Proton Chain Security Assessment

Metallicus Inc.

15 December 2020
Version: 1.1

Presented by:
Kudelski Security Research Team
Kudelski Security – Nagravision SA

Corporate Headquarters
Kudelski Security – Nagravision SA
Route de Genève, 22-24
1033 Cheseaux sur Lausanne
Switzerland

DOCUMENT PROPERTIES

Version:	1.1
File Name:	Proton Code Review v1.1_FINAL.docx
Publication Date:	15 December 2020
Confidentiality Level:	Confidential
Document Owner:	Mikael Björn
Document Recipient:	Metallicus Inc.
Document Status:	Approved

TABLE OF CONTENTS

EXECUTIVE SUMMARY 5
1.1 Engagement Limitations 5
1.2 Engagement Analysis 5
1.3 Observations 6
1.4 Issue Summary List 8
2. METHODOLOGY 9
2.1 Kickoff 9
2.2 Ramp-up 9
2.3 Review 9
2.4 Reporting 10
2.5 Verify 11
2.6 Additional Note 11
3. TECHNICAL DETAILS 12
3.1 Uninitialized Permissions 12
3.2 Missing checks on arithmetic operations 13
3.3 Free resources and SYS-token leakage 14
3.4 Insufficient permission checks for voting 16
3.5 Empty implementation of public action 17
3.6 Inline definition of core symbols 18
3.7 Faulty name check on account creation 19
3.8 Actual maximum RAM size differs from documentation 19
3.9 DApp config value check differs from its error 20
3.10 Missing RAM for DApp account 21
APPENDIX A: ABOUT KUDELSKI SECURITY 22
APPENDIX B: DOCUMENT HISTORY 23
APPENDIX C: SEVERITY RATING DEFINITIONS 24

TABLE OF FIGURES

Figure 1 Issue Severity Distribution 6
Figure 2 Methodology Flow 9

EXECUTIVE SUMMARY

Kudelski Security ("Kudelski"), the cybersecurity division of the Kudelski Group, was engaged by Metallicus Inc. ("Metallicus") client to conduct an external security assessment in the form of a Security Assessment of the Proton Chain application.

Proton Chain is a new public blockchain and smart contract platform designed for both consumer applications and peer-to-peer payments. It is built around a secure identity and financial settlements layer that allows users to directly link real identity and fiat accounts, pull funds and buy crypto.

The source code for the project was supplied by Metallicus through the GitHub repository at https://github.com/ProtonProtocol and specifically under the proton.contracts project.

The assessment was conducted remotely by the Kudelski Security Team. The tests took place from September 19, 2020 to November 6, 2020 and focused on the following objectives:

1. Perform manual source code review of C++ and other used languages (~16,500 lines)
2. Perform review of smart contract platform for security vulnerabilities and logic errors
3. Perform review of third-party libraries used in systems (automated test)
4. Perform analysis of any custom encryption used in development of platform and verification of third-party encryption libraries
5. Analyze the contracts layer in proton contracts to ensure that the logic is sound, that there are no security issues, and that the platform is free from vulnerabilities.

This report summarizes the tests performed and findings in terms of strengths and weaknesses. It also contains detailed descriptions of the discovered vulnerabilities, steps the Kudelski Security Teams took to exploit each vulnerability, and recommendations for remediation.

At the time of the final release of this report all findings and observations within this report were fixed and remediated.

1.1 Engagement Limitations

During the initial phase of the project, we recognized that the main code base is comprised of a fork of the EOSIO block chain project with the tag v1.9.1. This code was de-scoped from the project as Metallicus recognized the fact that the EOSIO project is well used and therefore could be seen as already reviewed. The project did use the forked version in the Proton Chain repository as a reference when needed.

Out of Scope: Review of existing EOSIO code from tag v1.9.1

1.2 Engagement Analysis

The Kudelski Security team conducted an audit of the Proton Chain system that included the smart contracts created to support the Proton chain. The project team has had good interactions with the development team and all findings have been disclosed in advance.

As a result of our work, we identified **0 High**, **0 Medium**, **2 Low**, and **8 Informational** findings.

