원자력시설 핵심디지털자산에 대한 코드 난독화 적용에 관한 연구

김상우, 김시원, 변예은, 권국희
한국원자력통제기술원
kjoey@kinac.re.kr, swkim@kinac.re.kr, hibye@kinac.re.kr, vivacita@kinac.re.kr

Applying Code Obfuscation to Vital Digital Assets at the Nuclear Facilities

Sangwoo Kim, Siwon Kim, Yeeun Byun, Kookheui Kwon
Korea Institute of Nuclear Nonproliferation and Control (KINAC)

요 약

원전에 대한 사이버위협이 지속됨에 따라 IAEA 및 각국에서는 원전 사이버보안 강화를 위해 노력하고 있다. 그 일환으로 국내에서는 규제기준 KINAC/RS-015를 통해 원전 내 안전·보안·비상대응 기능과 관련된 필수디지털자산에 대한 사이버보안 규제를 수행하고 있으나 원전 사고와 직접적으로 관련된 자산에 대해서는 보다 강화된 보안조치를 적용하여 보안성을 높이고자 한다. 이러한 강화 조치의 하나로 '코드 난독화 적용'이 있으며 이에 대해 상세히 살펴보고자 한다.

1. 서론

원전 제어시스템의 디지털화가 지속되고 원전에 대한 사보타주를 목적으로 한 악성코드들이 점차 늘어나고 있다. 대표적인 예로 2010년 이란 원자력시설을 대상으로 한 스턱스넷 사건이 있었으며, 이를 통해 산업제어시스템을 폐쇄망 구조로 운영하는 것만으로는 사이버공격으로부터 완전히 자유로울 수 없음이 드러났다. 이러한 원자력시설 대상 사이버공격에 대비하기 위해 국제원자력기구(IAEA)의 NSS 17과 미국 원자력규제위원회(NRC)의 RG 5.71 등이 발간되어 원자력시설의 주요 디지털자산에 사이버보안조치를 수행하도록 권고 및 규제하고 있다[1][2]. 국내에서는 규제기준 KINAC/RS-015에서 제시한 101개의 보안조치 항목을 통제하는 방식으로 대응하고 있다[3]. 만약 규제대상인 디지털자산 내에서도 원전의 노심 손상을 유발하여 심각한 영향을 가져올 수 있는 자산을 도출해 보다 강화된 보안조치를 적용한다면 원전의 보안성을 한층 높일 수 있을 것이다. 이에 본 연구에서는 원전의 노심 손상을 유발할 수 있는 디지털자산인 핵심디지털자산[4]에 대해 차등적으로 적용 가능한 강화된 보안조치를 찾고자 한다.

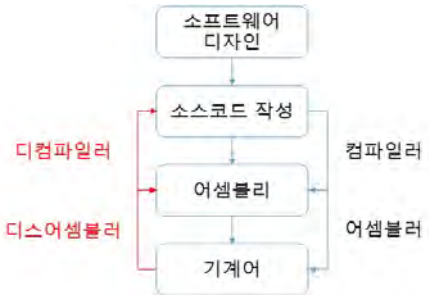
2. 핵심디지털자산 보안 요구사항 분석

핵심디지털자산을 위한 보안조치를 도출하기 위해 우선 원전의 디지털자산 중 사고를 직접적으로 일으키거나 사고완화 실패를 유발할 수 있는 제어시스템들의 하드웨어, 소프트웨어 및 네트워크 특성을 분석하여 이를 기반으로 핵심디지털자산의 공격 벡터를 식별하였다. 그리고 식별된 공격 벡터들을 예방하기 위해 사이버보안 기술, NIST 800-53, IAEA NSS 17 등의 국제 사이버보안 가이드라인 등을 분석하여 핵심디지털자산을 위한 보안 요구사항을 도출한 후 KINAC/RS-015에서 요구하는 보안 요구사항과 비교 및 분석한 결과, 핵심디지털자산을 위한 보안 요구사항이 KINAC/RS-015에서 요구하는 보안조치에 의해 대부분 만족되고 있음을 확인하였다[1][3][5]. KINAC/RS-015의 보안조치를 통해 완벽히 예방되지 않는 일부 보안 요구사항에 대해서는 기존 보안조치 중 일부를 강화하거나 추가적인 보안조치를 적용해 보안을 보다 강화하는 것이 필요하다. 이러한 추가 보안조치의 하나로 '코드 난독화(obfuscation) 적용'이 있으며, 동 보안조치는 공급망 통제, 신뢰성 확보 등의 보안조치가 포함되어 있는 KINAC/RS-015의 시스템 및 서비스 획득 관련 보안조치로 분류할 수 있다[3].

