주 제

임베디드시스템 보안

대구가톨릭대학교 장정숙, 전용희

차 례

I. 서 론

임베디드시스템은 자동차, 냉장고, 그리고 셀룰러 폰을 비롯하여 PDA, 디스크 컨트롤러, 라우터, 게이트웨이, 방화벽, 저장 서버, 웹 서버 그리고 스마트 홈의 온도조절장치로부터 마이크로웨이브 조절기까지 다양하게 다른 모양과 형태를 가지고 있다. 임베디드시스템의 주요 경향은 더 강력해지고 자동화되며 그리고 본질적으로 인터넷의 연결을 가능하게 하는 것이다. 임베디드시스템은 나아가 우리들의 일상을 유비쿼터스 환경으로 변화시킬 것이다. 임베디드시스템은 인터넷이 제공하는 대역폭의 증가를 필요로 하며 경제적으로 사회적으로 많은 영향을 끼치는 잠재력을 가지고 있다[1]. 이러한 임베디드시스템 장치들의 계속적인 확산과 상호연결성은 보안 문제와 프라이버시 문제를 야기 시킨다. 비록 보안과 프라이버시 문제들이 인터넷 공간에서 생성되어 확인이 어렵다 하드라고 이들 문제는 같은 기본적인 기능을 가진다. 그것은 보안과 프라이버시 문제가 유사하기 때문이다. 임베디드시스템에서 구현하는 보안은 범용 컴퓨터와는 많이 다르다. 임베디드시스템은 다음과 같이 일반적으로 엄격하게 제한된 자원을 가진다[2]:

- 임베디드시스템의 휘발성과 비휘발성 저장소는 일반 컴퓨터에서 보다 더 적다
- 임베디드시스템의 CPU 속도와 대역폭은 일반 컴퓨터에서 보다 더 느리다.

임베디드시스템 보안을 위해서 IPSec(Internet Protocol Security), SSL(Secure Socket Layer), WTLS(Wireless Transport Layer Security)같은 다양한 보안 프로토콜과 암호화알고리즘을 적용하는 보안 고려사항을 들 수 있다[3, 4]. 많은 임베디드시

스템은 운영 환경과 자원에 대하여 제한을 받는다. 임베디드시스템 보안은 기능적인 보안 측면에서 시스템 구조 측면, 즉 하드웨어/소프트웨어 측면으로 보안 고려사항이 이동되고 있다. 몇 가지 요인은 다음과 같다[5].

- 소프트웨어, 물리적 및 사이드-채널 공격과 같은 보안 위반 공격기술의 증가로 인하여, 악성 엔티티의 논리적 혹은 물리적 접근에 대하여 임베디드시스템이 안전할 필요가 있다.
- 보안과 비용, 혹은 보안과 성능 사이 상충관계(tradeoff)
- 제한적인 배터리, 저장소 및 계산 용량 같은 자원 제약 하에서의 보안
- 보안 메커니즘과 표준화 지원
- DoS(Denial of Service) 공격과 DRM(Digital Right Management)에 대한 보안 전문가와 임베디드시스템 설계자의 협력

본 논문에서는 임베디드시스템 보안에 대하여 고찰한다. 먼저 II장에서는 임베디드시스템 보안 요구사항에 대하여 살펴보고, III장에서는 임베디드시스템의 공격을 분류하며, IV장에서는 임베디드시스템이 가지고 있는 취약성과 현재 임베디드시스템 보안을 위한 보안 메커니즘에 대하여, V장에서는 임베디드시스템 공격에 따른 대응책을 기술한다. 대응책에서는 손상-저항 메커니즘과 시스템 보안 처리구조를 제시한다. VI장에서는 임베디드시스템 보안 사례연구를 기술하며 마지막으로 VII장에서 끝을 맺는다.

II. 보안 요구사항

임베디드시스템은 주요한 기능을 제공하며 악성 엔티티에 의해 파괴될 수 있다. 보안 요구 사항은 고려되는 관점에 의존하여 달라진다. 예를 들면, 고가의 이동 전화기에는 무선 음성, 멀티미디어 그리고 데이터 통신의 능력이 있다. 보안 요구 사항은 이동 전화기 내부 컴포넌트 제조자, 이동 전화기 제조자, 이동 전화 서비스 제공자, 컨텐츠 제공자, 이동 전화기 사용자 각각의 관점에 따라서 달라 질 수 있다. 컨텐츠 제공자의 주요 관심은 전달된 멀티미디어 컨텐츠의 복제를 방지하는 것이며, 종단 사용자의 주요 관심은 전화기에 의한 저장되고 통신된 개인적인 데이터의 보안일 것이다. 이동 전화기 제조사 관점에서 보면 전화기에 내재하고 있는 펌웨어 소유의 비밀에 관계된 것이다. 이들 경우에 잠재적인 악성 개체들의 집합도 또한 다양하다. 예를 들면 컨텐츠 제공자의 관점에서는 이동 전화기의 종단 사용자가 비신뢰된 개체가 될 수 있다.

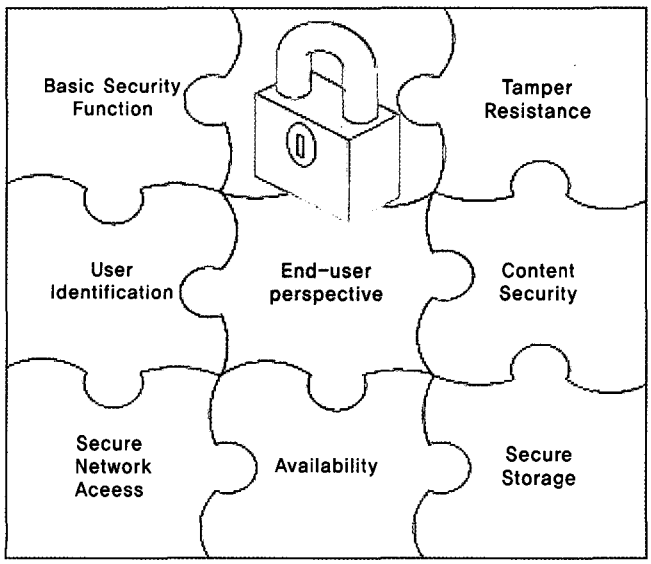
본 절에서는 임베디드시스템을 위한 보안 모델에 적용되는 전형적이고 광범위한 보안 요구사항을 기술한다[5]. 두 개체가 잠재적인 공격자들에 의하여 접근이 가능한 공중 네트워크 혹은 통신 채널을 사용하여 민감한 정보를 송수신할 때, 데이터 기밀성, 데이터 무결성, 그리고 피어 인증(peer authentication) 같은 보안 기능을 제공하는 것이 이상적이다. 데이터 기밀성은 원하지 않는 도청자들로부터 중요한 정보를 보호하는 것이고, 데이터 무결성은 정보가 비합법적으로 변경되지 않는 것을 보장하는 것이고, 피어 인증은 정보가 정당한 사용자에 의해 송수신되는 것을 검증한다. 기술된 보안 기능은 현재 넓은 범위의 응용에서 사용되는 임베디드시스템의 보안 요구사항이다.

보안은 컴퓨팅과 통신 시스템에 오랜 동안 관여해 왔다. 그리고 실제적인 보안문제를 해결하도록 노력해 왔다. 대칭 암호화, 공개 키 암호화 그리고 해시 함수를 포함하여 암호 알고리즘은 보안 메커니즘을 구성하는 빌딩 블록으로 사용되어 왔다. IPSec와 SSL

같은 네트워크 보안 프로토콜은 통신 개체들 사이 인증을 가능하게 하고 그리고 통신 데이터의 기밀성과 무결성을 보장하기위한 것이다. 실제로, 암호알고리즘 같은 기능적인 보안 메커니즘은 단독으로 완전한 보안 솔루션이 되진 못한다.