The findings during the review have been better than other projects of similar size and complexity. The findings are mostly of the character of: initialization issues, arithmetic operational checks, resource allocation and code clarity on permission checks. The reason for the rather low severity of the findings are due to the very good discussions and walkthroughs with the development team as these items were discussed to cause no significant security risks to the underlying system.

The general observations are also something that have been discussed and we feel that the development team has been responsive in taking into considerations any changes needed for the further development.

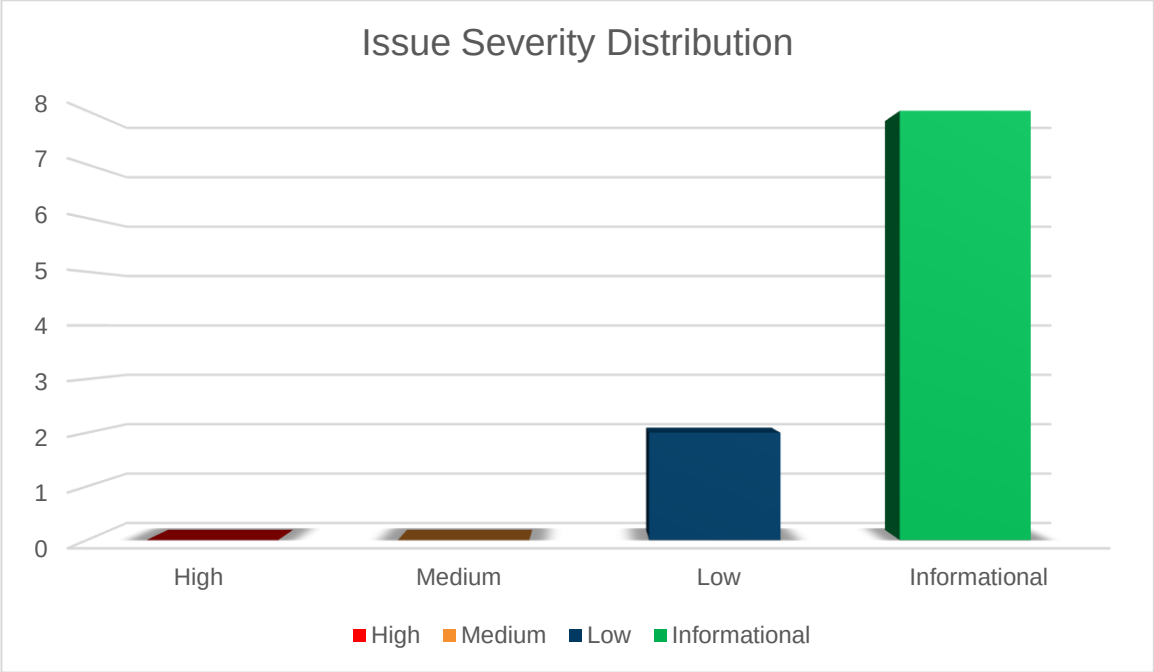

Figure 1 Issue Severity Distribution

1.3 Observations

There are a large number of common observations about the code that are non-impacting, of high volume, are too generic to be included in the specific findings in the technical findings chapter in this report.

Comments are largely missing on the added code compared to the code from the EOSIO project, which is the base for the Proton Chain. The lack of comments could lead to risk in the future as the project grows in maintainers.

The code base is in a development phase and we see what would be expected

- The code contains several TODOs.
- The code contains unused / dead code.
- The code contains out-commented code.

- The code contains spelling errors in error messages and variable names.
- Tests are not updated for added and changed functionality.

The repository “proton” is, at the moment of review, 220 commits after the EOSIO project repository master branch. We would suggest that a plan to incorporate updates and bug fixes from the EOSIO to the proton project is created and executed*. If this is not done, there will be a high probability that exploitable bugs in the EOSIO project will be exploited in the Proton Project.* If bug fixes in EOSIO are not duplicated in the proton repository, security issues and their severity will mirror those resolved by the bug fixes. If unique code is added to the EOSIO branch to support proton, we recommend an audit of any code added at a later time.

