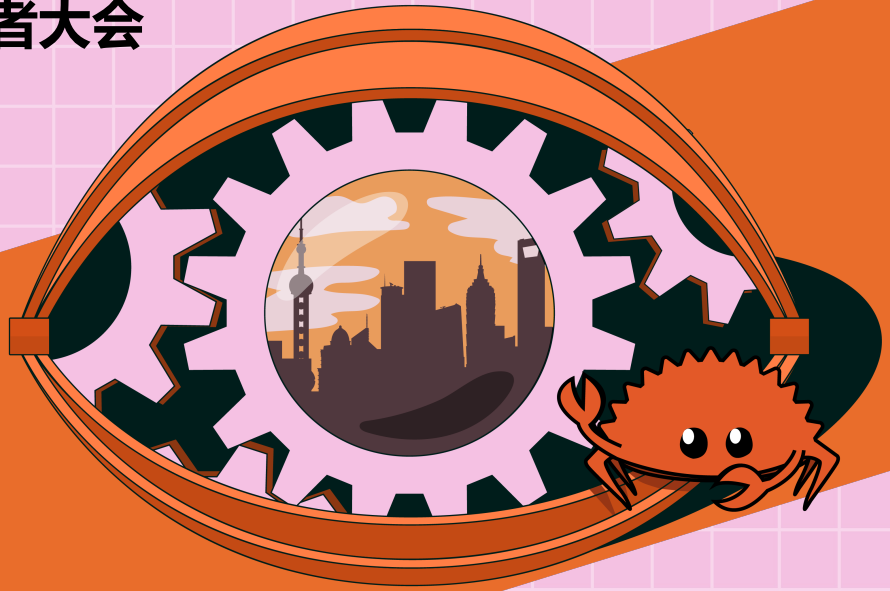
RUST CHINA CONF 2023

第三届中国Rust开发者大会



6.17-6.18 @Shanghai

Rustle: the first static analyzer for smart contracts in Rust

Matthew Jiang

Director of Security Team @ BlockSec

Outline

1. Motivation
2. Background
3. Design
4. Capability
5. Usability
6. Conclusion

Motivation

Why do we develop Rustle?

Motivation

Emerging chains with Rust as smart contracts language

Limitations of manual audit

- Time-consuming
- Expensive
- Skilled auditors
- Error-prone

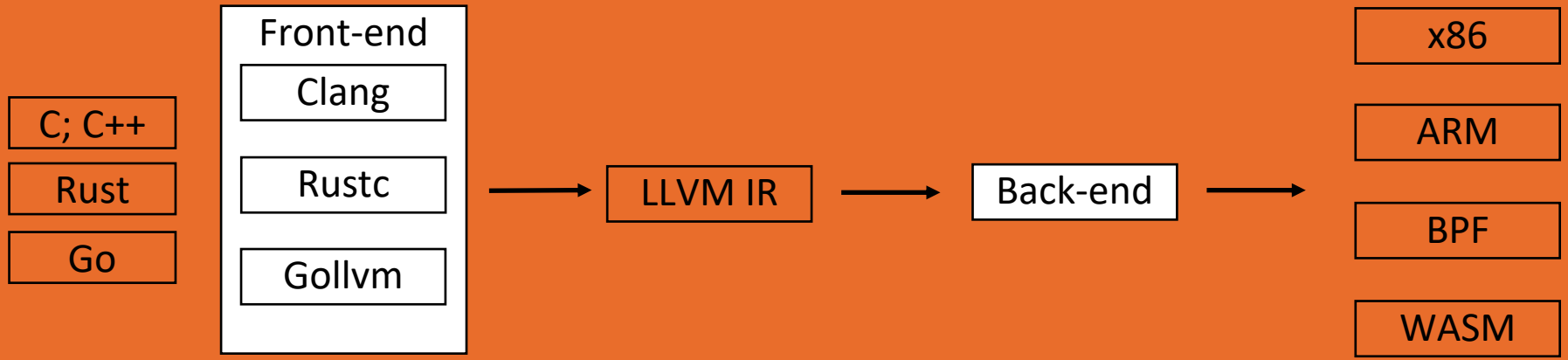
Current tools:
Don’t support contracts in Rust

