

GDSC 2022

2022

How to be good engineer ?

Agenda

01

1. 자기소개
2. 문제풀이, 대회가 해커가 되기 위해 어떤 도움이 될까
3. 연구? 리서치? 어디서 정보를 얻을 수 있을까
4. 좋은 분석가가 되기 위한 몇 가지 이야기
5. 여담

Self Introductions

02

자기소개

Self Introductions

Haechi Labs

- Email : jeremy@haechi.io
- Age : 25
- aka : skwid
- Role
 - Blockchain security researcher
 - Malware analyst
 - Reverse Engineer



HAECHI'

Self Introductions

Reverse Engineer in SG

- CodeGate
- HITCON
- SECCON
- DEFCON Finalist (But COVID-19..)
- LINE CTF
- etc..

Career

- 0-Day, 1-Day Researcher (2016 ~ 2020)
- Developer (LLVM & AI, 2020 ~ 2022)
- Blockchain security engineer (2022 ~)



CTF? Wargame?

03

문제풀이, 대회가 해커가 되기 위해 어떤 도움이 될까

CTF? Wargame?

CTF ?

- <https://ctftime.org/>
- Can find real-time ctf from ctftime
- Well known : Defcon (US), Seccon (JP) ..
- Last weekend : Hack.lu 2022
- Now : N1CTF 2022
 - I have to participate after end of this conference



CTF? Wargame?

Study Materials for Researchers

maybe, we can get flag with some revision



rkm0959 2022.09.25.

i'll let skwid finish this

you have to deploy bytecode in the create2'ed contract

you can do this with assembly

```
bytes memory bytecode = hex"6080...."  
assembly {  
    addr := create(0, add(bytecode, 0x20), mload(bytecode))  
}
```



skwid 2022.09.25.

```
```import { ethers } from "hardhat";  
import { Contract, ContractFactory, Signer } from "ethers";
import { text } from "stream/consumers";