(그림 1)은 종단 사용자의 관점에서 본 임베디드시스템의 보안 요구사항이다. 기밀성, 무결성, 인증 요구사항의 집합을 정의하기 위하여 (그림 1)의 요구사항을 기본 보안 기능(Basic Security Function)이라 한다[5]. 임베디드시스템에 대한 접근은 선택된 집합의 권한을 부여받은 사용자에 대하여만 제한되어야 한다. 반면에 네트워크 혹은 서비스에 대한 접근은 디바이스가 권한이 부여되면 제공된다. 따라서 임베디드시스템에 대한 인증은 사용자별로 되는 사용자 식별(user identification)이고, 네트워크에 대한 인증은 안전한 네트워크 접근(secure network access)으로 차이가 있다. 여기서 기본 보안 기능뿐만 아니라 바이오메트릭과 접근 통제 같은 인증 메커니즘을 이용하여 사용자 혹은 호스트 인증이 가능하다. 또 다른 주요한 보안 기능 중의 하나는 임베디드시스템의 가용성이다. 예로써 악성 개체가 성능을 감소시키거나 혹은 완전한 서비스 거부 공격을 초래하여 임베디드시스템이 기능을 수행할 수 없도록 할 수 있다.

임베디드시스템은 시스템의 수명동안 중요하고 민감한 정보를 보호할 필요가 있다. 안전한 저장은 시스템의 외부 혹은 내부이든, 임베디드시스템의 저장장치에 정보를 안전하게 보관하는 것이다. 컨텐츠 보안 혹은 디지털 저작권 관리(DRM)는 시스템 내에서 사용되는 디지털 컨텐츠의 권리를 보호하는 것이다. 마지막으로, tamper resistance는 이러한 보안 요구사항들이 장치가 악성 파티의 손에 들어갔을 때에나, 물리적 혹은 논리적으로 탐사될 수 있을 때에도 유지되어야 함을 의미한다.



(그림 1) 종단-사용자 관점에서 임베디드시스템의 일반적 보안 요구사항

III. 임베디드시스템 공격 분류

(그림 2)는 임베디드시스템에 대한 공격의 넓은 분류를 보여준다. 최상위 레벨에서 공격은 그들의 기능적인 목표를 기반으로 세 가지 주요 범주로 분류된다[6]:

- 프라이버시 공격: 임베디드시스템 내에서 저장되고, 통신되고, 다루어지는 민감한 정보 획득을 목표로 한다.
- 무결성 공격: 임베디드시스템과 연관된 데이터 혹은 코드 변경을 시도한다.
- 가용성 공격: 부적절한 시스템 자

원에 의해 시스템의 정상적인 기능을 붕괴시켜, 정상적인 운용을 불가하게 한다.

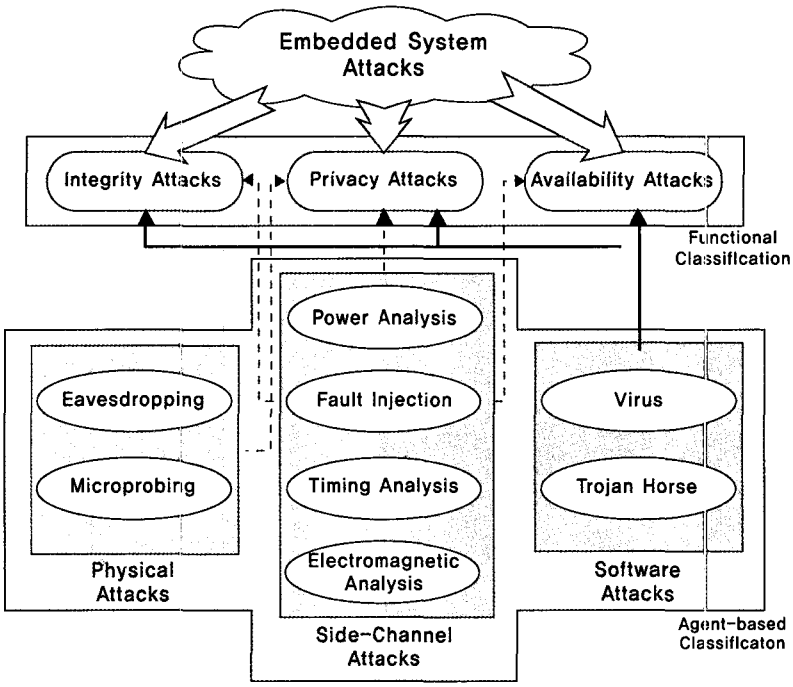
- 사이드-채널 공격: 암호 운용을 수행하는 동안, 실행 시간, 전력 소비, 결함 존재 시의 행위와 같은 시스템의 관측 특성을 기반으로 하는 공격

임베디드시스템 공격의 분류에서 두 번째 레벨은 공격을 개시하기 위하여 사용되는 에이전트 혹은 수단을 기반으로 한다. 이들 에이전트들은 (그림 2)에서처럼 세 가지 주요 범주로 전형적으로 그룹화 된다 [6]:

- 소프트웨어 공격: 바이러스, 트로이 목마, 웜 등과 같은 소프트웨어 에이전트를 통해 개시되는 공격
- 물리적 혹은 침입 공격: 칩, 보드, 혹은 시스템과 같은 어떤 레벨에서 시스템으로 물리적인 침입 공격

공격 개시에 사용되는 에이전트들은 시스템 실행에 관여하지 않은 수동적이거나 혹은 타겟 시스템의 운용에 적극적으로 관여할 수 있다. 무결성과 가용성 공격은 어떤 식으로든 시스템과의 간섭을 필요로 하며, 그리하여 활성 에이전트를 통하여만 개시 될 수 있다.

임베디드시스템에 대한 공격이 여러 가지 범주로 분류되어 있지만, 실제로 공격자들은 목표 달성을 위해 여러 가지 기술을 결합하여 사용한다. 예를 들면, 물리적 공격은 사이드-채널 공격이 일어나기 전에 선행 공격으로 사용될 수 있다.



(그림 2) 임베디드시스템 공격 분류[6]

3.1 소프트웨어 공격

소프트웨어 공격은 애플리케이션 코드를 다운로드하고 실행할 수 있는 임베디드시스템에 대한 주요한 위협을 나타낸다. 물리적 및 사이드-채널 공격에 비하여, 소프트웨어 공격은 비교적 싸고 쉽게 이루어질 수 있기 때문에, 안전한 임베디드시스템 설계에 심각한 즉각적인 도전이 된다. 이들 공격은 바이러스, 웜, 트로이 목마 같은 악성 에이전트를 통해 구현된다. 그리고 무결성, 비밀성 및 가용성의 모든 관점에서 시스템 보안을 침해할 수 있다[7, 8].

악성 소프트웨어 에이전트는 종단 시스템 구조의 약점을 이용하여 소프트웨어 공격을 시작한다. 소프트웨어 결점은 취약성 혹은 노출(exposures)이라고 말할 수 있는 소프트웨어 내의 결점 때문에 대표적으로 발생한다[7]. 노출은 공격자가 접근을 획득하기 위하여 간접적으로 이용하는 진입점이며, 취약성은 공격자에게 종단 시스템에 대한 직접적인 접근을 획득하는 것을 허용한다.