BlockSec
Rustle

Background

Design

LLVM IR

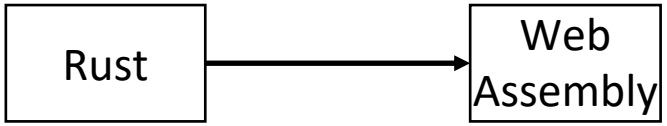


LLVM Pass: rich APIs to analysis the LLVM IR

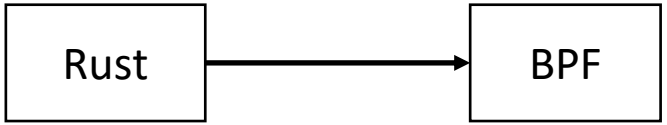
Background

Blockchains with smart contracts in Rust

Near



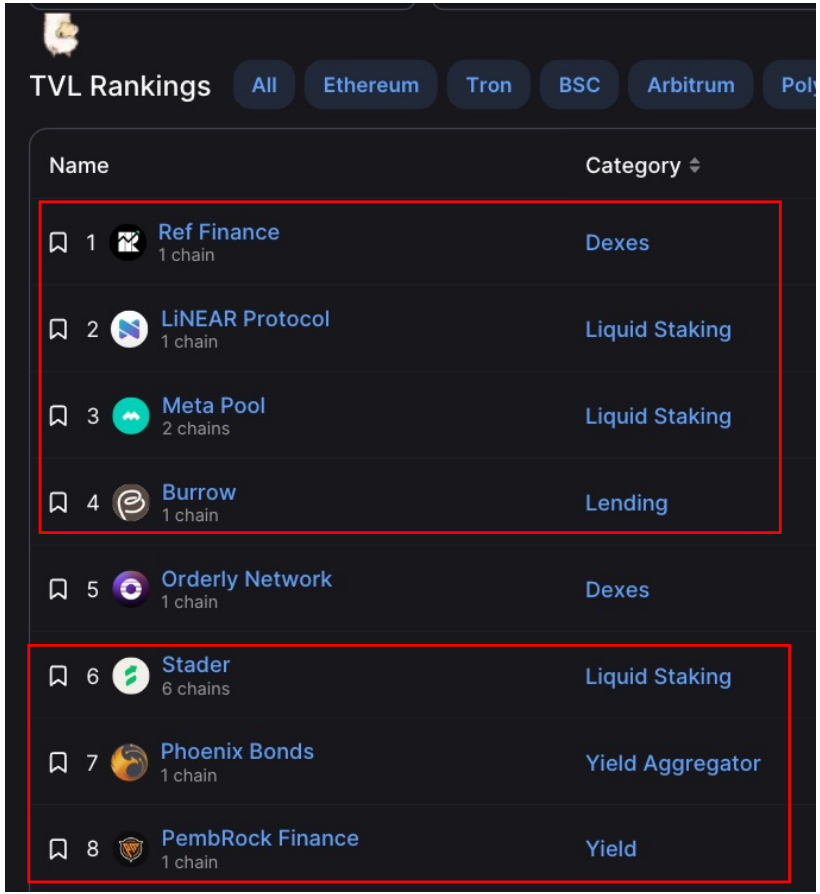
Solana



Though they are compiled into bytecode in different formats, we can analyze the logic on the LLVM IR