let Factory : ContractFactory;
let FactoryAssembly : ContractFactory;
```



# CTF? Wargame?

## Study Materials for Researchers

### Web

Imageflare ✓  
460

Mudbox ✓  
499

Mini Realworld ✓  
499

BitTrader  
500

Imageflare 2.0 ✓  
500

### Crypto

Euclid  
500

### Reversing

# CTF? Wargame?

CTF is useful for keeping up with trends

- Hardware, IoT Security ..
- Browser Security ..
- Blockchain Security ..
- etc ..

For Example ...

- Paradigm CTF ?
- ...



# CTF? Wargame?

## Example : Blockchain challenge from CTF

- <https://eips.ethereum.org/EIPS/eip-1014>

### EIP-1014: Skinny CREATE2 ↵

Author	Vitalik Buterin
Status	Final
Type	Standards Track
Category	Core
Created	2018-04-20

#### Table of Contents

- Specification
- Motivation
- Rationale
- Clarifications
- Examples

#### Specification

Adds a new opcode (`CREATE2`) at `0xf5`, which takes 4 stack arguments: endowment, memory\_start, memory\_length, salt. Behaves identically to `CREATE` (`0xf0`), except using `keccak256(0xff ++ address ++ salt ++ keccak256(init_code))[12:]` instead of the usual sender-and-nonce-hash as the address where the contract is initialized at.

The `CREATE2` has the same `gas` schema as `CREATE`, but also an extra `hashcost` of `GSHA3WORD * ceil(len(init_code) / 32)`, to account for the hashing that must be performed. The `hashcost` is deducted at the same time as memory-expansion gas and `CreateGas` is deducted *before* evaluation of the resulting address and the execution of `init_code`.

- `0xff` is a single byte,
- `address` is always 20 bytes,
- `salt` is always 32 bytes (a stack item).

The preimage for the final hashing round is thus always exactly 85 bytes long.

The `coredevcall` at 2018-08-10 decided to use the formula above

# CTF? Wargame?

## CREATE2 can create the same contract address

- But, how to abuse this specification?
- CTFs and Wargames give us learning opportunities

### EIP-1014: Skinny CREATE2 ↔

Author	Vitalik Buterin
Status	Final
Type	Standards Track
Category	Core
Created	2018-04-20

#### Table of Contents

- Specification
- Motivation
- Rationale
- Clarifications
- Examples

#### Specification

Adds a new opcode (`CREATE2`) at `0xf5`, which takes 4 stack arguments: endowment, memory\_start, memory\_length, salt. Behaves identically to `CREATE` (`0xf0`), except using `keccak256(0xff ++ address ++ salt ++ keccak256(init_code))[12:]` instead of the usual sender-and-nonce-hash as the address where the contract is initialized at.

The `CREATE2` has the same gas schema as `CREATE`, but also an extra `hashcost` of `GSHA3WORD * ceil((len(init_code) / 32))`, to account for the hashing that must be performed. The `hashcost` is deducted at the same time as memory-expansion gas and `CreateGas` is deducted *before* evaluation of the resulting address and the execution of `init_code`.

- `0xff` is a single byte.
- `address` is always 20 bytes.
- `salt` is always 32 bytes (a stack item).

The preimage for the final hashing round is thus always exactly 85 bytes long.

The `create2` call at 2018-08-10 decided to use the formula above

# Information for research

04

연구? 리서치? 어디서 정보를 얻을 수 있을까

# Information for research

## How can we get information before we start researching



Twitter is the best place to follow up on the latest security incidents, attack techniques, tech blogs, and more



A blog organized by a famous team of hackers makes it easy to get advanced technical issues



We can keep track of what's newly added components or services through GitHub commit logs

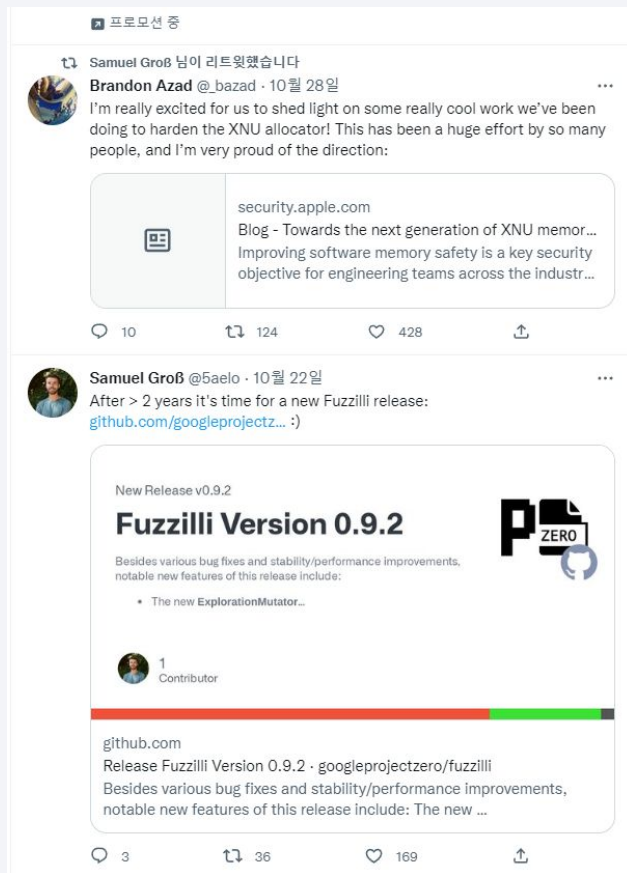


In addition, information can be obtained through various blogs, for example, the 360 Security Blog

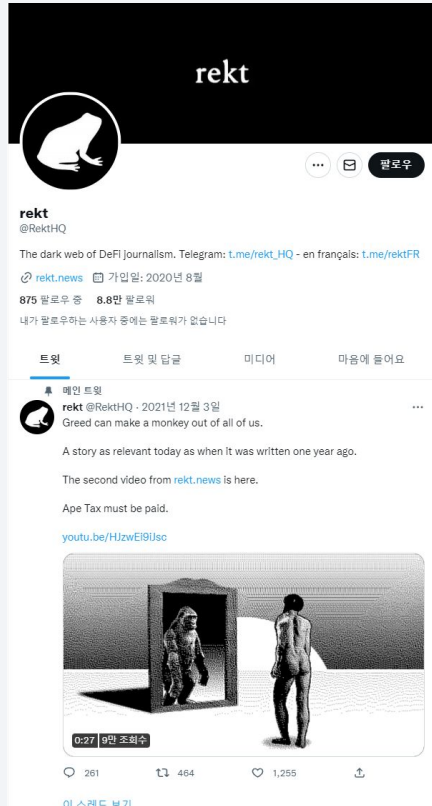
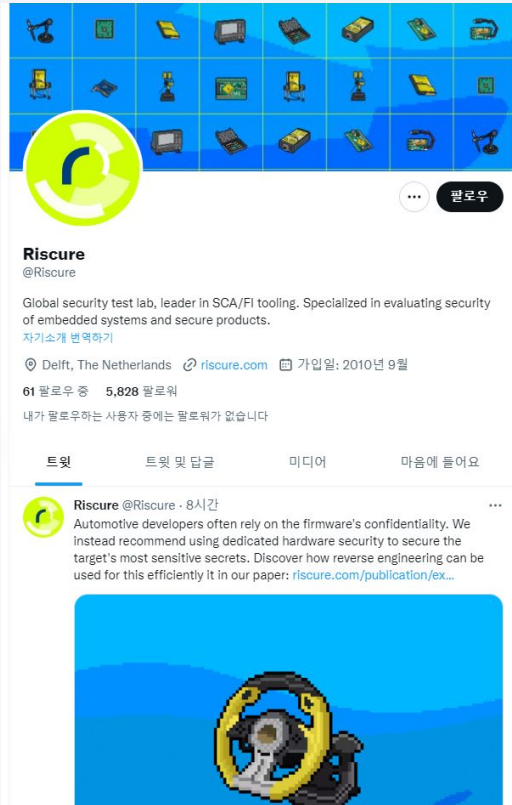
# Twitter ?

## Many hackers uses Twitter platform

- They even record their research progress and achievements on Twitter.
- There are many hackers, each with their own special skills (like saelo)
- Feedback on the latest research trends is really fast.



# Twitter ?





# Blogs ?

## Famous research teams have their own blogs

- **ProjectZero blogspot**
- **360 security blog ..**
- **An 'individual', not a 'team', also has a blog**
  - like saelo's phrack blog
- **ProjectZero monorail**
  - Not sure if this should also be classified as a blog

## Project Zero

News and updates from the Project Zero team at Google

Thursday, October 27, 2022

### RC4 Is Still Considered Harmful

By James Forshaw, Project Zero

I've been spending a lot of time researching Windows authentication implementations, specifically Kerberos. In June 2022 I found an interesting issue number [2310](#) with the handling of RC4 encryption that allowed you to authenticate as another user if you could either interpose on the Kerberos network traffic to and from the KDC or directly if the user was configured to disable typical pre-authentication requirements.

This blog post goes into more detail on how this vulnerability works and how I was able to exploit it with only a bare minimum of brute forcing required. Note, I'm not going to spend time fully explaining how Kerberos authentication works, there's plenty of resources online. For example [this blog post](#) by [Steve Sylfuhs](#) who works at Microsoft is a good first start.

### Background

Kerberos is a very old authentication protocol. The current version (v5) was described in [RFC1510](#) back in 1993, although it was updated in [RFC4120](#) in 2005. As Kerberos' core security concept is using encryption to prove knowledge of a user's credentials the design allows for negotiating the encryption and checksum algorithms that the client and server will use.

Search This Blog

Search

### Pages

- [About Project Zero](#)
- [Working at Project Zero](#)
- [Today "In the Wild"](#)
- [Today Exploit Root Cause Analyses](#)
- [Vulnerability Disclosure FAQ](#)

### Archives

Current issue : #70   Release date : 2021-10-05   Editor : The Phrack Staff	Get targz
Introduction	The Phrack Staff
Phrack Profile on xerub	The Phrack Staff
Attacking JavaScript Engines: A case study of JavaScriptCore and CVE-2016-4622	saelo
Cyber Grand Shellphish	Team Shellphish
VM escape - QEMU Case Study	Mehdi Talbi & Paul Fariello
NET Instrumentation via MSIL bytecode injection	Antonio 's4tan' Parata
Twenty years of Escaping the Java Sandbox	Ieu Euvoudum and disk noise
Viewer Discretion Advised: (De)coding an iOS Kernel Vulnerability	Adam Donenfeld
Exploiting Logic Bugs in JavaScript JIT Engines	saelo
Hypervisor Necromancy: Reanimating Kernel Protectors	Aris Thallas
Tale of two hypervisor bugs - Escaping from FreeBSD bhyve	Reno Robert
The Bear in the Arena	xerub
Exploiting a Format String Bug in Solaris CDE	Marco Ivaldi
Segfault.net eulogy	skyper
YouTube Security Scene	LiveOverflow
Title : Attacking JavaScript Engines: A case study of JavaScriptCore and CVE-2016-4622	
Author : saelo	

# Blogs ?

## What is SCM\_RIGHTS?

Linux developers can share file descriptors (fd) from one process to another using the [SCM\\_RIGHTS datagram with the sendmsg syscall](#). When a process passes a file descriptor to another process, [SCM\\_RIGHTS](#) will add a reference to the underlying `file` struct. This means that the process that is sending the file descriptors can immediately close the file descriptor on their end, even if the receiving process has not yet accepted and taken ownership of the file descriptors. When the file descriptors are in the "queued" state (meaning the sender has passed the fd and then closed it, but the receiver has not yet accepted the fd and taken ownership), specialized garbage collection is needed. To track this "queued" state, this [LWN article](#) does a great job explaining [SCM\\_RIGHTS](#) reference counting, and it's recommended reading before continuing on with this blogpost.

## Sending

As stated previously, a unix domain socket uses the syscall `sendmsg` to send a file descriptor to another socket. To explain the reference counting that occurs during [SCM\\_RIGHTS](#), we'll start from the sender's point of view. We start with the kernel function `unix_stream_sendmsg`, which implements the `sendmsg` syscall. To implement the [SCM\\_RIGHTS](#) functionality, the kernel uses the structure `scm_fp_list` for managing all the transmitted `file` structures. `scm_fp_list` stores the list of `file` pointers to be passed.