버퍼 오버플로 공격은 운영 체제와 응용 소프트웨어에서 일반적인 허점으로 소프트웨어 공격동안 이용될 수 있다. 오버플로 문제는 버퍼가 열악한 경계(bound) 검사를 가지고 있을 때마다 야기될 수 있다. 버퍼 경계는 부정확한 루프 경계, 포맷 스트링 공격 등으로 인해 위반될 수 있다. 버퍼 오버플로 영향으로 스택 메모리, 힙 그리고 함수 포인터에 겹쳐 쓰기가 발생한다. 공격자는 가까이 저장된 프로그램 주소를 겹쳐 쓰기하기 위하여 버퍼 오버플로를 사용할 수 있다. 이렇게 함으로써, 공격자가 악성 코드로 통제를 전환하여, 악성 코드 실행을 통하여 바람직하지 않은 효과를 가질 수 있게 한다.

가. 소프트웨어공격 예제: 하드웨어 바이러스[6, 9]

운영체제 커널에 대한 소프트웨어 공격으로 원격 슈퍼 유저 로그인이 가능한 루트킷(rootkit)에 의해 실행되는 공격이 있다. 커널은 시스템에 대한 완전한 접근을 가지고 그리고 주소 공간의 어떤 부분과도 통신이 가능하다. 이것의 의미는 공격자가 마더보드상의 BIOS(Basic Input Output System) 메모리 혹은 주변장치의 하드웨어에서 읽기 혹은 쓰기가 가능하다는 것이다.

초창기에는 BIOS 메모리는 소프트웨어로부터 갱신될 수 없는 ROM(Read Only Memory) 혹은 EPROM(Erasable Programmable Read Only Memory) 칩 안에 저장되었다. 이러한 구 시스템은 칩이 대체되거나 혹은 수동적인 소거와 재작성을 필요로 한다. 이것은 매우 비용 효율적이지 못하므로, 신규 시스템은 플래시 ROM으로 알려진 EEPROM (Electronic EPROM) 칩을 채택하였다. 플래시 ROM은 소프트웨어로부터 재작성이 가능하다. 현대의 임베디드시스템은 흔히 플래시 ROM을 포함한다.

어떤 컴퓨터에 여러 가지의 컨트롤러 카드와 마더보드에 수 메가바이트의 플래시 ROM을 가지고 있다. 이들 플래시 ROM 칩은 대부분 전부 이용되지 않고 백도어 정보와 바이러스를 저장할 수 있는 많은 공간을 남겨두고 있다. 공격자들은 이들 메모리 공간을 악용 할 수 있다. 공격은 감사하기가 어렵고 대부분 시스템상에서 수행되는 소프트웨어에겐 보이지 않는다. 그러한 하드웨어 메모리에 접근하기 위하여 드라이버 레벨 접근을 요구한다. 더욱이 이 메모리는 재부팅과 시스템 재 설치와는 상관이 없다. 누군가 바이러스 감염을 의심하여 테입이나 백업으로부터 시스템을 복구하는 것도 도움이 되지 않는다. 이와 같은 하드웨어 바이러스는 임베디드시스템의 운용에 심각한 위협을 줄 수 있다.

EEPROM 메모리는 많은 시스템에 일반적이다. 이더넷 카드, 비디오 카드, 그리고 멀티미디어 주변장치들은 EEPROM 메모리를 모두 포함하고 있다. 하

드웨어 메모리는 플래시 펌웨어를 포함하거나 데이터 저장을 위해 사용된다. 비휘발성 메모리칩이 하드웨어 장치에서 많이 사용된다: TV 튜너와 원격 제어, CD 플레이어, 각종 셀 전화기, 팩스, 카메라, 라디오, 자동차의 에어백, 잠금-방지 브레이크, 주행거리계, 키없는 진입시스템, 프린터, 복사기, 모뎀, 페이지, 위성수신기, 바코드 리더, 판매지점 터미널, 스마트카드, 잠금 박스, 차고 문 개방장치, 그리고 테스트 및 측정장비. EEPROM 칩이 파괴적인 코드 저장을 위한 주요한 영역으로 남아 있다. 더 많은 임베디드 장치들이 사용됨에 따라, EEPROM기반 바이러스는 더 많이 적용가능하고 위험해질 것이다.

3.2 물리적 공격과 사이드-채널 공격

다양한 물리적인 공격과 사이드 채널 공격이 임베디드시스템에 대하여 개시될 수 있다. 이들 공격의 다수는 스마트카드 같은 저급-종단 임베디드시스템의 문맥에서 조직화 되어 왔다. 그러나 이들 공격은 나날이 정교해지고 많은 전자 시스템에 대하여 설치될 수 있는 것으로 보여, 안전한 임베디드시스템 설계에 심각한 도전으로 생각된다[10, 11].

가. 물리적 공격

회로 보드 상의 임베디드시스템에 대한 물리적 공격은 컴포넌트 사이 통신을 도청하기 위하여 탐사를 이용하여 개시 된다. 그러나 SoC(System On Chip)에 대하여는 정교한 마이크로 탐사 기술이 필요하다. 그러한 공격의 첫 번째 단계는 de-packaging이다. De-packaging은 증기 산을 사용하여 실리콘을 덮고 있는 수지(resin)를 녹임으로 칩 패키지를 제거함을 말한다. 다음 단계는 현미경의 대칭적 조합을 사용하여 레이아웃을 재구성한다. 그리고 덮고 있는 계층을 제거한다. 레이아웃을 재구성하는 동안, 칩의 내부가 여러 가지 입상성에서 추론될 수 있다. 데이터와 주소 버스들, 메모리 그리고 프로세서 경계 같은 칩 내부의 더 높은 레벨의 구조가 쉽게 추출된다. 아울러 명령어 해독기와 프로세서 내의 연산장치(ALU), ROM 셀 같은 더 낮은-레벨 구조의 상세한 조감도 또한 획득되어진다. 수동적인 다이크로 탐사 혹은 전자빔 현미경 같은 기술들이 전형적으로 de-package된 칩 내 컴포넌트의 버스들과 인터페이스의 값들 관측에 대표적으로 사용된다.

칩 레벨에서의 물리적 공격은 비싼 인프라 요구 때문에 사용이 비교적으로 어렵다. 그러나 일단 수행되면 성공적인 비-침입 공격의 설계에 대한 선행으로 이용된다. 예를 들어, 레이아웃 재구성은 선택된 칩 영역 주위에 전자기적 방사 모니터링을 수행하기 전에 필요하다. 비슷하게 암호 루틴과 제어 데이터 같은 ROM 내용에 대한 지식은 공격자에게 적합한 비-침입 공격의 설계에 도움이 되는 정보를 제공할 수 있다[10].

나. 전력 분석 공격

어떤 하드웨어 회로의 전력 소모는 내부에 있는 유선에서의 스위칭 활동의 합수이다. 스위칭 활동이 데이터에 의존하므로 암호화 알고리즘에서 사용되는 키가 넓은 범위의 입력 데이터 상에서 모아진 전력 소모 통계치로부터 추론될 수 있다는 것은 당연하다. 이 공격은 암호키와 관련된 데이터를 처리할 때 칩이 사용하는 전력량과 그렇치 않을 경우의 전력량을 비교 분석하는 방법을 이용한다. 이 공격을 전력 분석 공격이라 부르고 스마트카드 같은 임베디드시스템을 부수는데 매우 효과적으로 이용 될 수 있다. 전력 분석 공격은 두 가지 주요한 클래스로 나눈다: 단순한 전력 분석(SPA: Simple Power Analysis), 차등 전력 분석(DPA: Differential Power Analysis) 공격[11].

• SPA

SPA공격은 어떤 시스템에서 암호 관련 계산의 전력 프로파일이 암호 정보를 밝히기 위하여 직접 사용될 수 있는 관측에 의존한다. DES 알고리즘을 구현한 ASIC에 대한 전력 소모 프로파일이 한 가지 예로 볼 수 있다. 프로파일 관찰로부터 DES 알고리즘의 16라운드를 쉽게 확인 할 수 있다. 실제로 SPA 공격은 전사적 공격(brute-force attack)을 보충하거나 단순화하는데 유용한 것으로 밝혀졌다.