Background

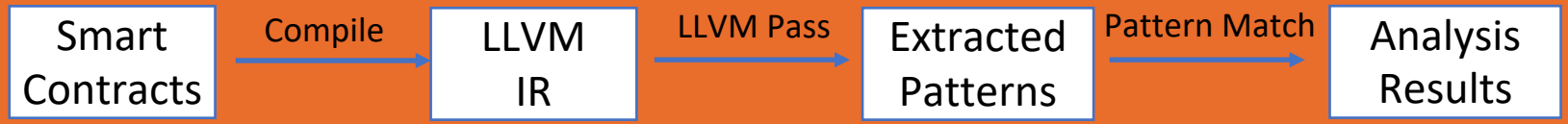
Rich experience in auditing smart contracts in Rust



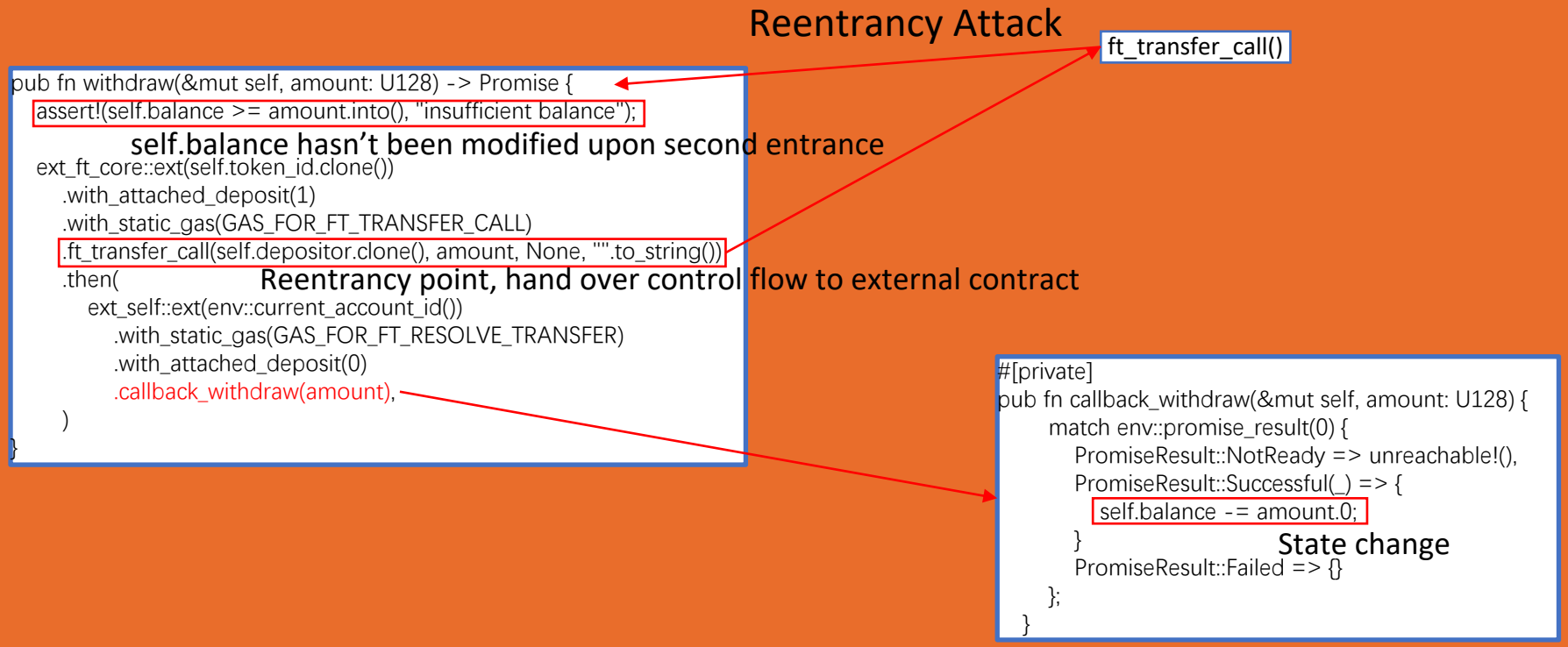
Design

Design

Workflow

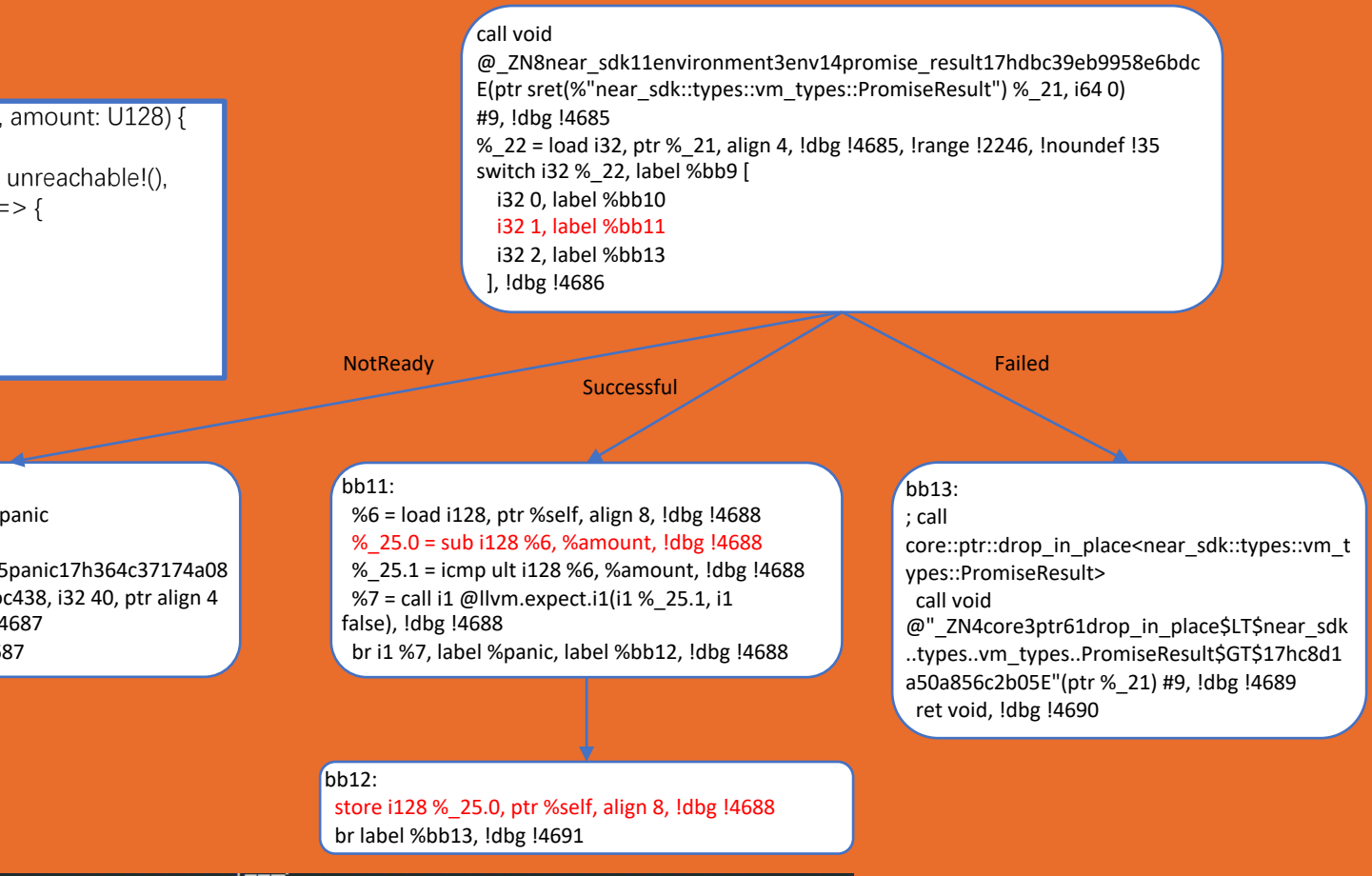


Example


Reentrancy Attack
ft_transfer_call()
pub fn withdraw(&mut self, amount: U128) -> Promise {
assert!(self.balance >= amount.into(), "insufficient balance");
self.balance hasn't been modified upon second entrance
ext_ft_core::ext(self.token_id.clone())
.with_attached_deposit(1)
.with_static_gas(GAS_FOR_FT_TRANSFER_CALL)
.ft_transfer_call(self.depositor.clone(), amount, None, "".to_string())
.then(
Reentrancy point, hand over control flow to external contract
ext_self::ext(env::current_account_id())
.with_static_gas(GAS_FOR_FT_RESOLVE_TRANSFER)
.with_attached_deposit(0)
.callback_withdraw(amount),
)
}
#[private]
pub fn callback_withdraw(&mut self, amount: U128) {
match env::promise_result(0) {
PromiseResult::NotReady => unreachable!(),
PromiseResult::Successful(_) => {
self.balance -= amount.0;
}
State change
PromiseResult::Failed => {}
};
}

Design

```
pub fn callback_withdraw(&mut self, amount: U128) {
    match env::promise_result(0) {
        PromiseResult::NotReady => unreachable!(),
        PromiseResult::Successful(_) => {
            self.balance -= amount.0;
        }