Observations that may cause issues in the future, but are just general observations for now:

- We noticed that not all added contracts have an associated Ricardian contract.
- The pushr.proton contract performs checks of input data, but nothing else is performed. This makes the contract consume resources but not producing anything. As the name suggest the contract will perform a function to push a token onto the chain but it is not clear from the code.
- The contract payr.proton is mentioned in the documentation but there is no code for this in the repository proton.contracts. Either the documentation is incorrect or the code for the contract is not yet created.
- In the reviewed code base, the user_resources table is defined twice, but the effect of this is unknown. Either it sets up two completely different tables in memory, or it refers to the same table. This will either work as intended or create a problem when trying to debug the application as the names will be referring to two different memory segments. This solution will also make the code more complex to maintain, as the names will introduce confusion as to which table is referenced.
- We have seen a system for whitelisted actions, cryptocurrencies and transfer receivers that is defined but not currently used by the block chain application.
- The mixed use of SYS and XPR tokens in the code, especially when it comes to voting on block producers, makes it more difficult to review and maintain the code. During the review, we were given the impression that users will not be exposed to the SYS token. This is wise but from a code maintenance perspective, it may be better to be very thorough in documenting why and when a specific token is used.
- There is an irregular use of EOSIO functions in the code base. For example _self vs get_self() are used interchangeable throughout the code. We suggest that one is chosen and used throughout the project. Many larger open source projects use a have a code standard guide and a test suite to see that code to be included adheres to the standard. It may be an overkill to create something like that for the Proton project but a coding guide will make the maintenance of the code base easier in the future.
- Table fields and variables of the bool type are often assigned int values even though true and false are supported.

- We also see some conditional expressions that compare similarity to the value 1 instead of true.
- One of the things that we see need some further development is the name bidding system. When creating an account, the part for name bidding is commented out, which in practice puts a stop to the entire bidding system. This means that there is no restriction on name, more than length and character set. Code to bid on names can in practice not be reached because rights are required to perform such an action, which will probably not be given to any account. Code is run at each block update to update the name-bidding table, which is always empty.
- The system for distribution to committee members has possible problems as the resumption of the process presupposes that the account next in turn remains. If the account is deleted, subsequent accounts lose their dividend.
- No new third-party dependencies appear to have been added other than the ones EOSIO require, which is good, as this does not add any further third party library issues.

1.4 Issue Summary List

ID	SEVERITY	FINDING
protoncpp#1	**Low**	
protoncpp#2	**Low**	
protoncpp#3	**Informational**	
protoncpp#4	**Informational**	
protoncpp#5	**Informational**	
protoncpp#6	**Informational**	
protoncpp#7	**Informational**	
protoncpp#8	**Informational**	
protoncpp#9	**Informational**	
protoncpp#10	**Informational**	

2. METHODOLOGY

Kudelski Security uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Figure 2 Methodology Flow

2.1 Kickoff

The project is kicked all of the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

2.2 Ramp-up

Ramp-up consists of the activities necessary to gain proficiency on the particular project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

2.3 Review

The review phase is where a majority of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project

3. Assessment of the cryptographic primitives used
4. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general list and not comprehensive, meant only to give an understanding of the issues we are looking for.

Cryptography

We analyzed the cryptographic primitives and components as well as their implementation. We checked in particular:

- Matching of the proper cryptographic primitives to the desired cryptographic functionality needed
- Security level of cryptographic primitives and their respective parameters (key lengths, etc.)
- Safety of the randomness generation in general as well as in the case of failure
- Safety of key management
- Assessment of proper security definitions and compliance to use cases
- Checking for known vulnerabilities in the primitives used

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

2.4 Reporting

Kudelski Security delivers a preliminary report in PDF format that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project as a whole. We may conclude that the overall risk is low, but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We not only report security issues identified but also informational findings for improvement categorized into several buckets:

- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps, that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

2.5 Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes withing a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

2.6 Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessment the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. These is a solid baseline for severity determination. Information about the severity ratings can be found in **Appendix C** of this document.

3. TECHNICAL DETAILS

This section contains the technical details of our findings as well as recommendations for improvement.