• DPA

DPA 공격은 전력 소모 데이터로부터 암호 키를 추론하기 위하여 통계 분석을 적용한다. 이런 공격은 SPA 기술과 관련된 측정 에러와 잡음의 단점을 극복하기 위하여 차등적인 추적의 개념을 적용한다. DPA는 DES 구현뿐만 아니라, 공개키 암호시스템을 부수는 것에 사용되었다.

다. 타이밍 공격

타이밍 공격은 암호화관련 계산 작업의 실행 시간이 데이터 의존적이라는 관측을 이용한다. 암호 알고리즘이 암호키와 관련된 정보를 처리하는데 걸리는 시간과 그렇지 않는 데이터를 처리하는데 걸리는 시간 사이의 차를 분석하여 암호 키 추론에 사용한다. 데이터는 명령 실행 시간 변이와 성능 최적화에 의존적인 특징을 보인다[12].

라. 결점 주입 공격

결점 주입 공격은 공급 전압, 클록, 온도, 방사 등과 같은 시스템의 외부적인 변수와 환경적인 조건에 의존한다. 주입된 결점은 컴포넌트에서의 결함을 유발하고, 일시적이거나 영구적이 된다. 다음과 같은 몇 가지 방법에서 시스템 보안을 손상시킬 수 있다[6].

- 가용성 공격: 결점은 시스템의 정상적인 기능을 중단시키기 위하여 주입된다. 칩 상의 임베디드 시스템 버스가 영구적 결점을 통해 컴포넌트 사이 통신을 수행 할 수 없도록 될 수 있다.
- 무결성 공격: 안전하거나 안전하지 않는 코드 혹은 메모리 같은 컴포넌트에 저장되어 있는 데이터를 붕괴시키는 것에 사용된다.
- 프라이버시 공격: 하나의 예로써 CRT(Chinese Remainder Theorem) 최적화를 사용하는 RSA(Rivest-Shamir-Adleman) 구현을 포함하는 암호 키를 찾기 위하여 결함 주입 공격을 사용하는 것이다. RSA에서 모듈러 지수 동작의 성능 향상을 의도하는 최적화는 결점 주입 공격에 대한 취약점을 증가시킨다[13].
- 선행 공격: 결함 주입 기술이 소프트웨어 공격에 대한 선행으로 또한 이용된다. 예를 들어, 열 때문에 발생하는 간단한 메모리 결점이 실행 환경의 완전한 통제를 가지기 위하여 프로세서 상의 신뢰되지 않은 프로그램에 의하여 이용될 수 있다[14].
- 전자기적 분석 공격: 전자기적 분석 공격(EMA: Electromagnetic analysis attack)은 80년대 이후 잘 서류화되어 있으며, 비디오 디스플레이 장치로 부터의 전자기적 방사가 스크린 컨텐츠를 재구성하기 위하여 사용되는 것에서 보여 진다. 이 공격은 장치에 의해 방출하는 전자기적 방사 측정을 시도하여 민감한 정보를 획득한다[15].

IV. 취약성과 보안 메커니즘

4.1 임베디드시스템 취약성

임베디드시스템 상의 모든 알려진 공격은 기능적

보안 메커니즘과 그것들의 암호 알고리즘의 구현 및 설치에 있어서의 약점을 이용하는 것으로 알려져 있다. 이들 약점은 공격자로 하여금 보안 솔루션의 이론적인 강도를 완전하게 우회하거나 상당히 약화시키는 것을 허용한다. 다음에 임베디드시스템에서의 구현 취약성을 기술한다[7].

- 비신뢰된 환경에서 운영: 많은 임베디드시스템은 비신뢰된 소유자의 물리적 소유 하에서 조차 안전한 운영을 보장해야 한다. 장치의 본질적인 물리적 보안에 의존하고, 시스템의 부분이 악성 개체에 의해 물리적으로 접근이 가능하지 않다고 가정한다면, 안전한 임베디드시스템 설계는 보다 쉬울 것이다. 그러나 임베디드시스템은 때때로 복잡한 신뢰 관계에서 작업하기를 요구한다. 임베디드시스템에 대한 또 다른 요구사항은 휴대 가능해야하며 분실 및 도난이 가능한 작은 크기를 유지해야 하며, 상당기간 동안 비신뢰적 엔티티의 통제하에서 운영하여야 한다는 것이다.
- 네트워크 유발 취약성: 점차적으로 네트워킹 능력을 가지는 임베디드시스템의 수가 증가하고 있으며, 이는 많은 공격 소스에 노출되게 한다. 이것은 보안 메커니즘을 부수기위해서 장치의 물리적인 소유가 필요하지 않다는 것이다. 무선 연결을 가지는 장치들 혹은 인터넷에 연결된 장치들이 가장 취약하다.
- 다운로드된 소프트웨어 실행: 임베디드시스템의 종단 사용자들에게 더 풍부한 기능과 향상된 고객 맞춤형을 제공하기위한 움직임이 자주 비-신뢰된 소프트웨어 실행 능력을 요구한다. 예를 들면 인터넷으로부터 다운로드된 프리웨어 혹은 제 3 자 소프트웨어 같은 것이 있다. 바이러스, 웜, 트로이 목마 같은 소프트웨어 프로그램들이 보안 공격을 개시하는데 사용된다. 이 문제는 임베디드시스템의 소프트웨어 콘텐츠가 증가함에 따라서 더욱 악화될 것이다.
- 복잡한 설계 프로세스: 엄중한 설계 소요시간과 비용 제한으로 복잡한 임베디드시스템이 복수 근원지로부터의 컴포넌트를 사용하여 조립된다. 시스템 보안을 보증하는 책임은 대표적으로 말단 생산품 제작자나 혹은 말단 생산품을 기반으로 서비스를 제공하는 엔티티의 몫이다. 그러나 보안을 보증하기 위하여 각 시스템 컴포넌트를 미리-검정하는 것이 불가능하다. 더구나, 시스템의 각 부분이 자체적으로는 안전하다 할지라도, 부품들의 합성으로 새로운 취약성에 노출될 수 있다고 알려져 있다. 적합한 설계 방법의 결여로 임베디드시스템 설계 동안의 보안의 모델링과 최적화는 이미 열악한 것으로 보인다.

[6]에서는 높은 레벨의 보안을 성취하기 위하여 변경-저항(tamper-resistant) 구현이 실체화된 강한 기능적 보안 메커니즘을 요구하고 있다.

4.2 임베디드 소프트웨어 취약성

임베디드시스템 소프트웨어는 보안 취약성의 주요 소스이며, 복잡성, 확장성 및 연결성과 같은 세 개의 주요 과제가 있다[6].

- 복잡성: 소프트웨어는 복잡하며, 가까운 미래에는 더욱 더 복잡해 질 것이다. 더 많은 라인의 코드는 버그와 보안 취약성의 가능성을 증가시킨다. 임베디드시스템이 인터넷과 통합되고 더 많은 코드가 추가됨에 따라, 임베디드시스템 소프트웨어는 더욱 복잡해지고 있다. 복잡성 문제는 불안전한 프로그래밍 언어, 즉 버퍼 오버플로 같은 단순한 공격을 막을 수 없는 C와 C++의 사용

에 의해 악화된다. 반면에 효율성 측면에서 C와 C++는 임베디드시스템을 위해 매우 대중적인 언어이다.