```
struct scm_fp_list {
 short count;
 short max;
 struct user_struct *user;
 struct file *fp[SCM_MAX_FD];
};
```

`unix_stream_sendmsg` invokes `scm_send` ([af\\_unix.c#L1886](#)) to initialize the `scm_fp_list` structure, linked by the `scm_cookie` structure on the stack.

The creator has not happened: A vulnerability post... (Dec)

- Windows Exploitation Tricks: Relaying DCOM Authent... (Oct)
- Using Kerberos for Authentication Relay Attacks (Oct)
- How a simple Linux kernel memory corruption bug ca... (Oct)
- Fuzzing Closed-Source JavaScript Engines with Cove... (Sep)
- Understanding Network Access in Windows AppContainers (Aug)
- An EPYC escape: Case-study of a KVM breakout (Jun)
- Fuzzing iOS code on macOS at native speed (May)
- Designing sockfuzzer, a network syscall fuzzer for... (Apr)
- Policy and Disclosure: 2021 Edition (Apr)
- Who Contains the Containers? (Apr)
- In-the-Wild Series: October 2020 0-day discovery (Mar)
- Déjà vu-Inerability (Feb)
- A Look at iMessage in iOS 14 (Jan)
- Windows Exploitation Tricks:

# Blogs ?

2308	Fixed	----	2022-May-28	Microsoft	Windows	forshaw	Windows: Credential Guard TGT Renewal Information Disclosure <a href="#">CCProjectZeroMembers</a>
2307	Fixed	----	2022-May-27	Microsoft	Windows	forshaw	Windows: Credential Guard Non-Constant Time Comparison Information Disclosure <a href="#">CCProjectZeroMembers</a>
2306	Fixed	----	2022-May-27	Microsoft	Windows	forshaw	Windows: Credential Guard KerblumGetNtImSupplementalCredential Information Disclosure <a href="#">CCProjectZeroMembers</a>
2305	Fixed	----	2022-May-27	Microsoft	Windows	forshaw	Windows: Credential Guard KerblumCreateApReqAuthenticator Key Information Disclosure <a href="#">CCProjectZeroMembers</a>
2304	Fixed	----	2022-May-27	Microsoft	Windows	forshaw	Windows: Credential Guard Kerberos Change Password EoP <a href="#">CCProjectZeroMembers</a>
2303	Fixed	----	2022-May-27	Microsoft	Windows	forshaw	Windows: Credential Guard Insufficient Checks on Kerberos Encryption Type Use <a href="#">CCProjectZeroMembers</a>
2302	Fixed	----	2022-May-27	Microsoft	Windows	forshaw	Windows: Credential Guard BCrypt Context Use-After-Free EoP <a href="#">CCProjectZeroMembers</a>
2301	Fixed	----	2022-May-27	Microsoft	Windows	forshaw	Windows: Credential Guard ASN1 Decoder Type Confusion EoP <a href="#">CCProjectZeroMembers</a>
2300	Fixed	----	2022-May-27	Google	Chrome	glazunov	Chrome: heap-use-after-free in LinkToTextMenuObserver::CompleteWithError <a href="#">CCProjectZeroMembers</a>
2299	Fixed	----	2022-May-20	Microsoft	Kernel	mjurczyk	Windows Kernel multiple memory problems when handling incorrectly formatted security descriptors in registry hives <a href="#">CCProjectZeroMembers</a>
2297	Fixed	----	2022-May-17	Microsoft	Kernel	mjurczyk	Windows Kernel invalid read/write due to unchecked Blink cell index in root security descriptor <a href="#">CCProjectZeroMembers</a>
2296	Fixed	----	2022-May-13	Google	Chrome	markbrand	Chrome: PaintImage deserialization OOB-read <a href="#">CCProjectZeroMembers</a>

# Blogs ?

project-zero project-zero ▾

New issue

All issues ▾

Q Search project-zero issues... ▾

☆ Starred by 2 users

Owner: [forshaw@google.com](#)

CC: [proje...@google.com](#)

Status: Fixed (Closed)

Components: ----

Modified: Sep 9, 2022

[Finder-forshaw](#)  
[Deadline-90](#)  
[Product-Windows](#)  
[Severity-low](#)  
[CCProjectZeroMembers](#)  
[Vendor-Microsoft](#)  
[Methodology-Manual](#)  
[Reported-2022-May-27](#)  
[MSRC-72169](#)  
[Fixed-2022-Aug-09](#)

**Issue 2304: Windows: Credential Guard Kerberos Change Password EoP**  
Reported by [forshaw@google.com](#) on Sat, May 28, 2022, 3:32 AM GMT+9 **Project Member**

Windows: Credential Guard Kerberos Change Password EoP  
Platform: Windows 10+  
Class: Elevation of Privilege  
Security Boundary: Virtual Secure Mode

Summary: Credential guard doesn't prevent using encrypted Kerberos keys to change a user's password leading to EoP.

Assumptions: The vulnerability described assumes that a user has authenticated to a machine but is currently not active. As in they're not typing in their password which could be captured by a malicious SSF

I also assume that you have SYSTEM and/or

PPL bypass (assuming such a thing exists) o

as you shouldn't be able attack Credential C

However, this research isn't going to verify t

LSASS process memory so it'd only be a m

Description: When a user authenticates to a

Expected Result:  
The creation of a password change ticket fails.

Observed Result:  
The password change ticket is created and the user's password can be changed.

















**This bug is subject to a 90-day disclosure deadline. If a fix for this issue is made available to users before the end of the 90-day deadline, this bug report will become public 30 days after the fix was made available. Otherwise, this bug report will become public at the deadline.**  
The scheduled deadline is 2022-08-25.

**poc\_kpass.zip**  
11.3 KB [Download](#)

# Github ?


As the developers here already know

- you can find out about the features that are currently under development through the commit log on GitHub.
- and sometimes, we can find vulnerability report from hacker through the commit log

Fix hasRareData() check in Element ...
 chirags27 authored and miwa committed 3 days ago
Fix error in entitlement array ...
 pvollan committed 3 days ago
Add PropertyWrapperFontWeight to bound the values for weight ...
 chirags27 authored and literum committed 3 days ago
SVG animations should respect Page::imageAnimationEnabled ...
 twilco committed 3 days ago
[Cocoa] Follow-up localized string addition for fullscreen mode ...
 jernoble committed 3 days ago
Add Web Extension code generator scripts, bindings glue, and some IDL... ...
 xeenon committed 3 days ago
Ingest managed domains for ResourceLoadStatistics and loosen restrict... ...
 pascoe committed 3 days ago
CG display lists should be replayed with asynchronous rendering if ap... ...
 hortont424 committed 3 days ago
[Cocoa] Enhance per-navigation "network connection integrity" setting ...
 sysrq authored and whsieh committed 3 days ago
Remove ./Source/WebCore/PAL/libavif/android_jni/gradle/wrapper/gradle... ...
 alancoon committed 3 days ago
[JSC] Simplify toThis operation ...
 Constellation committed 3 days ago
Remove WebGLLoadPolicy code ...
 grorg committed 3 days ago
[JSC][armv7] Use register numbers, not names, in disassembly ...
 jigriego authored and Constellation committed 3 days ago
nullptr crash in WebCore::IDBTransaction::dispatchEvent ...
 salinas-miguel authored and szewai committed 3 days ago
[JSC] uDFG should be able to watch JSGlobalObject WatchpointSets ...
 Constellation committed 3 days ago
[ITP Cleanup] Rename some resource load statistics feature flags to r... ...




# GitHub ?

 chromium / chromium Public

👁 Watch 515 🍴 Fork 5k ☆ Star 13.3k

[Code](#) [Pull requests 52](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

**[SharedHighlighting] check for UAF of render\_frame\_host** Browse files

Bug: 1329794  
Change-Id: Ia8dcba77226d00d86b72feb3faa3cd1143dded1e  
Reviewed-on: <https://chromium-review.googlesource.com/c/chromium/src/+3689843>  
Reviewed-by: Avi Drissman <avi@chromium.org>  
Reviewed-by: Cheick Cisse <cheickcisse@google.com>  
Auto-Submit: Jeffrey Cohen <jeffreycohen@chromium.org>  
Commit-Queue: Avi Drissman <avi@chromium.org>  
Cr-Commit-Position: refs/heads/main@{#1011529}

main  
109.0.5393.1 ... 104.0.5108.0

Jeffrey Cohen authored and Chromium LUCI CQ committed on 8 Jun 1 parent 0130770 commit ff961958932aab5259ca9e0642d5a38910849b99

Showing 2 changed files with 16 additions and 8 deletions. Split Unified

- chrome/browser/renderer\_conte...
  - link\_to\_text\_menu\_observer....
  - link\_to\_text\_menu\_observer.h

16 chrome/browser/renderer\_context\_menu/link\_to\_text\_menu\_observer.cc

95	RenderViewContextMenuProxy* proxy,	95	RenderViewContextMenuProxy* proxy,