3.1 Uninitialized Permissions

Finding ID: protoncpp#1

Severity: **Low**

Status: **Remediated**

Description

When permission is set with the setperm() function, not all permission types are initialized to 0, which means that the following permissions may not be initialized before use:

- delegate
- undelegate
- sellram
- buyram

Proof of Issue

Filename: : eosio.proton.cpp

Beginning Line Number: 89

```
perm.emplace( _self, [&]( auto& p ){
    p.acc = acc;
    p.createacc = 0;
    p.vote = 0;
    p.regprod = 0;
    p.regproxy = 0;
    p.setcontract = 0;
    p.namebids = 0;
    p.rex = 0;
    for (auto it=perms.begin(); it!=perms.end(); ++it){
        if(it->first == "createacc") { p.createacc = it->second; }
        if(it->first == "vote") { p.vote = it->second; }
        if(it->first == "regprod") { p.regprod = it->second; }
        if(it->first == "regproxy") { p.regproxy = it->second; }
        if(it->first == "setcontract") { p.setcontract = it->second; }
        if(it->first == "namebids") { p.namebids = it->second; }
        if(it->first == "rex") { p.rex = it->second; }
        if(it->first == "delegate") { p.delegate = it->second; }
        if(it->first == "undelegate") { p.undelegate = it->second; }
        if(it->first == "sellram") { p.sellram = it->second; }
        if(it->first == "buyram") { p.buyram = it->second; }
    }
});
```

Severity and Impact Summary

A user may be accidentally assigned rights or may be inadvertently blocked from certain actions.

Recommendation

Initialize all permissions to 0.

References

N/A

3.2 Missing checks on arithmetic operations

Finding ID: protoncpp#2

Severity: **Low**

Status: **Remediated**

Description

With few exceptions, the code lacks controls for underflow, overflow and division with zero.

Proof of Issue

Below are some examples of this from cfund.proton.cpp included.

Filename: cfund.proton.cpp (applies to several files)

Beginning Line Number: 95

```
  ACTION cfundproton::claimreward( const name& account){
        require_auth( account );

        conf config(_self, _self.value);
        _gstate = config.exists() ? config.get() : global{};

        users users_( _self, _self.value );
        auto itr = users_.find(account.value);

        check(itr != users_.end(), "Committee not found.");
        check ( current_time_point().sec_since_epoch() - itr->lastclaim  > claimInterval, "You
last claim was less than 24h ago.");
        check ( itr->claimamount > 0, "Nothing to claim.");
        check ( itr->active == 1, "committee member is not activated.");

        auto sendClaim = action(
            permission_level{ get_self(), "active"_n },
            TOKEN_CONTRACT,
            "transfer"_n,
            std::make_tuple(  get_self(),  itr->account,  asset(itr->claimamount,  TOKEN_SYMB),
std::string("Committee claim reward") )
        );

        sendClaim.send();

        _gstate.notclaimed -= itr->claimamount;

        users_.modify(itr, account, [&](auto& s) {
            s.claimamount = 0;
            s.lastclaim = current_time_point().sec_since_epoch();
        });

        config.set( _gstate, _self );
    }

// Unsigned integer division
uint64_t claim =  _gstate.processQuant / _gstate.totalausr ;
_gstate.processed += claim;
```

Severity and Impact Summary

This can lead to us getting incorrect values that are not detected.

Recommendation

It should be checked or commented on with a justification as to why the problem may not occur.

References

N/A

3.3 Free resources and SYS-token leakage

Finding ID: protoncpp#3

Severity: **Informational**

Status: **Remediated**

Description

When registering a DApp, this resource is allocated such that the account has at least a defined minimum amount of CPU, NET and RAM. The account that is assigned is the account that initiates the transaction and no further verification of the account is made. Nothing prevents multiple calls to the function of an account, unless the account is blocked from performing the transaction.

Given certain circumstances, a user can use this to obtain all SYS tokens in the blockchain.

Proof of Issue

See also proof of issue for the discovery Uninitialized Permissions.