- 확장성: 자바나 .NET같은 현대의 소프트웨어 시스템은 확장 구축이 가능하다. 오늘날 운영체제는 동적으로 적재가능한 장치 드라이버와 모듈을 통하여 확장성을 지원한다. 향상된 임베디드 시스템은 확장성 있게 설계된다. 그러나 확장성 있는 시스템의 성질이 소프트웨어 취약성을 방지하기 어렵게 한다.
- 연결성: 더욱 더 많은 임베디드시스템이 인터넷에 연결되고 있다. 높은 연결성 정도가 작은 결점을 전파시키고 대량의 보안 위반을 야기할 수 있다. 유비쿼터스 네트워킹은 더 많은 공격, 더 많은 임베디드 소프트웨어 시스템 공격, 열악한 소프트웨어 보안 실제로부터 더 많은 위협이 존재함을 의미한다.

4.3 보안 메커니즘

요구된 보안 기능을 성취하기위해 몇 가지 존재하는 보안 메커니즘을 기술한다. 2장에서 기술된 요구되는 기본 보안 기능은 세 가지 다른 클래스의 암호화 알고리즘을 사용하여 이루어질 수 있다[3, 4].

- 대칭적 암호(symmetric ciphers)
- 비대칭적 암호(asymmetric ciphers)
- 해쉬 알고리즘(hash algorithms)

RSA, Diffie-Hellman 같은 비대칭 암호는 큰 정수의 모듈러 지수 같은 더 계산 집중적인 수학 함수에 의존하기 때문에 대칭 암호보다는 속도가 훨씬 느리다. 이런 이유로, 대칭 암호가 대량 데이터 암호화에 전형적으로 사용된다. 반면에 비대칭 암호는 공중망 상의 대칭 암호화를 위한 비밀 키 전송에 사용된다.

II장에서 기술된 여러 가지 요구사항을 만족하기 위한 보안 솔루션은 보안 프로토콜의 문맥 안에서 이들 암호화 알고리즘의 조합을 사용하는 보안 메커니즘에 대표적으로 의존한다. 다양한 보안 기술과 메커니즘들은 특정한 보안 서비스를 제공하기위해서 이러한 암호화 알고리즘을 위주로 설계되었다. 몇 가지 예는 다음과 같다.

- 안전한 통신 프로토콜, 즉 보안 프로토콜은 임베디드시스템을 위한 안전한 통신 채널을 보장하는 방법을 제공한다. IPSec과 SSL은 널리 보급되어있는 보안 프로토콜이며, 각각 가상 사설망(Virtual Private Network)과 안전한 웹 처리를 위해 광범위하게 사용된다[16, 17].
- 디지털 인증서는 엔티티와의 식별을 관련시키는 방법을 제공하는 반면, 지문 인식과 음성 인식 같은 바이오메트릭 기술은 종단 사용자 식별을 도와준다. 자필 서명과 동등한 전자적기능인 디지털 서명은 데이터의 근원 인증과 신원 검증에 사용된다[18].
- OpenIPMP, ISMA, 그리고 MOSES 같은 디지털 저작권 관리(DRM) 프로토콜은 비합법적인 사용에 대하여 어플리케이션 컨텐츠를 보호하는 안전한 프레임워크를 제공한다[19, 20].
- 안전한 저장소와 안전한 실행은 시스템 구조가 보안 고려사항을 위하여 맞추어질 것을 요구한다. 예로써, 메모리의 보호된 영역에 대한 불법 접근을 차단하기 위하여 전용 하드웨어사용, 시스템 상에 실행되는 펌웨어와 소프트웨어의 인증, 어떤 애플리케이션이나 프로세스와 관련된 코드와 데이터의 기밀성과 무결성 유지, 메모리 계위를 통하여 데이터의 기밀성과 무결성을 유지하기 위한 하드웨어 및 소프트웨어 기술, 버스 탐사를 방지하기 위하여 프로세스 내부에서 암호화 코드의 실행 등이 있다[21].

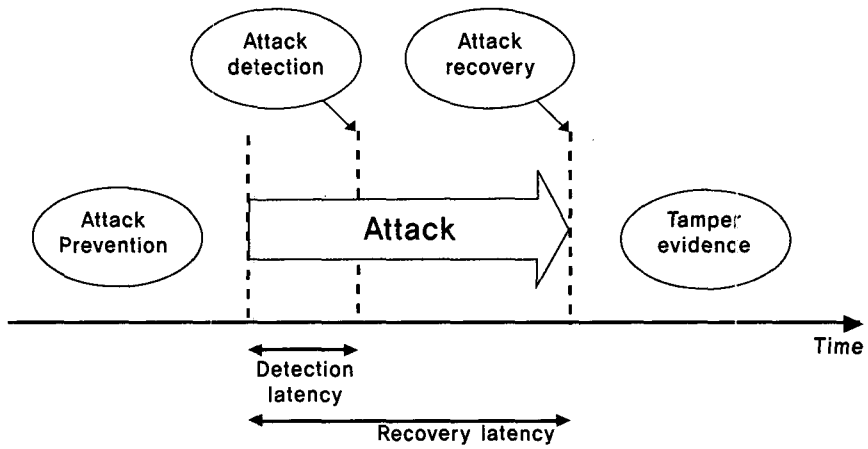
V. 임베디드시스템 보안 대응책

본 장에서는 III장에서 기술된 여러 가지 공격에 대하여 임베디드시스템을 강화하기위한 설계 기술들에 대하여 기술한다. 실제적인 접근방법으로 손상(tamper)-저항(resistant) 메커니즘에 대하여 소개하고자 한다[6]. 보안을 위한 임베디드 처리 구조에 대하여도 기술하고자 한다[5].

5.1 손상-저항 설계 기술

먼저 손상-저항 설계에 대한 이해와 비교를 위해서 손상-저항 설계의 목표를 분석한다. 그림 3은 공격에 대한 손상에 저항하는 설계 접근의 목표를 보여준다. 손상-저항 설계 기술은 다음과 같다.

- 공격 방지(Attack prevention): 공격 방지 기술은 임베디드시스템상에서 공격을 개시하는 것을 더 어렵게 만든다. 이들 기술은 물리적인 보호 메커니즘(예, 패키징), 타이밍과 전력 특성이 데이터와 독립적인 회로 구현을 위한 하드웨어 설계, 소프트웨어 설계(예, 실행 전에 소프트웨어 인증)를 포함한다.
- 공격 탐지(attack detection): 공격은 어떤 방지 기술에도 불구하고 개시될 수 있다. 공격 탐지 기술은 되도록 가능한 빨리 공격 탐지를 시도한다. 공격 개시와 탐지 사이의 경과된 시간 간격이 취약성 기간을 나타내며, 가능한 낮게 유지될 필요가 있다.
- 공격 회복(Attack recovery): 공격 회복은 공격에 대응하고, 시스템이 안전한 운영으로 돌아오도록 보장하는 기술을 의미한다. 공격 회복 기술은 시스템을 잠그고 추가 운영을 위하여 못쓰게 하고, 메모리상의 민감한 데이터를 없애고, 보안 경보를 디스플레이하고 시스템을 재부팅한다. 공격 회복 스킴의 설계는 보안의 수준과 사용자 불편 사이의 상충관계(tradeoff)를 포함한다.
- 손상 증거(tamper-evident): 손상 증거 기술은 후에 감사를 위해서, 임베디드시스템에서 공격의 지속적인 기록을 반박할 수 없게 유지하는 것



(그림 3) 손상-저항 설계 접근의 목표

이다. 손상 증거는 악성 엔티티에 의하여 반전시킬 수 없는 메커니즘을 요구한다.

