 1) * 4 + idx) * 32;
        for (i = 0; i < 8; i++) {
            prep[8 + i + idx] = prep[16 + i + idx] = prep[24 + i + idx]
= (v >> (56 - i * 8)) & 255;
            prep[i + idx] = 0;
        }
        prep[idx] = 128;
    }
```

Numen Cyber

STATUS:fixed

Fixed Git Commit:
https://github.com/armoniax/secp256k1-zkp/commit/2e35e5d58638b26d6a550be870d20bf8f7a31c5c

4.5 Null pointer access can cause node DOS

Code path:libraries/fc/src/rpc/cli.cpp

Risk description:

Failure to determine whether the memory request was successful after malloc may result in a null pointer access, which can cause node DOS.

```
char * dupstr (const char* s) {
   char *r;

   r = (char*) malloc ((strlen (s) + 1));
   strcpy (r, s);
   return (r);
}
```


Confidential

STATUS:fixed

Fixed Git Commit:
https://github.com/armoniax/amax.fc/commit/0c89881cc66b29a20450081b5bd95666de7beb82

4.6 Null pointer access can cause node DOS

Code path:libraries/eos-vm/include/eosio/vm/execution_context.hpp

Risk description:

In the call_host_function function, rhf will call the following code：

```
      template <typename Execution_Context>
      void operator()(Cls* host, Execution_Context& ctx, uint32_t index)
 {
         const auto& _func = get_mappings<wasm_allocator>().functions[in
dex];
         std::invoke(_func, host, ctx.get_wasm_allocator(), ctx.get_oper
and_stack());
      }
```

Numen Cyber

But get_mappings<wasm_allocator>() will return mappings of the type wasm_allocator.

```
      template <typename WAlloc>
      static mappings<WAlloc>& get_mappings() {
         static mappings<WAlloc> _mappings;
         return _mappings;
      }

    struct mappings {
         std::unordered_map<std::pair<std::string, std::string>, uint32_
t, host_func_pair_hash> named_mapping;
         std::vector<host_function>
                        host_functions;
         std::vector<std::function<void(Cls*, WAlloc*, operand_stack&)>>
                        functions;
         size_t
                        current_index = 0;
      };
```

An attacker can construct a malicious wasm file so that functions are empty, so when calling

const auto& _func = get_mappings<wasm_allocator>().functions[index]; may return a null pointer

Confidential

Finally when calling ` std::invoke(_func, host, ctx.get_wasm_allocator(), ctx.get_operand_stack()); Since _func is null, it will cause a null pointer access, resulting in a node DOS.

STATUS:fixed

Fixed Git Commit:
https://github.com/armoniax/eos-vm/commit/755648758472bce5c146f671e900a4c53bc386c0

4.7 The find method may return nullptr

Code path:libraries/chain/include/eosio/chain/wasm_eosio_injection.hpp

Risk description:

The find method here mapped_indexr may return null.

```
238         chktm_idx = 0;
239      }
240      static void accept( wasm_ops::instr* inst, wasm_ops::visitor_arg& arg ) {
241         auto mapped_index = injector_utils::injected_index_mapping.find(chktm_idx);
242
243         wasm_ops::op_types<>::call_t chktm;
244         chktm.field = mapped_index->second;
245         chktm.pack(arg.new_code);
246      }
247
248      static int32_t idx;
249      static int32_t chktm_idx;
250   };
251
```

STATUS:fixed

Fixed Git Commit:
https://github.com/armoniax/eos-vm/commit/755648758472bce5c146f671e900a4c53bc386c0

Confidential

4.8 Possible illegal characters

Code path:libraries/chain/name.cpp

Risk description:

to_string initializes the charmap character to ".12345abcdefghijklmnopqrstuvwxyz". However, when the set function passes in characters, it makes a length judgement on the str passed in, and does not make a judgement on illegal characters, which may cause security problems when illegal characters are passed in.

```
   void name::set( std::string_view str ) {
      const auto len = str.size();
      EOS_ASSERT(len <= 13, name_type_exception, "Name is longer than 13
 characters (${name}) ", ("name", std::string(str)));
      value = string_to_uint64_t(str);
      EOS_ASSERT(to_string() == str, name_type_exception,
                 "Name not properly normalized (name: ${name}, normalize
d: ${normalized}) ",
                 ("name", std::string(str))("normalized", to_string()));
   }

   // keep in sync with name::to_string() in contract definition for nam
e
   std::string name::to_string()const {
     static const char* charmap = ".12345abcdefghijklmnopqrstuvwxyz";

      std::string str(13,'.');

      uint64_t tmp = value;
      for( uint32_t i = 0; i <= 12; ++i ) {
         char c = charmap[tmp & (i == 0 ? 0x0f : 0x1f)];
         str[12-i] = c;
         tmp >>= (i == 0 ? 4 : 5);
      }

      boost::algorithm::trim_right_if( str, []( char c ){ return c == '.
'; } );
      return str;
   }
```

STATUS:fixed

Fixed Git Commit:

https://github.com/armoniax/amax.meta.chain/commit/fcd4461da49d6ecdd0d467fe926700dc42920b4b

Confidential

4.9 Design issue - block_num type should be set to uint64_t

Code path:libraries/chain/block_header_state.cpp

Risk description:

The block_num is 32 bits, since one block per second, after calculating almost 137 years, will cause the block num to overflow, it is recommended that block_num be set to 64 bits.

