**Best of Best 11**

**MC++ Report**

**김종명 멘토님 과제 #1 번**

**Best of Best 11기**

**보안제품 개발 희망 정경재**

**제출날짜:2022-07-07**

**목차**

1. **WeakPtr<>**
2. **scoped\_refptr<>**
3. **취약점 912947**
4. **취약점 998431**
5. **취약점 625404**
6. **취약점 925864**
7. **취약점 1082105**

Weak Ptr이 나온 배경은 다음과 같습니다.

만약 Shared Ptr A,B가 있을 경우, 이 A, B가 서로를 참조하는 경우가 있다고 생각한다면 이경우 순환 참조하는 Shared Ptr이 생성됩니다.

하지만 이 순환 참조하는 것이 생성에는 문제가 없으나 반대로 이를 해제하는 과정에서 문제가 발생하게 됩니다.

Shared Ptr이 해제가 되기 위해서는 각 Sharad Count가 0이 되어야 하지만 문제는 둘이 순환하기 때문에 A가 해제되기 위해서는 B가 해제되어야 하고 반대로 B가 해제되기 위해서는 A가 해제되어야 하는 OS의 DeadLock과 유사한 형태가 나타나게 됩니다.

따라서 이러한 문제점을 보완하기 위해서 나온 것이 바로 Weak Ptr입니다.

Diagram

Description automatically generated

Weak Ptr이 없을 경우 해당 문제를 풀기위한 방법은 2가지입니다.

1. 일반적인 ptr: 잊는다면 적적한 자원해제 불가.
2. Shared ptr: 순환 참조의 문제가 발생

따라서 이 둘의 중간성격을 가지는 Weak Ptr이 등장했습니다.

따라서 스마트 포인터로 안전한 객체 참조를 지원하며 Shared Ptr과 달리 Reference Count를 증가시키지 않습니다. 또한 이름대로 약한 ptr이므로 객체를 직접적인 참조가 불가능하여 반드시 Shared Ptr로 변경해서 사용해야 한다는 점을 지니고 있습니다.

만약 객체가 사라졌다면 빈 Shared Ptr이 반환되고 있다면 해당 객체를 가리키는 Shared Ptr이 반환됩니다.

Scoped\_refptr이 나온 배경은 다음과 같습니다.

이 역시 앞의 Weak Ptr과 같이 Shared Ptr의 문제점을 보안하기 위해 고안된 스마트 포인터 입니다.

만약 하나의 object에 대해 Shared Ptr을 생성한다면 Shared Ptr은 다음과 같은 결과를 발생할 것입니다.

Diagram

Description automatically generated

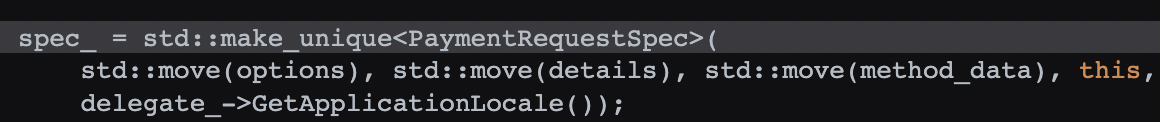
따라서 같은 Object에서 같은 Control Block이 생성되지 않게 방어하기 위해 나온 것이 바로 Scoped\_refptr입니다.

하지만 이러한 Scoped\_refptr는 다중 쓰레드에서 문제가 발생할 확률이 높습니다.

다중 쓰레드의 환경에서 ref Count를 증가시키기 위해 다수의 쓰레드가 동시에 접속한다면 결국은 문제가 발생하기 때문에 따라서 RefCountedThreadSafe와 RefCounted가 Scoped\_refptr의 내부에 있어 다중 쓰레드의 경우는 RefCountedThreadSafe을 통해 안전하게 값을 증가시키고 단일 쓰레드의 경우에는 직접적으로 접근해 값을 증가시키는 방향으로 적절하게 성능을 유지하는 방향으로 해당 문제를 방어했습니다.

취약점 912947은 UAF(Used After Free) 일종의 취약점으로 써 할당 받은 공간을 사용하다 해제를 한 이후 메모리 할당 시 같은 공간을 재사용하면서 생기는 취약점입니다.

해당 912947는 PaymentRequest 객체가 사용됩니다.

해당 PaymentRequest는 PaymentRequestSec이 있고 해당 PaymentRequestSec은 Unique Ptr로 작성되어 있습니다.   
해당 PaymentRequestSec은 PaymentRequest::init에 대한 호출 중에 초기화 되는 문제를 지니고 있습니다.

Text

Description automatically generated

그림과 같이 초기화 PaymentRequest에서 PaymentRequest::Show()를 호출하면 내부의 PaymentRequestSec의 포인터는 새 PaymentRequestSheetController에 전달되면 문제가 발생하게 됩니다.

Graphical user interface, text

Description automatically generated

해당 문제를 방어하기 위해서 위 사진과 같이 새로운 변수인 is\_initialized 처음으로 생성할 시는 false로 선언하고 이후 초기화가 되면 해당 변수를 true로 변경하고 난 이후 만약 새로운 init이 들어올 때 해당 변수를 검사해서 만약 true라면 error로그를 띄워주고 종료하는 식으로 해당 문제를 방어했고 추가적으로 만약 생성되지 않았는데 해당 함수를 호출하는 경우도 방어하는 코드 역시 추가 되었습니다.