Filename: eosio.proton.cpp, delegate_bandwidth.cpp

Beginning Line Number: 401, 398

```
void eosioproton::dappreg(name account){
    ...
    permissions perm( _self, _self.value );
    auto uperm = perm.find( account.value );

    if ( uperm != perm.end() ) {
        check ( uperm->setcontract != 4 , "Sorry, account banned." );
        perm.modify( uperm, _self, [&]( auto& p ){
             p.setcontract = 1;
        });
    } else {
        perm.emplace( _self, [&]( auto& p ){
            // Set permission info
            ...
        });
    }
    ...
    if (_dcstate.dappcpu > cpu.amount)
        addCpu = _dcstate.dappcpu - cpu.amount;
    if (_dcstate.dappnet > net.amount)
        addNet = _dcstate.dappnet - net.amount;

    if (addRam > 10){
        auto act = action(
            permission_level{ "wlcm.proton"_n, "newacc"_n },
            "eosio"_n,
            "buyrambytes"_n,
            std::make_tuple( "wlcm.proton"_n, account, addRam )
        );
        act.send();
    }

    if (addCpu + addNet > 0) {
        auto act = action(
            permission_level{ "wlcm.proton"_n, "newacc"_n },
            "eosio"_n,
            "delegatebw"_n,
            std::make_tuple( "wlcm.proton"_n,
                account,
                asset(addNet, symbol("SYS", 4)),
                asset(addCpu, symbol("SYS", 4)),
                0 )
        );
        act.send();
    }
}

void system_contract::undelegatebw( const name& from, const name& receiver,
                                    const    asset&    unstake_net_quantity,    const    asset&
unstake_cpu_quantity )
{
    ...
    check (system_contract::checkPermission(from, "undelegate")==1, "You are not authorised to
undelegate.");

    changebw( from, receiver, -unstake_net_quantity, -unstake_cpu_quantity, false);
`
```

Severity and Impact Summary

If a user does not have the undelegate right, the damage is only that arbitrary users can access free resources - the impact is low. For multiple calls, the user does not receive more resources, but the user receives resources such that you have what is specified by the DApp configuration.

With the right conditions, the impact of the discovery is great.

To carry out the attack, the account needs the right undelegate. When a user applies to register a DApp and receives resources according to the dappconf table, the user can then submit an undelegate transaction to unstake their CPU and NET resources for SYS tokens.

At this time, when the user no longer has any CPU and NET resources, the user can apply to register a DApp again, as no checks are made on this, to get new resources, which can unstake again. This can be done an infinite number of times, or until it is noticed and the user is blocked from performing the transaction.

A mitigating factor is that the payment is not made immediately, but is postponed for 3 days. However, this is still scriptable, and can be done regularly until the rights are revoked, which (apparently based on code) is done manually.

A user does not necessarily have to be assigned the undelegate right. According to the discovery Uninitialized Permissions, undefined behavior occurs in the uninitialized fields. If the fields are not reset, the fields can be initialized with random data, which can give the user the necessary right. However, this is very unlikely, as the random data needs to have the value 1 on the space of 8 bits.

Note that the attack also works given the right sellram.

Recommendation

Check if an account has already registered a DApp.

References

N/A

3.4 Insufficient permission checks for voting

Finding ID: protoncpp#4

Severity: **Informational**

Status: **Remediated**

Description

The voting functions do not check all rights. The voter producer feature is the feature that allows the user to vote with XPR tokens. The function lacks a control over the right to vote. The function that allows voting with SYS tokens - voteprodsys - performs REX operations without verifying that the user has the right rex.

Proof of Issue

Filename: voting.cpp

Beginning Line Number: 228

```
void system_contract::voteproducer( const name& voter_name, const name& proxy, const
std::vector<name>& producers ) {
   require_auth( voter_name );
   // Missing vote permission check
   update_xpr_votes( voter_name, voter_name, proxy, producers, true );
}
 void system_contract::voteprodsys( const name& voter_name, const name& proxy, const
std::vector<name>& producers ) {
   require_auth( voter_name );
   check (checkPermission(voter_name, "vote")==1, "You are not authorised to Vote");
   // Missing REX-permission check

   update_votes( voter_name, proxy, producers, true );
   auto rex_itr = _rexbalance.find( voter_name.value );
   if( rex_itr != _rexbalance.end() && rex_itr->rex_balance.amount > 0 ) {
      check_voting_requirement( voter_name, "voter holding REX tokens must vote for at least 21
producers or for a proxy" );
```

Severity and Impact Summary

Actions can be performed by users who do not really have that right to do it.