가. 소프트웨어 공격 대응책

소프트웨어 공격에 대한 대응책은 다음의 고려사항을 가지고 설계된다:

- 임베디드시스템 내 소프트웨어 실행의 모든 단계 동안 민감한 코드와 데이터의 비밀성과 무결성을 보장한다.
- 어떤 프로그램을 실행하기 위하여 보안 관점에서 안전하다는 것을 확실성을 가지고 결정한다.
- 공격에 대하여 시스템 취약성을 만드는 소프트웨어 내 보안 허점을 제거한다.

대부분의 시스템 단계 대책은 적어도 위에 기술된 처음 고려사항을 다룬다. 대책의 공통적인 특징은 하드웨어와 소프트웨어 변경의 결합을 통하여, 부트 프로세스, 정상적 실행, 인터럽트 모드 같은 실행의 다른 단계 동안에, 레지스터 · 메모리 영역 · 보안 보조-프로세서 같은 시스템의 다른 부분으로, 여러 가지 운영체제?다운로드 코드 같은 소프트웨어 컴포넌트의 접근 제어를 포함한다. 효율적인 대응책은 시스템으로 하여금 전력 공급 상태로부터 시작하여 시스템의 보안에 대한 보증을 제공하는 것을 허용해야 하며, 대부분의 대책은 다양한 하드웨어와 소프트웨어 자원 사이에서 보안 경계(security perimeter) 혹은 신뢰 경계(trust boundary) 개념을 정의 한다. 이것은 시스템으로 하여금 신뢰경계의 위반을 탐지하도록 하며 그리고 회복 메커니즘을 실행하도록 한다. 그리하여 신뢰경계는 보안에 대한 분명한 결정 시스템을 위해서 자연스럽고 편리한 기초를 제공한다.

① 하드웨어 지원

손상-저항을 구현하는 통상적인 접근은 별도의 안전한 보조 프로세서 모듈을 사용한다. 안전한 보조-프로세스에서 보내질 필요가 있는 모든 민감한 정보는 암호화된다[22].

② 안전한 부팅

초창기 신뢰경계의 개념을 조사한 것으로 IBM PC 구조에서 부트 프로세스의 안전문제를 조사하는 AEGIS 구조가 있다. AEGIS는 부트 프로세스의 계층화된 성질을 이용하여 문제에 대한 계층적인 해결책을 제공한다. 무결성 검사는 부트 프로세스 컴포넌트의 해시 값을 계산하고 그리고 안전하게 저장된 값과의 비교를 수행한다[23].

③ 운영제제 증진

대부분 보안 스킴은 민감한 코드 혹은 데이터를 보호하기위해서 운영체제 변경에 의존한다. 하나의 예로써, 강한 프로세스 고립을 제공하고 그라고 프로세스-레벨 증명을 수행하는 모드가 있다. 프로세스 고립은 한 프로세스의 개별적인 자원이 또 다른 프로세스로부터 보호됨을 보장하는 것이며, 증명은 코드가 프로세스들과 장치들 사이에 통신 채널을 설정하기 전에 인증됨을 보장한다[24].

④ 소프트웨어 인증과 검증

시스템에서 소프트웨어의 안전한 실행은 실행 전에 검증이 요구된다. 알려진 소프트웨어의 조각은 무결성을 검증하기위해 일반적인 기술, 즉 코드의 해쉬 혹은 체크섬을 계산하고 그리고 전에 계산된 값들과 비교를 통해 검증한다. 비-신뢰 어플리케이션코드 실행을 위해서 증명을 수반하는 코드(proof-carrying code) 같은 소프트웨어 메커니즘을 사용할 수 있다[25].

나. 물리적 및 사이드 채널 공격에 대한 대응책

패키징 기술, 암호프로세서의 사용을 통한 중요 정보의 물리적 보안, 환경적인 공격 보호 대책, 입력 데이터에 둔감한 타이밍과 전력 같은 성질이 되도록 하드웨어와 소프트웨어의 각별한 설계 등이 물리적 및 사이트-채널공격에 대응하는 통상적인 방법이다. 이러한 대응책은 아래와 같다.

① 물리적 공격 보호

여러 향상된 패키징과 공격 대응 기술은 미국연방정보처리 표준(FIPS140-2)에 의해 권고되었다. 해당 표준은 4 가지 레벨의 물리적 보안 요구사항을 명시하고 있다.

- 보안 수준 1: 최소한의 물리적 보호를 요구한다.
- 보안 수준 2: 봉인 같은 손상-명백한 메커니즘의 추가를 요구한다.
- 보안 수준 3: 더 강한 탐지와 대응 메커니즘을 기술한다.
- 보안 수준 4: 환경적인 실패 보호와 테스팅을 의무화한다.

② 버스 암호화

버스 탐사 공격에 대항하는 좋은 대응책은 전역적인 버스로 보내는 모든 정보를 암호화하는 프로세서를 사용하는 것이다. 프로세서에서 높은 수준의 보안을 달성하려면 상당한 성능 오버헤드를 수반한다.

③ 사이드-채널 공격 보호방법

사이드 채널 공격에 대한 여러 가지 대응책은 전력, 타이밍, 전자기적 방사 같은 사이드-채널 정보의 모니터링과 분석에 대하여 임베디드시스템을 취약하게 만드는 증상을 제거하는 것이다. 무작위화(Randomization)는 공격자가 언제 어떤 작업이 수행되는지를 알 필요가 있는 사이드-채널 공격에 대하여 효율적인 방법으로 빈번하게 사용된다.

전력 분석 공격에 대응하는 기술은 중요한 정보를 숨기기 위한 데이터 마스킹, 감소된 신호 진폭의 사용, 전력 측정 데이터에 대한 잡음의 도입 등을 포함한다. 환경적인 변화를 모니터하는 센서는 여러 가지 결점 주입 공격 탐지에 효과적이다.

5.2. 보안을 위한 임베디드 처리구조

임베디드시스템은 요구되는 보안 기능을 이상적으로 제공하고 효율적으로 구축해야 하며, 악성 공격에 대하여 방어기능을 가져야 한다. 본 절에서는 임베디드시스템 보안에 대한 대응책으로, 보안을 위한 임베디드 처리 구조에 대하여 기술한다. (그림 4)는 안전한 임베디드 프로세싱 시스템을 위한 구조적인 설계 방법을 보여주고 있다[26].

(그림 4)에서 첫 번째 줄은 다른 마크로-구조 모델을 보여주고 있다. 여기에는 내장된 일반적인 목적의 프로세서(EP)를 포함하고 있으며, ASIP(Application-Specific Instruction set Processor), 프로세서 버스에 연결된 하드웨어 가속기를 가진 EP 등을 포함하고 있다. 두 번째 줄은 명령어-집합 구조와 적절한 베이스 프로세서 튜닝을 위한 마이크로-구조 선택사항을 열거하고 있다. 세 번째 줄은 선택되거나 혹은 설계되어야 하는 보안 프로세싱 특징을 나타내고 있다. 네 번째 줄은 임베디드 프로세서와 임베디드시스템 설계에서 공격에 저항적인 특징에 대한 선택을 포함한다. (그림 4)에서는 소프트웨어 공격뿐만 아니라 물리적인 공격에 대하여도 보호한다. 이것은 안전한 메모리 공간관리를 위한 증진된 메모리 관리, 프로세스 고립 구조, 전력 분석 공격 저지를 위한 추가적인 여분의 회로, 그리고 결점 탐지 회로를 포함하고 있다.