96	content::RenderFrameHost* render_frame_host,	96	content::RenderFrameHost* render_frame_host,
97	CompletionCallback callback)	97	CompletionCallback callback)
98	- : proxy_(proxy),	98	+ : content::DocumentUserData<LinkToTextMenuObserver> render_frame_host,

HAECHI

# Some tips for analyst

06

좋은 분석가가 되기 위한 몇 가지 이야기

# Difficulty

Which part of code should i analyze ?

- **General source code is too huge**
  - Like chromium, arbitrum ..
- **We need to choose component to be analyzed**
- **Sufficient information needs to be gathered before analysis**
  - Like Whitepaper, class references ..
- **How to reduce repetitive tasks**

OffchainLabs / arbitrum Public

<> Code Issues 49 Pull requests 71 Actions Projects

master arbitrum / packages /

PlasmaPower Merge branch 'master' into arb\_legacy\_sender

..

arb-avm-cpp	Fix tracing for block 1
arb-bridge-eth	chore: lint
arb-bridge-peripherals	Update packages/arb-brid
arb-evm	export message batches
arb-node-core	Fix retryable test
arb-os @ 234cf67	Update arbos pin
arb-rpc-node	Merge branch 'master' into
arb-ts	Remove arb-ts references i
arb-upgrades	Merge branch 'master' into
arb-util	Disable classic sequencer f
go-ethereum @ 8dd7cd4	update geth
tools	Fix nitro migrator linting
.dockerignore	.dockerignore reverse sym
MACHINEHASH	Fix retryable test
arb-node.Dockerfile	Fix docker build



# Whitepaper ?

Whitepaper helps you get rid of code you don't need to analyze

Whitepaper helps you to draw an image of architecture in your brain

Whitepaper helps to understand the components and concepts that the software implements

## Arbitrum Nitro: A Second-Generation Optimistic Rollup

*Lee Bousfield, Rachel Bousfield, Chris Buckland, Ben Burgess, Joshua Colvin, Edward W. Felten, Steven Goldfeder, Daniel Goldman, Braden Huddleston, Harry Kalodner, Frederico Arnaud Lacs, Harry Ng, Aman Sanghi, Tristan Wilson, Valeria Yermakova, and Tsahi Zidenberg*

*Offchain Labs, Inc.*

### Abstract

We present Arbitrum Nitro, a second-generation Layer 2 blockchain protocol. Nitro provides higher throughput, faster finality, and more efficient dispute resolution than previous rollups. Nitro achieves these properties through several design principles: separating sequencing of transactions from deterministic execution; combining existing Ethereum emulation software with extensions to enable cross-chain functionalities; compiling separately for execution versus proving, so that execution is fast and proving is structured and machine-independent; and settling transaction results to the underlying Layer 1 chain using an optimistic rollup protocol based on interactive fraud proofs.

one participant in the Nitro protocol is behaving honestly. The protocol is termed “optimistic” because execution is more *efficient* when parties behave according to their incentives.

A variant of Nitro, called AnyTrust, provides lower cost in exchange for an additional trust assumption. The main body of this paper describes regular Nitro, and the differences in AnyTrust are described in Section 7.

Nitro has been deployed on the Arbitrum One chain, with Ethereum as the underlying Layer 1, since August 31, 2022. Nitro’s source code is available at <https://github.com/offchainlabs/nitro> and its submodules.

### 1 Introduction

In previous work, we described Arbitrum [9], a system and protocol for improving the performance and scalability of smart contracts. This paper describes Arbitrum Nitro, a significantly improved design that offers advantages over the original, including greater efficiency, reduced latency, stronger liveness guarantees, and better incentive compatibility.

#### 1.2 Design Approach

Nitro’s design has four distinctive features, which we will use to organize the presentation.

- *Sequencing followed by deterministic execution:* Nitro handles submitted transactions in two stages. First, it puts transactions into the sequence in which they will be processed, and commits to that sequence. Second, it applies a deterministic state transition function to each transaction, in sequence.
- *Geth at the core:* The core execution and state maintenance functions in Nitro are handled by code from the open source go-ethereum (“geth”) package, which is the most popular Ethereum execution layer node software. By compiling in that geth code as a library, Nitro ensures that its execution and state are highly compatible with Ethereum’s.
- *Separate execution from proving:* Nitro compiles the code of its state transition function for two targets. The code is compiled for native execution when used in ordinary operation in a Nitro node,

#### 1.1 Properties of Arbitrum Nitro

Nitro supports execution of smart contracts. The system is implemented as a “Layer 2” on top of Ethereum [14], although in principle it could be implemented on any blockchain system that supports at least basic smart contract functionality. Nitro provides an Ethereum-compatible chain: Nitro runs smart contract applications deployed in Ethereum Virtual Machine (EVM) code, and Nitro nodes support the same API as common Ethereum nodes.

The Nitro protocol guarantees both safety and

# Whitepaper ?

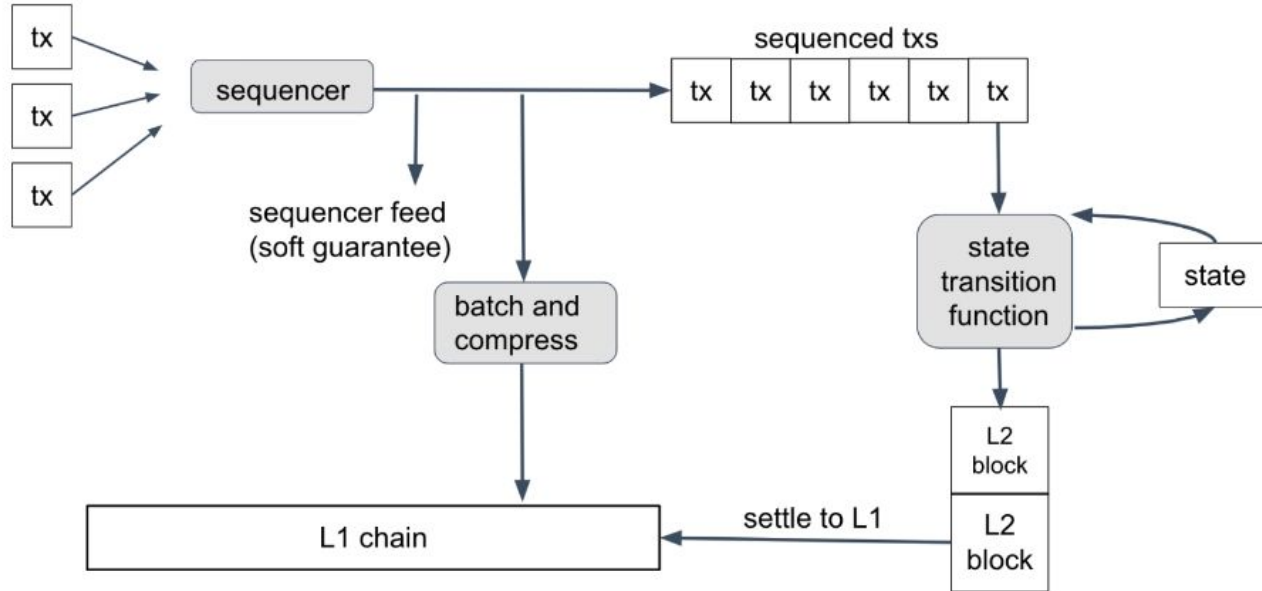


Figure 1: Processing of transactions in Nitro. The sequencer establishes an ordering on transactions, and publishes the order as a real-time feed and as compressed data batches on the L1 chain. Sequenced transactions are processed one at a time by a deterministic state transition function, which updates the chain state and produces L2 blocks. These blocks are later settled to the L1 chain.

# Whitepaper ?

L1 chain will cause an included message to disappear or change. The current Sequencer implementation includes Delayed Inbox messages after ten minutes. If the Sequencer fails to include a Delayed Inbox message within a fixed interval<sup>4</sup>, any party can call the Inbox to force inclusion of the message, which occurs by forcing a Sequencer batch including the message(s) into the Inbox. The ability to submit a message to the Delayed Inbox and force its inclusion without relying on the Sequencer supports Nitro's guarantee of liveness.

## 3.2.4 Retryable Tickets

Layer 1 contracts can submit transactions to a Nitro chain, but those transactions necessarily run asynchronously on the Nitro chain, so the submitting Layer 1 transaction cannot see whether they succeed. This poses problems for applications such as token bridging which

---

<sup>4</sup>Setting this parameter reflects a tradeoff between the desire for prompt inclusion, and the desire to avoid unexpected behavior if the Sequencer experiences downtime. Currently it is set to 24 hours, but we expect the value to be reduced as the perceived risk of Sequencer downtime diminishes.

submitters know exactly what the fee will be.

## 3.2.5 Token Bridge

Nitro's cross-chain messaging affordances can be used to create a Token Bridge, an application that allows for the effective transfer of assets between the Ethereum and Nitro chains. The Offchain Labs team has implemented and released a Token Bridge informally referred to as "canonical", though the Nitro core protocol grants it no special recognition or affordances; it is effectively an application like any other. (Note that, similarly, Nitro has no natively recognized notion of tokens nor of any particular token standard, much like Ethereum.)

At its core, the Token Bridge offers the ability to deposit (transfer from Ethereum to Nitro) and withdraw (transfer from Nitro to Ethereum) fungible tokens. To deposit  $n$  tokens, a transaction is sent to Ethereum which carries out two operations: it sends  $n$  tokens to an L1 contract (known as a Token Gateway), and creates a retryable transaction (Section 3.2.4) that mints  $n$  tokens of an L2-counterpart contract. The two token contracts are counterparts, due to the guarantee that a holder of

# Whitepaper ?