3. 역공학에 의한 제어시스템 위협

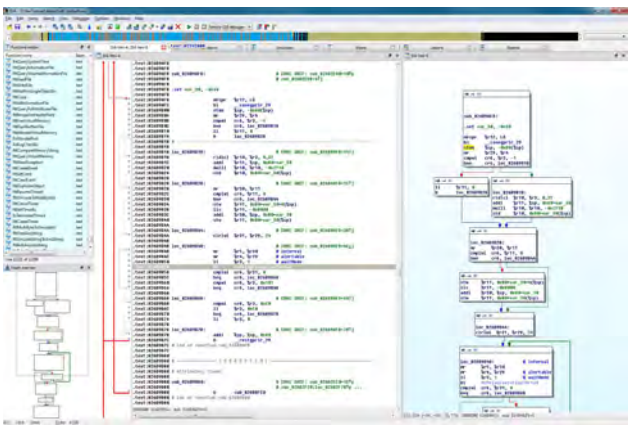
소프트웨어 공격 벡터 측면에서 분석한 결과, 소

프트웨어 역공학(reverse engineering) 또는 퍼징(fuzzing) 등을 통한 제어시스템 관련 신규 취약점 발견이 늘어나고 있다. 소프트웨어 소스코드를 어셈블리 파일로 변환하기 위해 컴파일 과정을 거치며, 어셈블리 파일을 최종적으로 바이너리 파일로 만들기 위해 어셈블 과정을 거치게 된다. 이러한 과정을 거쳐 생성된 바이너리 파일은 실행 파일의 구조를 이해하더라도 코드 섹션의 기계어를 분석하기 쉽지 않아 거의 블랙박스로 취급되고 있다. 그러나 많은 공격자들은 기계어로 생성된 바이너리 파일을 대상으로 디스어셈블러와 디컴파일러를 활용해 간소화된 소스코드를 추출하고 있으며, 이러한 소스코드를 바탕으로 코드들의 계층 구조 및 제어 흐름을 분석하고 있다. 이를 소프트웨어 역공학이라 하며, 아래 그림은 역공학을 통한 소스코드 추출 과정을 보여주고 있다[6].



(그림 1) 역공학을 통한 소스코드 추출 과정

악성코드 분석, 소프트웨어 보안성 테스트 등 다양한 목적을 위해 역공학이 사용되고 있으며, 이를 지원하기 위해 다양한 디스어셈블 및 디컴파일 도구들이 사용되고 있다. 대표적인 역공학 도구로는 IDA Pro, OllyDbg 등이 있으며, 아래는 IDA Pro를 이용하여 소프트웨어의 기계어 함수 구성 및 흐름을 분석하는 화면이다[7][8].



(그림 2) IDA Pro를 이용한 분석 예시 화면

산업제어시스템에서 사용되는 바이너리 파일 중 제조사에서 공개하고 있는 일부 파일들은 공격자가 역공학을 통해 소스코드를 추출하는데 사용되기도 한다. 원자력시설에서 사용되는 제어시스템의 경우에는 전용시스템으로 만들어지거나 보안 상의 문제로 일반적인 산업제어시스템 대비 외부 공개가 훨씬 제한적인 상황이다. 그럼에도 동일 제조사가 만든 제어시스템의 경우 유사하거나 동일한 라이브러리를 사용하는 경우가 있기 때문에 이를 공격자가 파악하여 분석하게 된다면 원자력시설의 제어시스템에 대한 사이버공격에 사용될 가능성도 있다.

4. 코드 난독화

앞서 설명한 바와 같이 공격자는 특정 시스템의 바이너리 파일을 확보한 후 역공학을 수행하여 어셈블리 코드나 소스코드를 추출하고 이를 분석해 해당 시스템의 취약점을 찾아낸다. 이러한 공격을 예방하기 위해서 코드 난독화를 적용할 수 있다. 코드 난독화는 프로그램 변환 기법의 일종으로 역공학을 사용해 소프트웨어의 행위를 분석하는 것을 어렵게 만들기 위해 프로그램의 원래 의미는 보존하면서 코드의 일부 혹은 전체를 변형하는 프로그램 보호 기법을 말한다. 프로그램의 원래 의미를 보존하기 위해서는 반드시 코드 변형 후에도 변형 전과 동작이 동일해야 한다[9].