Recommendation

Add rights checks for these features.

References

N/A

3.5 Empty implementation of public action

Finding ID: protoncpp#5

Severity: **Informational**

Status: **Remediated**

Description

The function is defined and marked as action but is not implemented.

Proof of Issue

Filename: eosio.system.cpp

Beginning Line Number: 418

```
void native::setcode( const name& account, uint8_t vmtype, uint8_t vmversion, const
std::vector<char>& code ) {
  check (system_contract::checkPermission(account, "setcontract")==1, "You are not authorised
to setcode.");
```

Severity and Impact Summary

The feature is publicly available as an action. It may give users a misconception that the feature exists. The user should reasonably be able to rely on the function performing the intended work - for example updating its contract code - but will then not be notified that nothing has happened.

Recommendation

Implement the function or delete the code.

References

N/A

3.6 Inline definition of core symbols

Finding ID: protoncpp#6

Severity: **Informational**

Status: **Remediated**

Description

There are definitions for core symbols inside the code. Should the definition of core symbol change, this can cause problems that become difficult to detect.

Proof of Issue

Filename: eosio.proton.cpp

Beginning Line Number: 480

```
auto act = action(
    permission_level{ "wlcm.proton"_n, "newacc"_n },
    "eosio"_n,
    "delegatebw"_n,
    std::make_tuple( "wlcm.proton"_n,
        account,
        asset(addNet, symbol("SYS", 4)),
        asset(addCpu, symbol("SYS", 4)),
        0 )
);
```

Severity and Impact Summary

This can lead to problems if core_symbol() is changed.

Recommendation

Use core_symbol().

References

N/A

3.7 Faulty name check on account creation

Finding ID: protoncpp#7

Severity: **Informational**

Status: **Remediated**

Description

Verification of the correct name differs from EOSIO's name management. Names in EOSIO are encoded as 12 * 5 bits with a remainder of 4 bits. In the name verification, the rest (4 bits) are handled twice, which leads to illegal names being accepted.

Proof of Issue

Filename: eosio.system.cpp

Beginning Line Number: 340

```
uint64_t tmp = newact.value >> 4;
...
static const char* charmap = ".12345abcdefghijklmnopqrstuvwxyz";
...
for( uint32_t i = 0; i < 12; ++i ) {
    ...
    char c = charmap[tmp & (i == 0 ? 0x0f : 0x1f)];
    ...
}
...
```

Severity and Impact Summary

Example of accepted, incorrect name: "asd...k"

Recommendation

Validate the name correctly.

References

https://github.com/EOSIO/eosio.cdt/blob/master/libraries/eosiolib/core/eosio/name.hpp

3.8 Actual maximum RAM size differs from documentation

Finding ID: protoncpp#8

Severity: **Informational**

Status: **Remediated**

Description

The system checks that the minimum value of the amount of RAM as standard for DApps does not exceed 16GB RAM. This check is incorrect when the comparison value is less than 16GB, as a factor of the value is 1014 and not 1024.

Proof of Issue

Filename: eosio.proton.cpp

Beginning Line Number: 392

```
check (ram < uint64_t(16*1024*1024)*1014 , "To much RAM");
```

Severity and Impact Summary

The implemented value differs from the documentation.

Recommendation

Change the value so that it corresponds to the documented value.

References

3.9 DApp config value check differs from its error

Finding ID: protoncpp#9

Severity: **Informational**

Status: **Remediated**

Description

If only one of the values is positive, the remaining values can be zero. These values are then added to the table, which affects the registration of DApp accounts. This check should check for conjunction instead of disjunction.

Proof of Issue

Filename: eosio.proton.cpp

Beginning Line Number: 391

```
check (ram > 0 || cpu > 0 || net > 0, "Action values should be positive
numbers");
```

Severity and Impact Summary

Unexpected configuration update behavior.