Text

Description automatically generated

취약점 99843은 Smart Pointer에 관련된 취약점을 가진 버그입니다.

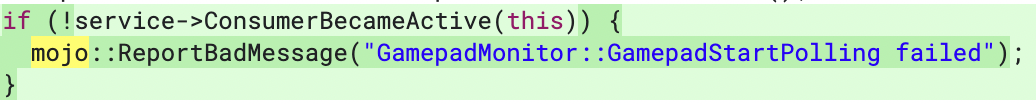
Mojo 인터페이스를 통해 GamepadMonitor::GamepadStartPolling을 먼저 호출하지 않고 GamepadConsumer::GamepadStartPolling를 먼저 호출하면 Consumer\_는 GamepadConsumer(GamepadMonitor)를 GamepadConsumer를 포함하지 않는 문제가 발생합니다. 이는 결국은 consumer\_.end()가 역 참조되고 DCHECKER가 CHECkER로 변경이 됩니다.

Text

Description automatically generated

해당 사진을 보다 싶이 해당 변수를 바인딩을 진행한 이후에 초기화 없이 해당 변수의 gamepadstoppolling을 불러 취약점을 발생시킵니다.

이를 해결하기 위해서 아래와 같은 방법으로 취약점을 해결했습니다.



미리 실행하기 전에 init이 있는지 확인하는 코드와 A picture containing text

Description automatically generated

기존의 서비스가 void타입에서 인자를 전달해주는 방법으로 변경하여 문제가 있다면 false, 아니면 true를 발생시켜 이와 같은 방법들을 통해 해당 취약점을 수정했습니다.

취약점 625404의 취약점은 AddEventMatcher(event\_filters.cc)에서Smart pointer가 특정 조건에서 std::move를 하지 않는 문제점이 있습니다.

Text, letter

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Text

Description automatically generated

위의 사진에서 보다 싶이 int id작업을 할 때 ParseEventMatcher는 EventMatcher 객체를 생성하고, filter는 인자에 전달되어서 AddEventMatcher가 filter\_Weak를 해제하는 UAF 문제가 발생합니다.

Table

Description automatically generated따라서 해당 문제를 해결하기 위해 위와 같이 코드를 수정했습니다.

parseEventMatcher(std::move(filter))의 결과 값을 id에 담아 해당 값이 -1이라면 종료하도록 하고 그것이 아니라면 기존과 같이 동작하도록 코드가 변경되었습니다.

취약점 925864는 Unique\_ptr로 인해 발생하는 UAF 취약점입니다.  
Graphical user interface, text, application

Description automatically generatedFileSystemOperationRunner operations\_는 생성된 모든 FileSystemOperation 개체를 필드를 통해 관리합니다.  
operations\_즉 map<id, unique\_ptr<FileSystemOperation>>, FileSystemOperation에 대한 unique\_ptr 포인터를 보유하고 있습니다.



해당 사진에서 OperationID가 int 이므로 emplace 중에 로드 된 연산이 너무 많아 값이 overflow 된다면 이전에 Map에 로드 된 id가 다시 연산을 수행하게 될 수 있습니다.  
Text

Description automatically generated with low confidence

이로 인해서 기존의 두 번째 로딩 시 operations\_unique\_ptr이므로 때문에 키가 같을 수 없으므로 두 번째 로딩은 실패하고 op2 unique\_ptr는 그 자리에서 소멸되어 그 안에 저장된 원래 포인터는 Empty가 아니므로 포인터가 가리키는 FileSystemOperation도 소멸되고 operation\_raw가 댕글링 포인터가 됩니다.

이를 보호하기 위해 int를 uint\_64를 통해 방지했습니다.

Graphical user interface, text

Description automatically generated

해당 취약점 역시 Unique\_ptr으로 인한 UAF취약점 입니다.

Graphical user interface, text, application, email

Description automatically generated  
navigator.credentials.create결국 새로운 인증 장치를 추가하기 위해 AuthenticatorAdded로 전송됩니다.

Graphical user interface, text, application

Description automatically generated

  
active\_authenticators\_요소를 삽입할 때 키 충돌이 발생하면 AuthenticatorState unique\_ptr authenticator\_state개체의 원래 포인터가 즉시 소멸됩니다.

삽입이 실패한 후 소멸된 객체의 원시 포인터를 weak\_authenticator\_state매개 FidoRequestHandlerBase::InitializeAuthenticatorAndDispatchRequest변수로 callback에 바인딩하고 PostTask를 현재 스레드의 메시지 큐에 바인딩하여 실행을 기다리고 있기에 메시지 루프가 callback을 가져와 실행할 때 실행됩니다.   
Chart

Description automatically generated with low confidence

해당 취약점은 미리 삽입 되었는지 확인하는 것과 다양한 조건을 통해 위의 사진과 같이 수정되었습니다.