        PromiseResult::Failed => {}
    };
}
```




```
[!] Changing state upon PromiseResult::Successful at src/lib.rs:71:17 may lead to reentrancy.
```

Design

Advantages

- Support different chains with smart contracts in Rust
- Support to add new detectors for extension
- Support different platforms/OSs

Capability

- Being able to locate 30 different types of vulnerabilities
- Being able to find real world vulnerabilities

Capability

Rich Detectors

- Each detector can locate one specific type of security issues
- Rustle currently supports 30 different detectors
- For the complete list, refer to https://github.com/blocksecteam/rustle#detectors

Detectors

All vulnerabilities **Rustle** can find.

Detector ID	Description	Severity
`unhandled-promise`	find `Promises` that are not handled	High
`non-private-callback`	missing macro `#[private]` for callback functions	High
`reentrancy`	find functions that are vulnerable to reentrancy attack	High
`unsafe-math`	lack of overflow check for arithmetic operation	High
`self-transfer`	missing check of `sender != receiver`	High
`incorrect-json-type`	incorrect type used in parameters or return values	High
`unsaved-changes`	changes to collections are not saved	High
`nft-approval-check`	find `nft_transfer` without check of `approval id`	High
`nft-owner-check`	find approve or revoke functions without owner check	High

Capability

Real case

SPL-Lending
Incorrect rounding

Assume **collateral_amount** is 3, **self.0** is 2,
then **liquidity_amout** will be round(3 / 2) = 2
the **returned collateral_amount** will be round(2 * 2) = 4 > 3,
you can get more than you collateralized

```
555     pub fn collateral_to_liquidity(&self, collateral_amount: u64) -> Result<u64, ProgramError> {
556         Decimal::from(collateral_amount)
557             .try_div(self.0)?
558             .try_round_u64()
559     }
```

rounding can be manipulated

```
570     pub fn liquidity_to_collateral(&self, liquidity_amount: u64) -> Result<u64, ProgramError> {
571         self.0.try_mul(liquidity_amount)?.try_round_u64()
572     }
```

Rustle's Result:

```
[!] Find round at token-lending/program/src/state/reserve.rs:556:9
[!] Find round at token-lending/program/src/state/reserve.rs:571:9
[!] Find round at token-lending/program/src/state/reserve.rs:681:30
[!] Find round at token-lending/program/src/state/reserve.rs:688:17
```

Capability

Real case

NearX - $830K
Self-transfer

Rustle's Result:

```
[*] Find ft_transfer _ZN6near_x14fungible_token11ne
[!] Lack self-transfer check for ft_transfer
[*] Find ft_transfer_call _ZN6near_x14fungible_toke
[!] Lack self-transfer check for ft_transfer_call
```