```
auto itr = producer_to_last_produced.find( proauth.producer_name );
if( itr != producer_to_last_produced.end() ) {
  EOS_ASSERT( itr->second < (block_num+1) - num_prev_blocks_to_confirm, producer_double_confirm,
              "producer ${prod} double-confirming known range",
              ("prod", proauth.producer_name)("num", block_num+1)
              ("confirmed", num_prev_blocks_to_confirm)("last_produced", itr->second) );
}

result.block_num                                = block_num + 1;
result.previous                                 = id;
result.timestamp                                = when;
result.confirmed                                = num_prev_blocks_to_confirm;
result.active_schedule_version                  = active_schedule.version;
result.prev_activated_protocol_features         = activated_protocol_features;

result.valid_block_signing_authority            = proauth.authority;
result.producer                                 = proauth.producer_name;

result.blockroot_merkle = blockroot_merkle;
```

STATUS: Ignore

4.10 Unreasonable use of space

Code path:libraries/fc/secp256k1/upstream/src/hash.h

Risk description:

secp256k1_sha256_t struct, s[32] needs to be modified to s[8], s[] array, only 8 used, no more space needed, needs to be modified to the actual space used.

```
typedef struct {
    uint32_t s[32];
    uint32_t buf[16]; /* In big endian */
    size_t bytes;
} secp256k1_sha256_t;

static void secp256k1_sha256_initialize(secp256k1_sha256_t *hash) {
    hash->s[0] = 0x6a09e667ul;
    hash->s[1] = 0xbb67ae85ul;
    hash->s[2] = 0x3c6ef372ul;
    hash->s[3] = 0xa54ff53aul;
```


Confidential

```
    hash->s[4] = 0x510e527ful;
    hash->s[5] = 0x9b05688cul;
    hash->s[6] = 0x1f83d9abul;
    hash->s[7] = 0x5be0cd19ul;
    hash->bytes = 0;}
```

STATUS：Ignore

4.11 Possible performance problems caused by too much data

Code path:libraries/chain/merkle.cpp

Risk description:

The merkle function is implemented to determine whether the incoming ids parameter is not empty, but does not limit the maximum value of the incoming parameters, which may lead to lower performance in processing data if the incoming data is larger.

```
digest_type merkle(vector<digest_type> ids) {
   if( 0 == ids.size() ) { return digest_type(); }

   while( ids.size() > 1 ) {
      if( ids.size() % 2 )
         ids.push_back(ids.back());

      for (size_t i = 0; i < ids.size() / 2; i++) {
         ids[i] = digest_type::hash(make_canonical_pair(ids[2 * i], ids
[(2 * i) + 1]));
      }

      ids.resize(ids.size() / 2);
   }

   return ids.front();
}
```

STATUS：Ignore

5.Audit Conclusion

Numen Cyber Labs conducted a comprehensive audit of the code security and business logic security of the Armonia public chain using black-box testing, white-box testing, and grey-box testing methods.
The audit result is PASSED.

Confidential

6.Disclaimer

Note that this audit does not give any warranties on finding all possible security issues of the given public blockchain, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of public blockchain. Last but not least, this security audit should not be used as investment advice.

Numen Cyber

Confidential

Numen Cyber is a Singapore-based cybersecurity company that specializes in Web3 Security Solutions. Our team of world-class security experts has discovered critical vulnerabilities in some of the world's most well-known Web3 projects, such as Aptos, Sui, Eos, Ripple, and Tron.

Numen Cyber offers a comprehensive range of Web3 security services that cover all stages of a Web3 project's lifecycle. Our Cyber Labs team provides security audits for Smart Contracts, Public Blockchains, Smart Wallets, and Exchanges, ensuring that your Web3 project is secure from the ground up. We also offer on-chain Smart Contract Threat Detection and Response, Web3 Security Situational Awareness, Digital Currency Tracing, and Web3 Threat Intelligence to safeguard the digital asset security of Web3 projects and their users.

At Numen Cyber, we believe that creating a safer cyberspace is a shared responsibility. Let us work together to ensure the security of your Web3 project and protect your users' digital assets.

Official Website
www.numencyber.com/

Contact
+65 6355 5555

Email
contact@numencyber.com