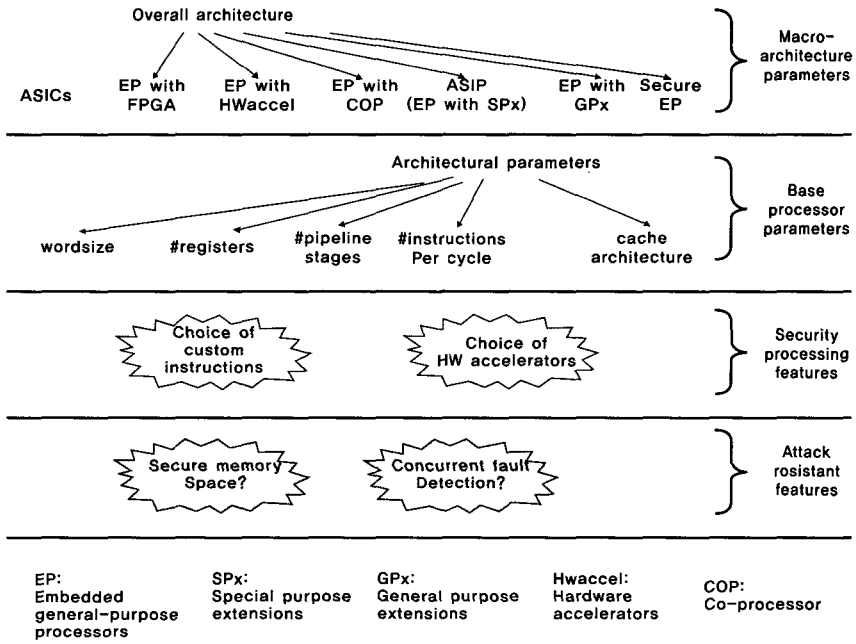
보안 처리 구조에서는 하드웨어-접근과 소프트웨

어-접근을 제공한다. 하드웨어-접근에서는 하드웨어에서 주어진 암호 알고리즘을 구현하기 위하여 ASIC를 사용한다. 하드웨어 알고리즘은 한개 혹은 단지 몇 개의 암호법이 필요하여, ASIC이 대량으로 생산될 때 매우 비용 효율적이다. 그러나 다중 보안 프로토콜, 표준, 그리고 많은 장치와의 상호 운용성을 지원하기 위하여 다양한 암호법이 요구될 때, 비용과 유연성 측면에서 효율성이 낮아진다.

소프트웨어-접근은 네트워크 링크 속도에서 보안 프로토콜과 암호화 프로세싱 실행을 위해 임베디드 범용 프로세서(EP) 코어를 사용한다. 이 방법은 프로세싱 갭(processing gap)과 배터리 갭(battery gap)이라는 두 가지 문제가 있다. 프로세싱 갭이란 암호를 사용하는 표준 보안 프로토콜의 계산적인 요구가 임베디드시스템에서 이용할 수 있는 프로세서 능력보다 더 높게 되는 사실을 의미하고, 배터리 간격은 배터리로 전력을 공급하는 무선 장치에서 보안 프로세싱동안 상당한 추가적인 에너지 소모가 있음을 의미한다.

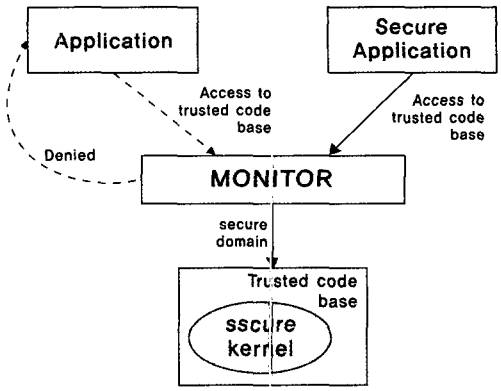
복합적인 하드웨어-소프트웨어 접근에서는 더 효율적인 보안 기능을 보인다. 특히, 대부분 보안 프로세싱에서는 성능이 중요하므로, 하드웨어 가속기와 함께 범용 임베디드 프로세서 코어를 사용한다. 보안 프로세싱은 또한 보안 프로토콜 처리를 포함한다. IPSec과 SSL 같은 안전한 네트워크 통신 프로토콜에는 처리 오버헤드 최소화, 데이터 버퍼링 그리고 메모리 라운드-트립을 수행한다. 보안 프로토콜 엔진으로는 IP 네트워크 프로토콜을 처리하는 IPSec 보안 프로토콜이 있다.

안전한 임베디드시스템을 위한 적절한 공격-저항



(그림 4) 안전한 정보 처리를 위한 구조적인 설계 공간

구조의 구체화에는 디지털 저작권 관리(DRM)를 위한 TCPA(Trusted Computing Platform Alliance)를 비롯하여 NGSC(Next Generation Secure Computing Base)와 TCG(Trusted Computing Group) 등이 있다[27, 28].



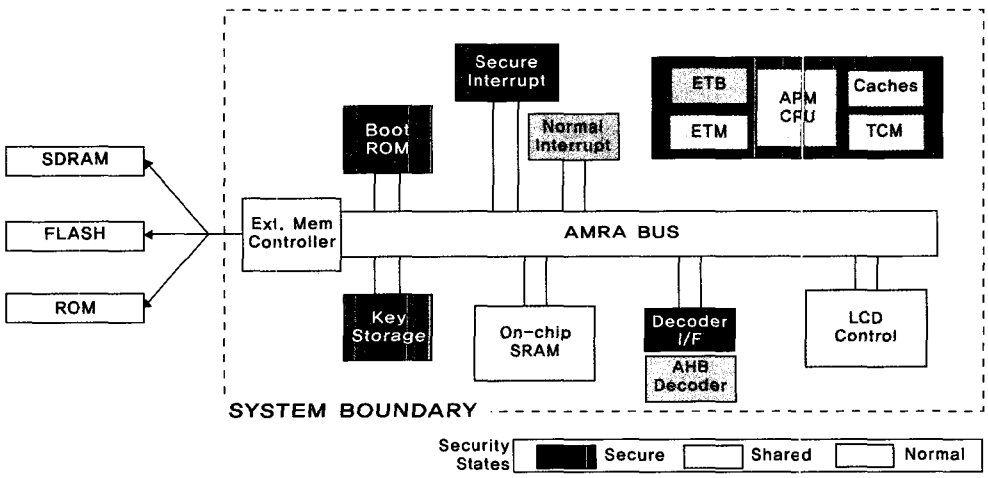
(그림 5) ARM TrustZone: 악성 소프트웨어 공격 보안

VI. 사례 연구

ARM(Advance Risk Machine)은 임베디드 프로세서와 SoC시장에서 TrustZone 기술이라 불리는 안전한 프로세서 코어를 제안하였다. 이 기술은 로그인 시 패스워스를 훔치는 공격에 대항한다(그림 5참조)[29].

소프트웨어 공격에 대항하는 TrustZone 보안 기술 목표는 ARM 기반 SoC 구조의 중요한 정보와 다른 하드웨어/소프트웨어 부분으로의 접근에 있어 명백한 분리를 확립하는 것이다. 이것은 프로세서의 안전한 영역에 거주하는 신뢰된 코드 베이스(trusted code base)를 사용한 안전한 도메인에 의하여 이루어진다. 신뢰된 코드 베이스는 시스템 부트 순서로부터 시작하여 전체 시스템의 보안을 조정하는 책임이 있다. 그 외에, 신뢰된 코드는 키 조작을 포함하는 모든 보안 태스크를 책임진다.

신뢰된 코드 베이스는 (그림 6)에서 보여주는 것과 같이 안전한 도메인으로 분리되어 보호된다. 비-안전



(그림 6) 영역 분리된 임베디드 SoC 구조 컴포넌트

어플리케이션은 신뢰된 코드 베이스에 대한 접근이 거절된다. 신뢰된 어플리케이션은 안전한 도메인으로 접근을 제공하기 전에 확인된다. 이 경계는 구조 전체에서 "S-bit"로 불리는 보안 태그를 첨가함으로써 실행된다. S-bit는 시스템의 보안 동작 상태를 정의하며 안전한 시스템의 부분을 정의하기 위하여 사용된다. S-bit로의 접근은 모니터 모드(monitor mode)로 불리는 별도의 프로세서 운영모드를 통하여 이루어진다. 모니터 모드는 어플리케이션에 의해 행해지는 데이터와 명령어 접근이 허용되는 것을 검증하면서, S-비트를 통제하고, 안전한 상태와 비-안전한 상태 사이에서 안전한 전이를 보장할 책임이 있다.