(both ft_transfer and ft_transfer_call invoke internal_neax_transfer)

```rust
impl NearxPool {
    pub fn internal_nearx_transfer(
        &mut self,
        sender_id: &AccountId,
        receiver_id: &AccountId,
        amount: u128,
    ) {
        assert!(amount > 0, "The amount should be a positive number");
        let mut sender_acc = self.internal_get_account(sender_id);
        let mut receiver_acc = self.internal_get_account(receiver_id);
        assert!(
            amount <= sender_acc.stake_shares,
            "{} does not have enough NearX balance {}",
            sender_id,
            sender_acc.stake_shares
        );
                    sender_acc and receiver_acc can be the same
        sender_acc.stake_shares -= amount;
        receiver_acc.stake_shares += amount;

        self.internal_update_account(sender_id, &sender_acc);
        self.internal_update_account(receiver_id, &receiver_acc);
    }
```

Capability

Real case

Wormhole - $321M
Lack of owner-check

```
92          let current_instruction = solana_program::sysvar::instructions::load_current_index(
93              &accs.instruction_acc.try_borrow_mut_data()?,
94          );
95          if current_instruction == 0 {
96              return Err(InstructionAtWrongIndex.into());
97          }
98
99          // The previous ix must be a secp verification instruction
100         let secp_ix_index = (current_instruction - 1) as u8;
101         let secp_ix = solana_program::sysvar::instructions::load_instruction_at(
102             secp_ix_index as usize,
103             &accs.instruction_acc.try_borrow_mut_data()?,
104         )          secp_ix can be controlled with a fake accs.instruction_acc
105         .map_err(|_| ProgramError::InvalidAccountData)?;
```

Rustle's Result:

```
[!] Require owner check at program/src/api/verify_signature.rs:92:31
[!] Require owner check at program/src/api/verify_signature.rs:101:19
```

Usability

With complex technology stack, Rustle is easy to use

Usability

Easy to use

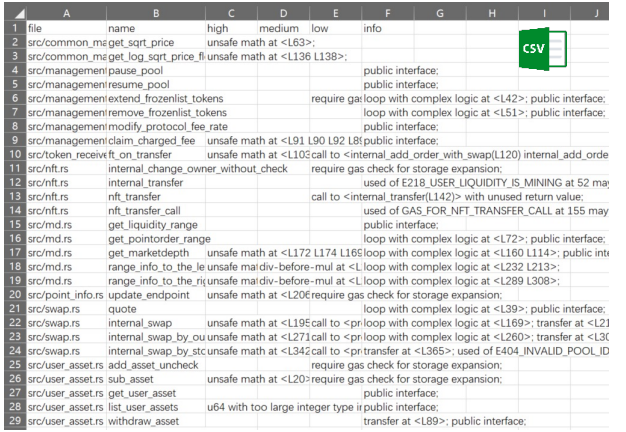
Launch Rustle with a single command in multiple platforms

- Easy to set up: run a few commands to install
- Easy to launch: **./Rustle –h**
- Easy to deploy: Linux, macOS, Docker

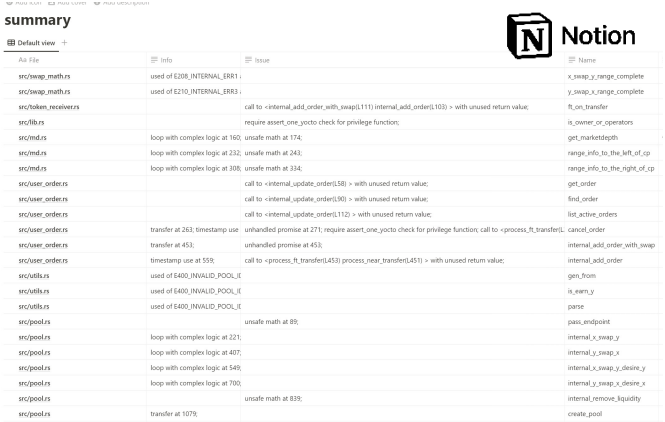
Usability

Results in different formats

CLI, CSV, Notion, and etc.



