# CWE Detail – CWE-1244

## Description

The product uses physical debug or test  
 interfaces with support for multiple access levels, but it  
 assigns the wrong debug access level to an internal asset,  
 providing unintended access to the asset from untrusted debug  
 agents.

## Extended Description

Debug authorization can have multiple levels of  
 access, defined such that different system internal assets  
 are accessible based on the current authorized debug  
 level. Other than debugger authentication (e.g., using  
 passwords or challenges), the authorization can also be  
 based on the system state or boot stage. For example, full  
 system debug access might only be allowed early in boot  
 after a system reset to ensure that previous session data is  
 not accessible to the authenticated debugger. If this protection mechanism does not ensure that  
 internal assets have the correct debug access level during  
 each boot stage or change in system state, an attacker could  
 obtain sensitive information from the internal asset using a  
 debugger.

## Threat-Mapped Scoring

Score: 3.0

Priority: P2 - Serious (High)

## Observed Examples (CVEs)

**•** CVE-2019-18827: After ROM code execution, JTAG access is disabled. But before the ROM code is executed, JTAG access is possible, allowing a user full system access. This allows a user to modify the boot flow and successfully bypass the secure-boot process.

## Related Attack Patterns (CAPEC)

* CAPEC-114

## Attack TTPs

**•** T1548: Abuse Elevation Control Mechanism (Tactics: privilege-escalation, defense-evasion)

## Modes of Introduction

**•** Architecture and Design: N/A

**•** Implementation: N/A

## Common Consequences

**•** Impact: Read Memory — Notes:

**•** Impact: Modify Memory — Notes:

**•** Impact: Gain Privileges or Assume Identity, Bypass Protection Mechanism — Notes:

## Potential Mitigations

**•** Architecture and Design: For security-sensitive assets accessible over debug/test interfaces, only allow trusted agents. (Effectiveness: High)

**•** Architecture and Design: Apply blinding [REF-1219] or masking techniques in strategic areas. (Effectiveness: Limited)

**•** Implementation: Add shielding or tamper-resistant protections to the device, which increases the difficulty and cost for accessing debug/test interfaces. (Effectiveness: Limited)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Demonstrative Examples

**•** This means that since the end user has access to JTAG at system reset and during ROM code execution before control is transferred to user software, a JTAG user can modify the boot flow and subsequently disclose all CPU information, including data-encryption keys.

**•** The following code [REF-1377] illustrates an instance of a vulnerable implementation of debug mode. The core correctly checks if the debug requests have sufficient privileges and enables the debug\_mode\_d and debug\_mode\_q signals. It also correctly checks for debug password and enables umode\_i signal.

## Notes

**•** Relationship: CWE-1191 and CWE-1244 both involve physical debug access,  
 but the weaknesses are different. CWE-1191 is effectively  
 about missing authorization for a debug interface,  
 i.e. JTAG. CWE-1244 is about providing internal assets with  
 the wrong debug access level, exposing the asset to  
 untrusted debug agents.