```
func (rs *RetryableState) CreateRetryable(
 id common.Hash, // we assume that the id is unique and hasn't been used before
 timeout uint64,
 from common.Address,
 to *common.Address,
 callvalue *big.Int,
 beneficiary common.Address,
 calldata []byte,
) (*Retryable, error) {
 sto := rs.retryables.OpenSubStorage(id.Bytes())
 ret := &Retryable{
 id,
 sto,
 sto.OpenStorageBackedUint64(numTriesOffset),
 sto.OpenStorageBackedAddress(fromOffset),
 sto.OpenStorageBackedAddressOrNil(toOffset),
 sto.OpenStorageBackedBigUint(callvalueOffset),
 sto.OpenStorageBackedAddress(beneficiaryOffset),
 sto.OpenStorageBackedBytes(calldataKey),
 sto.OpenStorageBackedUint64(timeoutOffset),
 sto.OpenStorageBackedUint64(timeoutWindowsLeftOffset),
 }
 _ = ret.numTries.Set(0)
 _ = ret.from.Set(from)
```

# Whitepaper ?

```
_ = ret.numTries.Set(0)
_ = ret.from.Set(from)
_ = ret.to.Set(to)
_ = ret.callvalue.Set(callvalue)
_ = ret.beneficiary.Set(beneficiary)
_ = ret.calldata.Set(calldata)
_ = ret.timeout.Set(timeout)
_ = ret.timeoutWindowsLeft.Set(0)

// insert the new retryable into the queue so it can be reaped later
return ret, rs.TimeoutQueue.Put(id)
}
```

# Whitepaper ?