	A	B	C	D	E	F	G	H	I	J
1	file	name	high	medium	low	info				
2	src/common_ma	get_sqrt_price	unsafe math at <L63>;							
3	src/common_ma	get_log_sqrt_price_fl	unsafe math at <L136 L138>;							
4	src/managemen	pause_pool				public interface;				
5	src/managemen	resume_pool				public interface;				
6	src/managemen	extend_frozenlist_tokens			require gas	loop with complex logic at <L42>; public interface;				
7	src/managemen	remove_frozenlist_tokens				loop with complex logic at <L51>; public interface;				
8	src/managemen	modify_protocol_fee_rate				public interface;				
9	src/managemen	claim_charged_fee	unsafe math at <L91 L90 L92 L8			public interface;				
10	src/token_receive	ft_on_transfer	unsafe math at <L103			call to <internal_add_order_with_swap(L120) internal_add_orde				
11	src/nft.rs	internal_change_owner_	without_check			require gas check for storage expansion;				
12	src/nft.rs	internal_transfer				used of E218_USER_LIQUIDITY_IS_MINING at 52 may				
13	src/nft.rs	nft_transfer				call to <internal_transfer(L142)> with unused return value;				
14	src/nft.rs	nft_transfer_call				used of GAS_FOR_NFT_TRANSFER_CALL at 155 may				
15	src/md.rs	get_liquidity_range				public interface;				
16	src/md.rs	get_pointorder_range				loop with complex logic at <L72>; public interface;				
17	src/md.rs	get_marketdepth	unsafe math at <L172 L174 L16			loop with complex logic at <L160 L114>; public int				
18	src/md.rs	range_info_to_the_le	unsafe ma	div-before-mul at <L		loop with complex logic at <L232 L213>;				
19	src/md.rs	range_info_to_the_ri	unsafe ma	div-before-mul at <L		loop with complex logic at <L289 L308>;				
20	src/point_info.rs	update_endpoint	unsafe math at <L20			require gas check for storage expansion;				
21	src/swap.rs	quote				loop with complex logic at <L39>; public interface;				
22	src/swap.rs	internal_swap	unsafe math at <L195			call to <pr			loop with complex logic at <L169>; transfer at <L21	
23	src/swap.rs	internal_swap_by_ou	unsafe math at <L271			call to <pr			loop with complex logic at <L260>; transfer at <L30	
24	src/swap.rs	internal_swap_by_stc	unsafe math at <L342			call to <pr			transfer at <L365>; used of E404_INVALID_POOL_ID	
25	src/user_asset.rs	add_asset_uncheck				require gas check for storage expansion;				
26	src/user_asset.rs	sub_asset	unsafe math at <L20			require gas check for storage expansion;				
27	src/user_asset.rs	get_user_asset				public interface;				
28	src/user_asset.rs	list_user_assets	u64 with too large integer type i			public interface;				
29	src/user_asset.rs	withdraw_asset				transfer at <L89>; public interface;				