Recommendation

Change the disjunction to conjunction or change the error message so that both parts agree on the behavior.

References

3.10 Missing RAM for DApp account

Finding ID: protoncpp#10

Severity: **Informational**

Status: **Remediated**

Description

When the user is to register a DApp, the system allocates resources to the user, such that accounts have access to a minimum amount of RAM. When checking to determine if the account lacks RAM, it is checked that the amount needed is more than 10 bytes, instead of the expected 0.

Proof of Issue

Filename: eosio.proton.cpp

Beginning Line Number: 462

```
uint64_t ram = ures->ram_bytes;
...

auto addRam = _dcstate.dappram - ram;
...

if (addRam > 10){
    auto act = action(
        permission_level{ "wlcm.proton"_n, "newacc"_n },
        "eosio"_n,
        "buyrambytes"_n,
        std::make_tuple( "wlcm.proton"_n, account, addRam )
    );
    act.send();
```

Severity and Impact Summary

A registered DApp may be allocated less RAM than expected.

Recommendation

Justify the choice of 10 if it is conscious, otherwise the value should be corrected to the expected value 0.

References

APPENDIX A: ABOUT KUDELSKI SECURITY

Kudelski Security is an innovative, independent Swiss provider of tailored cyber and media security solutions to enterprises and public sector institutions. Our team of security experts delivers end-to-end consulting, technology, managed services, and threat intelligence to help organizations build and run successful security programs. Our global reach and cyber solutions focus is reinforced by key international partnerships.

Kudelski Security is a division of Kudelski Group. For more information, please visit https://www.kudelskisecurity.com.

Kudelski Security

route de Genève, 22-24

1033 Cheseaux-sur-Lausanne

Switzerland

Kudelski Security

5090 North 40th Street

Suite 450

Phoenix, Arizona 85018

APPENDIX B: DOCUMENT HISTORY

VERSION	STATUS	DATE	AUTHOR	COMMENTS
0.1	Draft	10 November 2020	Mikael Björn	Initial Draft
0.5	Draft	12 November 2020	Mikael Björn	Internal QA Draft
0.6	Draft	13 November 2020	Scott Carlson	Customer Draft
0.9	Proposal	2 December 2020	Mikael Björn	Customer Proposal
1.0	Final	4 December	Mikael Björn	Revised
1.1	Final	14 December	Mikael Björn	Final

REVIEWER	POSITION	DATE	VERSION	COMMENTS
Scott Carlson	Head of Blockchain	13 November 2020	0.6	Final Draft
Scott Carlson	Head of Blockchain	4 December 2020	0.6	Final Draft
Scott Carlson	Head of Blockchain	15 December 2020	1.1	Final

APPROVER	POSITION	DATE	VERSION	COMMENTS
Scott Carlson	Head of Blockchain	13 November 2020	0.6	Final Draft
Scott Carlson	Head of Blockchain	4 December 2020	0.6	Final Draft
Scott Carlson	Head of Blockchain	15 December 2020	1.1	Final

APPENDIX C: SEVERITY RATING DEFINITIONS

Kudelski Security uses a custom approach when determining criticality of identified issues. This is meant to be simple and fast, providing customers with a quick at a glance view of the risk an issue poses to the system. As with anything risk related, these findings are situational. We consider multiple factors when assigning a severity level to an identified vulnerability. A few of these include:

- Impact of exploitation
- Ease of exploitation
- Likelihood of attack
- Exposure of attack surface
- Number of instances of identified vulnerability
- Availability of tools and exploits

SEVERITY	DEFINITION
High	The identified issue may be directly exploitable causing an immediate negative impact on the users, data, and availability of the system for multiple users.
Medium	The identified issue is not directly exploitable but combined with other vulnerabilities may allow for exploitation of the system or exploitation may affect singular users. These findings may also increase in severity in the future as techniques evolve.
Low	The identified issue is not directly exploitable but raises the attack surface of the system. This may be through leaking information that an attacker can use to increase the accuracy of their attacks.
Informational	Informational findings are best practice steps that can be used to harden the application and improve processes.