```
func initializeRetryables(statedb *state.StateDB, rs *retryables.RetryableState, initData statetransfer.RetryableDataReader, currentTimestamp uint64) error {
 var retryablesList []*statetransfer.InitializationDataForRetryable
 for initData.More() {
 r, err := initData.GetNext()
 if err != nil {
 return err
 }
 if r.Timeout <= currentTimestamp {
 statedb.AddBalance(r.Beneficiary, r.Callvalue)
 continue
 }
 retryablesList = append(retryablesList, r)
 }
 sort.Slice(retryablesList, func(i, j int) bool {
 a := retryablesList[i]
 b := retryablesList[j]
 if a.Timeout == b.Timeout {
 return arbmeth.BigLessThan(a.Id.Big(), b.Id.Big())
 }
 return a.Timeout < b.Timeout
 })
 for _, r := range retryablesList {
 var to *common.Address
 if r.To != (common.Address{}) {
 to = &r.To
 }
 statedb.AddBalance(retryables.RetryableEscrowAddress(r.Id), r.Callvalue)
 _, err := rs.CreateRetryable(r.Id, r.Timeout, r.From, to, r.Callvalue, r.Beneficiary, r.Calldata)
 if err != nil {
 return err
 }
 }
 return initData.Close()
}
```

# Whitepaper ?

```
52 func InitializeArbosInDatabase(db ethdb.Database, initData statetransfer.InitDataReader, chainConfig *params.ChainConfig)
53 stateDatabase := state.NewDatabase(db)
54 statedb, err := state.New(common.Hash{}, stateDatabase, nil)
55 if err != nil {
56 log.Crit("failed to init empty statedb", "error", err)
57 }
58
59 commit := func() (common.Hash, error) {
60 root, err := statedb.Commit(true)
61 if err != nil {
62 return common.Hash{}, err
63 }
64 err = stateDatabase.TrieDB().Commit(root, true, nil)
65 if err != nil {
66 return common.Hash{}, err
67 }
68 statedb, err = state.New(root, stateDatabase, nil)
69 if err != nil {
70 return common.Hash{}, err
71 }
72 return root, nil
73 }
74
75 burner := burn.NewSystemBurner(nil, false)
76 arbosState, err := InitializeArbosState(statedb, burner, chainConfig)
77 if err != nil {
78 log.Crit("failed to open the ArbOS state", "error", err)
79 }
```

## Class/Structure ref

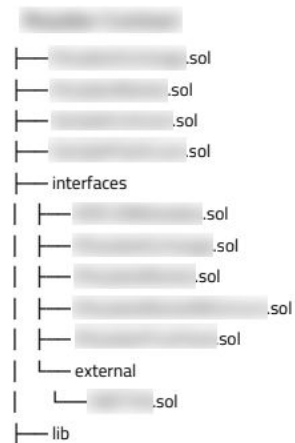
1. **How to control the value of a particular memory or variable**
2. **Which class dependencies should be considered**
3. **What class is class A related to**





## Class/Structure ref

- 1. Reviewing class references reduces the scope of the code you have to look at**
- 2. You can check which variables and class instances are derived from which functions or classes**
- 3. It makes you to check in advance the factors to be considered when writing the exploit code**



# Make scenario

1. It is efficient to organize the types of vulnerabilities that can occur in a specific situation in perform analysis.
2. For example, in a bridge application of a blockchain, duplicated withdrawal problem or an address calculation problem may occur.
3. Whether it's useful to come up with a scenario can vary from person to person.

# Make scenario

1. Many user can stake tokens into staking smart contract.
2. If end of this staking period, users receive regular rewards against the funds they have deposited
3. User's reward request will be appended into pending request queue
4. Admin can accept pending requests from queue when he want

# Make scenario

## 🚫 5. Denial of Service

including **gas limit reached, unexpected throw, unexpected kill, access control breached**

---

I accidentally killed it.

— devops199 on the Parity multi-sig wallet

---

Denial of service is deadly in the world of Ethereum: while other types of applications can eventually recover, smart contracts can be taken offline forever by just one of these attacks. Many ways lead to denials of service, including maliciously behaving when being the recipient of a transaction, artificially increasing the gas necessary to compute a function, abusing access controls to access private components of smart contracts, taking advantage of mixups and negligence, etc. This class of attack includes many different variants and will probably see a lot of development in the years to come.

**Loss:** estimated at 514,874 ETH (~300M USD at the time)

**Real World Impact:**

# Make scenario

```
submissionFee := retryables.RetryableSubmissionFee(len(tx.RetryData), tx.L1BaseFee)
if arbmeth.BigLessThan(tx.MaxSubmissionFee, submissionFee) {
 // should be impossible as this is checked at L1
 err := fmt.Errorf(
 "max submission fee %v is less than the actual submission fee %v",
 tx.MaxSubmissionFee, submissionFee,
)
 return true, 0, err, nil
}

// collect the submission fee
if err := transfer(&tx.From, &networkFeeAccount, submissionFee); err != nil {
 // should be impossible as we just checked that they have enough balance for the max submission fee,
 // and we also checked that the max submission fee is at least the actual submission fee
 glog.Error("failed to transfer submissionFee", "err", err)
 return true, 0, err, nil
}
withheldSubmissionFee := takeFunds(availableRefund, submissionFee)