summary

Default view

Aa File	Info	Issue	Name
src/swap_math.rs	used of E208_INTERNAL_ERR1		x_swap_y_range_complete
src/swap_math.rs	used of E210_INTERNAL_ERR3		y_swap_x_range_complete
src/token_receiver.rs		call to <internal_add_order_with_swap(L111) internal_add_order(L103) > with unused return value;	ft_on_transfer
src/lib.rs		require assert_one_yocto check for privilege function;	is_owner_or_operators
src/md.rs	loop with complex logic at 160;	unsafe math at 174;	get_marketdepth
src/md.rs	loop with complex logic at 232;	unsafe math at 243;	range_info_to_the_left_of_cp
src/md.rs	loop with complex logic at 308;	unsafe math at 334;	range_info_to_the_right_of_cp
src/user_order.rs		call to <internal_update_order(L58) > with unused return value;	get_order
src/user_order.rs		call to <internal_update_order(L90) > with unused return value;	find_order
src/user_order.rs		call to <internal_update_order(L112) > with unused return value;	list_active_orders
src/user_order.rs	transfer at 263; timestamp use	unhandled promise at 271; require assert_one_yocto check for privilege function; call to <process_ft_transfer(L	cancel_order
src/user_order.rs	transfer at 453;	unhandled promise at 453;	internal_add_order_with_swap
src/user_order.rs	timestamp use at 559;	call to <process_ft_transfer(L453) process_near_transfer(L451) > with unused return value;	internal_add_order
src/utils.rs	used of E400_INVALID_POOL_ID		gen_from
src/utils.rs	used of E400_INVALID_POOL_ID		is_earn_y
src/utils.rs	used of E400_INVALID_POOL_ID		parse
src/pool.rs		unsafe math at 89;	pass_endpoint
src/pool.rs	loop with complex logic at 221;		internal_x_swap_y
src/pool.rs	loop with complex logic at 407;		internal_y_swap_x
src/pool.rs	loop with complex logic at 549;		internal_x_swap_y_desire_y
src/pool.rs	loop with complex logic at 700;		internal_y_swap_x_desire_x
src/pool.rs		unsafe math at 839;	internal_remove_liquidity
src/pool.rs	transfer at 1079;		create_pool

Usability

Extend Rustle to support locating new vulnerabilities

Welcome to Contribute

- Write an LLVM pass as detector plugin
- Add compiling option in **Makefile**
- Specify severity in **Rustle**
- Complete documentation and samples is preferred

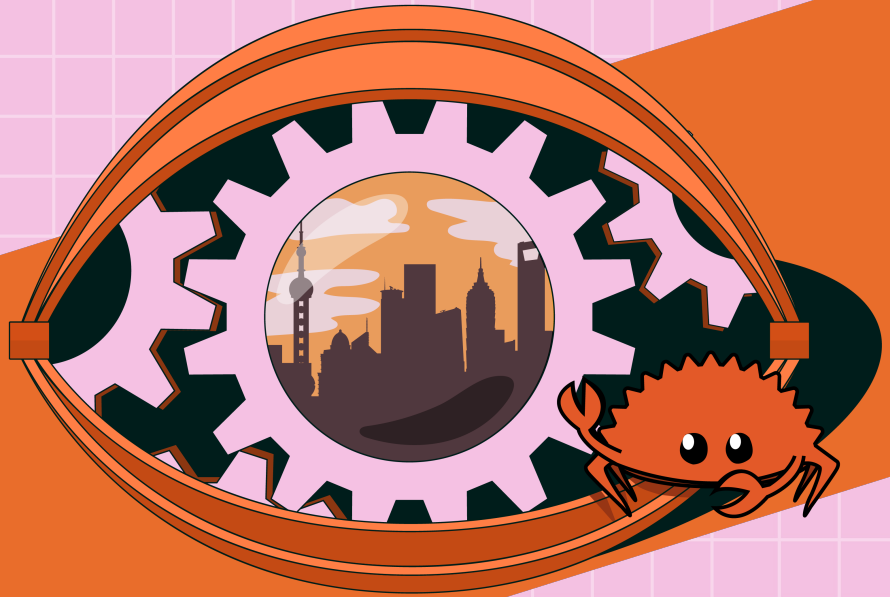
BlockSec Rustle

BlockSec

Conclusion

- Rustle is the first static analyzer for smart contracts in Rust and has been integrated into our workflow
- Rustle supports checking various types of security vulnerabilities and can identify them in the wild
- Rustle is highly scalable and can be easily extended to support new features
- Rustle is open source (https://github.com/blocksecteam/rustle) and powered by BlockSec (https://blocksec.com/)

Thank you !



Q & A

- Email: contact@blocksec.com
- Website: https://www.blocksec.com
- Medium: https://blocksecteam.medium.com