Lab 3 – Rust

CY5130 Computer System Security

Team Members:
Rama Krishna Sumanth Gummadapu and Nassim Bouchentouf-Idriss

Variable bindings:

When it comes to the security implications of using variables with Rust, it does not allow for the use for uninitialized variables. This could lead a program to reference data that is not intended to be referenced and could lead to information leakage.

Variable 1:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=46076ef8e71b8a12bd96587280e61b10

Root cause:
The variable type was not declared.

Variable 2:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=c2f3ed112c0c8196a3f0686f1c62a73d

Root cause:
The variable value was not defined.

Variable 3:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=ff64fad1fedb8af1d8c895b0ab6871b1

Root cause:
An immutable variable was reassigned value.

Variable 4:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=9621e0f503c4b7e82ef32c589f11e868

Root cause:
Value not initialized for the variable.

Functions:

When it comes to security implications of using functions in Rust may lead to calling functions which have an undefined behavior. The function may be deemed to be unsafe or functions which contains an unsafe method invoked in them.

Function 1:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=20ea0e763d831b1cdf0cff5368ac663a

Root cause:
Function not declared.

Function 2:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=4114e656ce1c0ab75ee8cc7cf8fd3183

Root cause:
Function parameter partially complete. The type of the parameter was missing.

Function 3:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=f6b5c2894a03d4537a36288024869018

Root cause:
Variable was not declared to be passed to the function as parameter.

Function 4:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=6fc0d94ac5be0b07b8505fdce1f465e7

Root cause:
The return type of the function is missing.

Function 5:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=f34ae2a4cee8d974c31a593a626107dd

Root cause:
Return statement expected as function declared explicit return type i32 but returns () which is the default return type if none is defined.

Primitive Types:

Integer overflow can be a security implication when using primitive types in Rust in a scenario where the value of the variable is set outside its range. In this situation when compiling in release mode, Rust does not check for panics. This can lead to an integer overflow and exiting with program with an error. It is important to explicitly wrap in order to achieve the expected variable value.

Primitive variables 1:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=155539b2bba37edfcf361ebff6a51e67

Root cause:
Boolean variable not declared for the variable to compile.

Primitive variables 2:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=87e781b3cbc10396aa90051a289fd721

Root cause:
Declare the undeclared variable and give variable different values for execution and check values for the output.

Primitive variable 3:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=07e916263186d0ead3bd88642fcefa77

Root cause:
Undeclared array attribute extraction.

Primitive variables 4:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=eeb1109c8410300b8363405022df0d8a

Root cause:
Variable not borrowed from the parent variable 'a' and not sliced to get the perfect match.

Primitive variables 5:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=2218cd70ea984c94e98d2236f53ac677

Root cause:
Unwrap variables not present for the names "name & age".

Primitive variables 6:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=f5ced0ca9eb1b0296b1e5c96a287dab3

Root cause:
Tuple extraction not present.

Strings:

The string type is part of the Rust's standard library. In this case, there could be a vulnerability identified in the standard library which could be exploited. In a situation where ranges are not being used to create string slices, this can be exploited as this could lead to the program to crash.

String 1:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=3b71b09a3c2d402ebca55a1c34128c27

Root cause:
String literal present instead of the String.

String 2:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=66900488296c53fdb00ceaf3a79555da

Root cause:
The string has to be borrowed. Only string reference is accepted.

String 3:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=27a288f18c98f1ab3184957b23e0008f

Root cause:
Where reference is needed and where reference is not needed is tested and added the functions respectively.

Move Semantics:

The security implications that can be exposed here can be related to memory safety violations. This can lead to certain security vulnerabilities that are related to unintentional data leakage, remote code execution. Double free error can lead to freeing memory twice which can lead to memory corruption. It is important to keep track in order to prevent data duplication and clear out any unused data on the heap.

Move Semantic 1:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=e0329cbd5fa6ca78dd71a160db83f662

Root cause:
Unable to modify the un mutable vector.

Move Semantic 2:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=e9b2ba80befa812b7477510af67544ce

Root cause:
Value moved instead of being borrowed.

Move Semantic 3:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=0b331c02b37b6bc6b4dadfdf8ed87c30

Root cause:
Immutable binding is present in the function parameter of fill_vec.

Move Semantic 4:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=a4e9b46e102792b175e96d90649a263f

No root cause, rewrite the method by creating variable in the function and passing it to main function.

Threads:

Thread safety bugs, which are similar to memory safety bugs, could lead to an invalid resource use. These security implications could involve privilege escalation, arbitrary code execution, and avoiding security checks. In this context, preventing data races can still be a difficult problem since the interaction between threads could lead to the code to be susceptible to more errors.

Thread 1:

Fix:
https://play.rust-lang.org/?version=stable&mode=debug&edition=2015&gist=568f55135554abf66d733361f63c6a7e

Root cause:
Mutable(MUTEX) not used to lock the thread and relieve race condition.

Reference:

https://hacks.mozilla.org/2019/01/fearless-security-memory-safety/
https://mssun.me/assets/rustrush-18-building-safe-and-secure-systems-in-rust.pdf
https://doc.rust-lang.org/std/primitive.unit.html
https://hacks.mozilla.org/2019/02/fearless-security-thread-safety/