// refund excess submission fee
submissionFeeRefund := takeFunds(availableRefund, arbmeth.BigSub(tx.MaxSubmissionFee, submissionFee))
if err := transfer(&tx.From, &tx.FeeRefundAddr, submissionFeeRefund); err != nil {
 // should never happen as from's balance should be at least availableRefund at this point
 glog.Error("failed to transfer submissionFeeRefund", "err", err)
}
```

# Make scenario

1. How to submit retryable transaction with zero submission fee ?
2. Can bridge balance was bypassed with underflow vulnerability ?
3. Is there any problem in calculating the escrow address ?
4. etc...

# Make scenario

There are more components to think about some scenarios

- ArbOS
- WAVM
- Retryable Ticket
- Standard Library Problem ..
- Etc..

# Choose best tools

**Classic and simple tools are good, but there are many useful tools.**

**When choosing a tool, choose one that eliminates as much hassle as possible**

**Tools related to automation are good, but analysis tools with nice interface should also be selected in consideration of the above.**

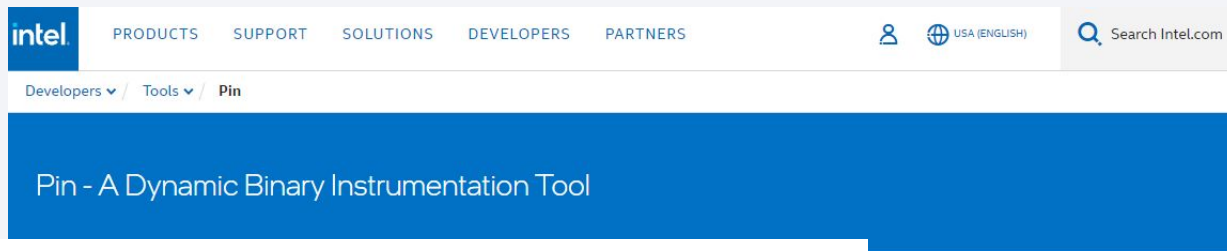
**So, which tools are examples of good tools ?**



# DBI ?

1. For example, PIN and Frida ..
2. DBI provides functions such as tracking the flow of the program you want to analyze, tracking the data movement flow in memory, and hooking.
3. You can do it quickly through DBI programming, without having to manually trace function calls or track movements of input values.
4. This effectively reduces your time consumption.

# DBI ?



## FRIDA

[OVERVIEW](#)[DOCS](#)[NEWS](#)[CODE](#)[CONTACT](#)

# Dynamic instrumentation toolkit for developers, reverse- engineers, and security researchers.

A-32, x86-64 and MIC instruction-set architecture tools built with Pin are Intel® VTune™ Amplifier, Intel® VTune™ Amplifier (Intel® SDE). The tools created using Pin can instrument applications on Linux\*, Windows\* and macOS. The tools are formed at run time on the compiled binary files. The tools instrumenting programs that dynamically generate

instruction-set idiosyncrasies and allows context i as parameters. Pin automatically saves and resto plication continues to work. Limited access to sy

Scriptable

Portable

Free

Battle-tested

Inject your own scripts into

Works on Windows, macOS

Frida is and will always be

We are proud that

HAECHI

# Debugging Scripts

1. For example, immScript, pykd ..
2. Debugging scripts make using the debugger efficient and simplifying hard debugging tasks.
3. In my case, I mainly use pykd when using windbg for operating system and kernel analysis.

# Debugging Scripts



pykd 

This project can help to automate debugging and crash dump analysis using Python. It allows one to take the best from both worlds: the expressiveness and convenience of Python with the power of WinDbg!



Star

2

HTTPS ▼

<https://github.com/0x00b0/pykd>

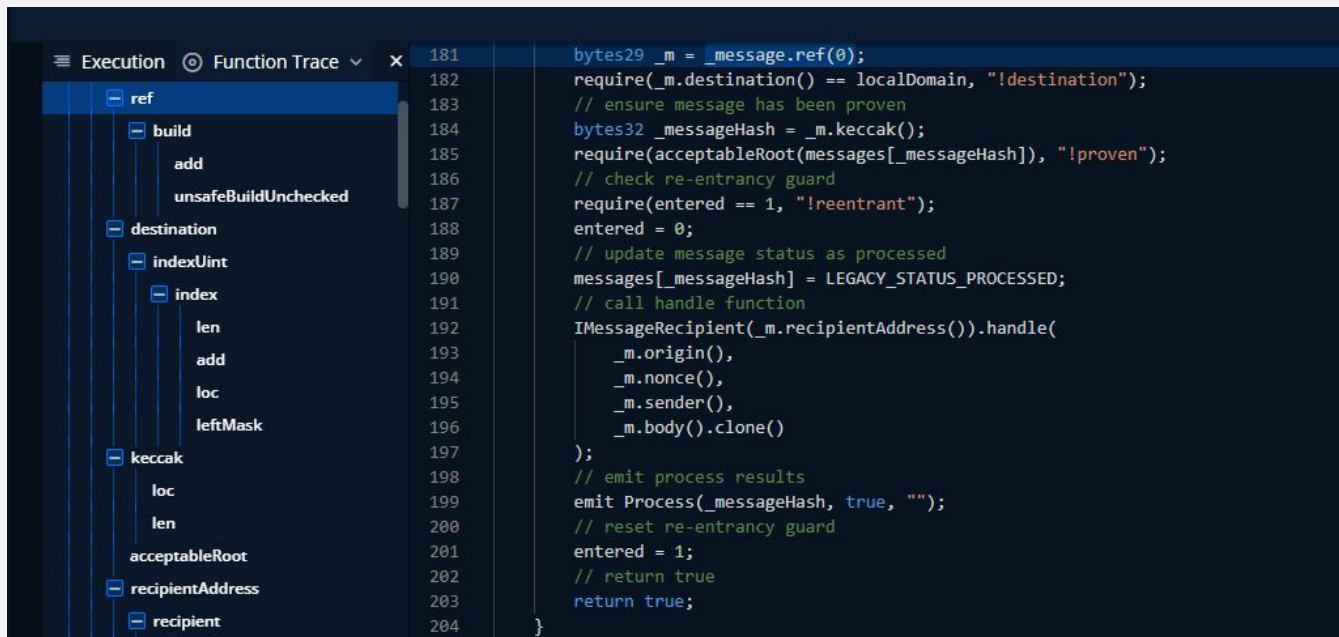


Files (7.7 MB) Commits (1,148) Branches (7) Tags (0) Readme MIT License

# From blockchain ?

1. For example, slither, tenderly ..
2. I'm working on analytics for malicious transactions on blockchain network.
3. For this reason, a tool called tenderly is very helpful to me.
4. Tenderly graphically shows token transferring and transaction execution very efficiently.

# From blockchain ?



The screenshot displays a debugger interface with a function trace on the left and source code on the right.


**Function Trace (Left Panel):**

- ref** (selected)
  - build**
    - add
    - unsafeBuildUnchecked
  - destination**
  - indexUint**
    - index**
      - len
      - add
      - loc
      - leftMask
  - keccak**
    - loc
    - len
    - acceptableRoot
  - recipientAddress**
  - recipient**

**Source Code (Right Panel):**