TrustZone 기술은 운영체제와는 독립적인 신뢰된 코드 베이스를 실행하고, 신뢰된 소프트웨어의 인증을 가능하게 하는 구조적-레벨 보안 솔루션을 제공하고, 악성 소프트웨어 공격에 대한 보호를 제공한다.

VII. 맺음말

본 논문에서는 점차적으로 증가하고 있는 임베디드시스템에 대하여 무엇보다도 필요한 보안에 대하여 기술하였다. 먼저, 임베디드시스템 보안에서 사용자의 관점에서 보는 일반적인 보안 요구사항에 대하여 살펴보았고, 임베디드시스템에 대한 공격에 대하여 분류하였으며, 임베디드시스템이 가지고 있는 취약성들과 현재 임베디드시스템 보안을 책임지고 있는 보안 메커니즘에 대하여 기술하였다. 그리고 임베디드시스템 공격에 따른 대응책을 소개하였다. 대응책으로는 공격 수행에 따른 손상에 저항하는 메커니즘과 하드웨어와 소프트웨어 측면에서 보안을 위한 임베디드 처리 구조, 그리고 임베디드시스템 보안에 대한 사례를 기술하였다.

현재까지 알려지지 않는 보안문제도 있을 수 있다. 향후 안전한 임베디드시스템을 위해서, 종합적인 정보보호 대책과 전역적이고 능동적인 통합보안에 관한 체계적인 연구가 필요하다고 생각된다.

[참 고 문 헌]

[1] Philip Koopman, "*Embedded System Security,*" *EMDEDDED COMPUTING*, pp.95-97, July 2004.

[2] William A. Arbaugh, Leedert van Doorn, "*Embedded Security: Challenges and Concerns*", *IEEE Security*, pp. 40-41, 2001.

[3] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 1998.

[4] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*. John Wiley and Sons, 1996.

[5] Paul Kocher, Ruby Lee, Gary McGraw, Anand Raghunathan and Srivaths Ravi, *"Security as a New Dimension in Embedded System Design"*, *ACM*, pp. 753-760, June 2004.

[6] S. Ravi, A. Raghunathan, and S. Chakradhar, *"Tamper Resistance Mechanisms for Secure Embedded Systems," in Proc. Int. Conf. VLSI Design*, Jan. 2004.

[7] *Common Vulnerabilities and Exposures*. (http://cve.mitre.org/).

[8] *Virus Information*. Computer Security Resource Center, National Institute of Standards and Technology (http://csrc.nist.

gov/virus/).

[9] G. Hoglund and G. McGraw, *Exploiting Software: How to Break Code* http://www.exploitingsoftware.com), Addison-Wesley, 2004.

[10] O. Kommerling and M. G. Kuhn, *"Design principles for tamper-resistant smartcard processors," in Proc. USENIX Workshop on Smartcard Technology (Smartcard '99)*, pp. 9-20, May 1999.

[11] P. Kocher, J. Jaffe, and B. Jun, *Introduction to differential power analysis and related attacks.* (http://www.cryptography.com/resources/ whitepapers/).

[12] D. Brumley and D. Boneh, "Remote Timing Attacks Are Practical ," in *Proc. 12th USENIX Security Symp.*, pp. 1-14, Aug. 2003.

[13] D. Boneh, R. DeMillo, and R. Lipton, "On the importance of checking cryptographic protocols for faults", *in Proc. of Eurocrypt '97*, pp. 37-51, 1997.

[14] S. Govindavajhala and A. W. Appel, "Using Memory Errors to Attack a Virtual Machine," in Proc. *IEEE Symposium on Security and Privacy*, pp. 154-165, May 2003.

[15] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," *in Proc. Cryptographic Hardware and Embedded Systems*, pp. 251-261, 2001.

[16] *IPSec Working Group.* http://www.ietf.org/html.charters/ipsec-charter.html.

[17] *SSL 3.0 Specification.* http://wp.netscape.com/eng/ssl3/.

[18] *Biometrics and Network Security.* Prentice Hall PTR, 2003.

[19] *OpenIPMP.* http://www.openipmp.org.

[20] *Internet Streaming Media Alliance.* http://www.isma.tv/home.

[21] *Discretix Technologies Ltd.* (http://www.discretix.com).

[22] B. Yee, *Using Secure Co-processors.* PhD thesis, Carnegie Mellon University, 1994.

[23] A. Arbaugh, D. J. Farber, and J. M. Smith, "A Secure and Reliable BootStrap Architecture," *in Proc. of IEEE Symposium on Security and Privacy*, pp. 65-71, May 1997.

[24] D. Lie, C. A. Thekkath, and M. Horowitz, "Implementing an untrusted operating system on trusted hardware," in *Proc. ACM Symposium on Operating Systems Principles*, pp. 178-192, Oct. 2003.

[25] G. C. Necula and P. Lee, "*Proof-Carrying Code*," Tech. Rep. MU-CS-96-165,Carnegie Mellon University, Nov. 1996.

[26] S. Ravi, A. Raghunathan, and N. Potlapally, "Securing wireless data: System architecture challenges," in *Proc. Intl. Symp. System Synthesis*, pp. 195-200, October 2002.

[27] *Next-Generation Secure Computing Base (NGSCB).* Microsoft Inc. (http://www.microsoft.com/resources/ngscb/productinfo.mspx).

[28] *Trusted Computing Group.* (https://www.trustedcomputinggroup.org/home).

[29] R. York, *A New Foundation for CPU Systems Security. ARM* Limited (http://www.arm.com/armtech/TrustZone?OpenDocument).

장정숙

1989년 ~ 1991년 경일대학교 공과대학 컴퓨터공학과 (공학사)
1992년 ~ 1995년 대구가톨릭대학교 교육대학원 전자계산교육전공(석사)
1998년 ~ 2004년 현재 대구가톨릭대학교 대학원 컴퓨터 · 정보통신공학 전공 (이학박사)
2004년 ~ 현재 대구가톨릭대학교 컴퓨터 · 정보통신공학부 IT교수
관심분야 : 임베디드 네트워크 보안, BcN & QoS 보안, 홈네트워크 보안, 통신망 성능분석

전용희

1971년 ~ 1978년 고려대학교 전기공학과
1985년 ~ 1987년 미국 플로리다공대 대학원 컴퓨터공학과
1987년 ~ 1992년 미국 노스캐롤라이나주립대 대학원 Elec. and Comp. Eng. 석사, 박사
1978년 ~ 1978년 삼성중공업(주)
1978년 ~ 1985년 한국전력기술(주)
1979년 ~ 1980년 벨기에 Belgatom 연수
1989년 ~ 1989년 미국 노스캐롤라이나주립대 Dept of Elec. and Comp. Eng. TA
1989년 ~ 1992년 미국 노스캐롤라이나주립대 부설 CCSP (Center For Comm. & Signal Processing) RA
1992년 ~ 1994년 한국전자통신연구원 광대역통신망연구부 선임연구원
1994년 ~ 현재 대구가톨릭대학교 컴퓨터 · 정보통신공학부 교수
2001년 ~ 2003년 대구가톨릭대학교 공과대학장 역임
2004년 ~ 2005년 한국전자통신연구원 정보보호연구단 초빙연구원
관심분야 : 네트워크 보안, BcN QoS & Security, 통신망 성능분석