```
181 bytes29 _m = _message.ref(0);
182 require(_m.destination() == localDomain, "!destination");
183 // ensure message has been proven
184 bytes32 _messageHash = _m.keccak();
185 require(acceptableRoot(messages[_messageHash]), "!proven");
186 // check re-entrancy guard
187 require(entered == 1, "!reentrant");
188 entered = 0;
189 // update message status as processed
190 messages[_messageHash] = LEGACY_STATUS_PROCESSED;
191 // call handle function
192 IMessageRecipient(_m.recipientAddress()).handle(
193 _m.origin(),
194 _m.nonce(),
195 _m.sender(),
196 _m.body().clone()
197);
198 // emit process results
199 emit Process(_messageHash, true, "");
200 // reset re-entrancy guard
201 entered = 1;
202 // return true
203 return true;
204 }
```


# From blockchain ?

 **Transaction**  
0xb1fe26cc88...5d63ae28

Re-SimulateShareView in Explorer

**Transaction Hash:** 0xb1fe26cc8892f58eb468f5208baaf38bac422b5752cca0b9c8a871855d63ae28

**Network:**  Mainnet

**Status:**  Success **Block:** 15259101 (614,584 blocks ago) **Transaction Index:** 3


**Timestamp:** 3 months ago (02/08/2022 06:32:31) **Nonce:** 25 **Value:** 0 Ether **Gas Used:** 170,294





**Gas Price:** 16,885.619 Gwei **Gas Limit:** 207,712 **Transaction Fee:** 2.876 Ether




**Raw Input:** 0x928bc4b20000000000...0000000000000000


**Caller Address:** 0xb5c55f76f90cc528b2609109ca14d8d84593590e

**Contract Address:** UpgradeBeaconProxy (0x5d94309e5a0090b165fa4181519701637b6daeba)

 **Tokens Transferred**

 from  UpgradeBeaconProxy to  0xb5c55f76f9...93590e 100  Wrapped BTC (WBTC)


Search in execution trace  Function Trace  Expand all  Collapse all

 fallback in UpgradeBeaconProxy:65

# Others ...

## Scylla, IDAPython ..

The Rinkeby testnet explorer has been discontinued and set to read-only on October 5th, 2022. Please migrate your contracts and deploy new ones on Goerli or Sepolia. Read more [here](#).

 Etherscan

Rinkeby Testnet Network

All Filters

Search by Address / Txn Hash / Block / Token / Ens

HomeBlockchainTokensMiscRinkeby

Contract 0x187BF594ed3dbe0A89a326b83Fd18a999B0522e0

Overview

Balance: 0 Ether

More Info

My Name Tag: Not Available

Contract Creator: 0xb33c4486d4fff866d42f... at bxn  
0x1de1abb5c59b76a695...

TransactionsInternal TxnsErc20 Token TxnsContract Reinit

Latest 3 from a total of 3 transactions

Txn Hash	Method	Block	Age	From	To	Value	Txn Fee
<a href="#">0x3e8be1aec4f53e06cc8...</a>	<a href="#">0x03fb03e3</a>	11441473	38 days 18 hrs ago	<a href="#">0xb33c4486d4fff866d42f...</a>	<a href="#">IN</a> <a href="#">0x187bf594ed3dbe0a89...</a>	0 Ether	0.00009222
<a href="#">0xc31b2923b070ccb009...</a>	<a href="#">Close</a>	11441470	38 days 18 hrs ago	<a href="#">0xb33c4486d4fff866d42f...</a>	<a href="#">IN</a> <a href="#">0x187bf594ed3dbe0a89...</a>	0 Ether	0.00004251
<a href="#">0x79e5659823e9d3a5fe...</a>	<a href="#">Deploy B</a>	11441451	38 days 18 hrs ago	<a href="#">0xb33c4486d4fff866d42f...</a>	<a href="#">IN</a> <a href="#">0x187bf594ed3dbe0a89...</a>	0 Ether	0.00012202



# For example ..

From zer0pts ctf 2021 reverse engineering challenge

- Super Secret Login
- Which software developed with Autoit3
- It consists of a flow that parses the code on the emulator and executes the bytecodes.



# For Example ..

```
.text:00419FC6 mov dword_4C0CAC, 1
.text:00419FD0 mov dword_4C0CB0, 2
.text:00419FDA mov byte_4C0CB4, 0
.text:00419FE1 mov dword_4C0CB8, offset aDllcall ; "DLLCALL"
.text:00419FEB mov dword_4C0CC4, offset sub_47E237
.text:00419FF5 mov dword_4C0CC8, 0
.text:00419FFF mov dword_4C0CCC, 0
.text:0041A009 mov dword_4C0CD0, 3
.text:0041A013 mov dword_4C0CD4, 43h
.text:0041A01D mov byte_4C0CD8, 0
.text:0041A024 mov dword_4C0CDC, offset aDllcalladdress ; "DLLCALLADDRESS"
.text:0041A02E mov dword_4C0CE8, offset sub_47E24B
.text:0041A038 mov dword_4C0CEC, 0
.text:0041A042 mov dword_4C0CF0, 0
.text:0041A04C mov dword_4C0CF4, 2
.text:0041A056 mov dword_4C0CF8, 42h
.text:0041A060 mov byte_4C0CFC, 0
.text:0041A067 mov dword_4C0D00, offset aDllcallbackfre ; "DLLCALLBACKFREE"
```

# For Example ..

```
StringLen : 0x482331
GuiCtrlSetState : 0x47064C
GuiCtrlCreateCheckBox : 0x481917
DllStructCreate : 0x47e7e5
GuiCtrlCreateCheckBox: 0x472D2D
DllStructSetData : 0x47eb06
StringLen : 0x482331
DllStructCreate : 0x47e7e5
DllStructSetData
StringLen
DllStructCreate
DllStructGetPtr : 0x47e99b
DllStructGetPtr
DllStructGetPtr
StringLen
DllCall : 0x47e237
 -> CallWindowProcA -> Execute Generated Function (Check the stack)
DllStructGetData : 0x47e893
GuiCtrlSetState
GuiCtrlCreateCheckBox : 0x4819d5
String : 0x473a3b
MsgBox : 0x47564f
GuiGetMsg : 0x47094f
```

# For Example ..

```
}
while (v6 != 256);
v9 = 0;
v10 = 0;
v11 = 0;
do
{
 ++v9;
 v12 = (unsigned __int8)(v10 + 1);
 v13 = v16[v12];
 v14 = (unsigned __int8)(v13 + v11);
 LOBYTE(v7) = v16[v14];
 v16[v14] = v13;
 v16[v12] = v7;
 v15 = (unsigned __int8)(v7 + v13);
 LOBYTE(v7) = *(_BYTE *)(a3 + v9 - 1);
 v7 ^= (unsigned __int8)v16[v15];
 sub_94();
}
while (v9 != a4);
```

# Promotion

07

회사 홍보 (대표님한테 부탁받음)

# Promotion

We are belong in blockchain security company, and blockchain researchers.

- <https://career.haechi.io/>
- <https://career.haechi.io/audit/security-researcher>
- 문의 : [\*\*people@haechi.io\*\*](mailto:people@haechi.io)
- 오피스 이쁘니까 구경하러 오세요~

# Q&A

**Jeremy Lim**

Email : [jeremy@haechi.io](mailto:jeremy@haechi.io)

Company : Haechi Labs