코드 난독화는 난독화 대상에 따라 바이너리 난독화와 소스코드 난독화로 분류할 수 있다. 바이너리 난독화는 컴파일 후에 생성된 바이너리 자체에 대해 역공학을 통해 분석하기 어렵도록 변형하는 것이다. 대표적으로 예로 프로그램 언어 특성에 맞게 심볼 정보를 제거하거나 심볼 이름을 변경하는 방법을 통해 공격자가 바이너리를 이해하고 분석하는데 지장을 주는 것이 있다. 소스코드 난독화의 경우 프로그램의 소스코드를 알아보기 힘들게 변형하는 것으로서 특정 코드를 과도하게 복잡하게 만들거나 아무 것도 수행하지 않는 코드를 추가하는 방법, 서로 관련이 없는 다양한 함수들을 섞는 방법, 데이터를 알아보기 어렵게 변경하거나 변수명에 의미가 포함되지 않도록 일반화하는 방법, 주석의 정보를 제거하는 방법 등이 이에 해당한다[10][11]. 이러한 방식을 통해 소스코드를 난독화한 후 컴파일을 통해 바이너리를 만들게 되면 해당 바이너리를 분석하기가 어려워지며, 소스코드 자체가 유출되는 경우에도 분석에 많은 지장을 줄 수 있게 된다.

5. 코드 난독화 적용을 위한 고려사항

이러한 코드 난독화 기법을 핵심디지털자산의 소프트웨어에 적용하기 위해서는 다음의 요건들을 적용하는 것이 필요하다. 난독화는 공격자가 핵심디지털자산의 취약점을 분석하는데 보다 많은 시간과 비용을 소모시켜 공격을 어렵게 만들지만 침투 자체를 완벽히 불가능하게 할 수는 없으며, 난독화를 높은 수준으로 적용할수록 시스템의 성능을 저하시키고 비용이 증가하게 된다. 따라서 난독화에 투입되는 비용과 효과를 분석하여 난독화의 범위와 수준을 결정하는 것이 필요하다. 또한, 난독화로 인해 변형된 코드는 변형 전과 동일하게 동작해야 하며[6], 높은 수준의 난독화는 시스템에 부하를 발생시켜 성능을 저하시킬 수 있으므로 난독화가 적용된 후에도 시스템에 요구되는 최소 성능을 보장할 수 있어야 한다. 디버깅 및 유지보수에 영향을 주지 않기 위해서 심볼, 주석 등의 정보가 제거되더라도 추적 가능하도록 해당 정보들을 별도로 보관하는 등의 추가적인 조치가 필요하다. 이러한 요건들을 통해 난독화로 인한 부작용을 방지할 수 있을 것이다.

핵심디지털자산의 소프트웨어에 코드 난독화 기법을 적용한다면 공격자들이 핵심디지털자산의 취약점을 분석하는데 보다 많은 시간과 비용이 소모될 것이며 이를 통해 사이버공격을 예방하고 완화시키는데 도움을 줄 수 있을 것으로 기대한다. 다만, 프로그램 업데이트가 자주 일어나지 않는 환경에서는 효과가 저하될 수 있기 때문에 현재의 원자력시설 환경에서 코드 난독화 만으로는 취약점 분석을 통한 사이버공격을 원천적으로 막기 어려울 수도 있다는 점도 같이 고려되어야 한다. 또한, 코드 난독화를 적용하는 보안조치들은 대부분 설계단계에서만 적용이 가능하며 시스템의 성능과 이에 따른 안전 요건에 영향을 줄 수도 있기 때문에, 신규 원자력시설 건설 혹은 기존 시설의 설비 개선 시 시스템 설계 단계에서부터 보안요건과 안전요건을 모두 고려하여 진행되어야 할 것이다.

5. 결론

본 연구에서는 원전 사고와 직접적으로 관련된 핵심디지털자산에 대해 기존의 KINAC/RS-015 대비 추가적으로 적용이 필요한 보안 요구사항 중 하나로 코드 난독화를 제시하였다. 이러한 코드 난독화와 몇 가지 추가적인 보안조치를 핵심디지털자산에 적용한다면 원자력시설의 보안성을 보다 강화할 수 있을 것으로 기대된다.

ACKNOWLEDGMENT

본 연구는 원자력안전위원회의 재원으로 한국원자력안전재단의 지원을 받아 수행한 원자력안전연구사업의 연구결과입니다. (No. 1605007)

참고문헌

[1] IAEA, "Computer Security at Nuclear Facilities," Nuclear Security Series(NSS) 17, 2011.

[2] U.S.NRC, “Cyber Security Program for Nuclear Facilities,” Regulatory Guide 5.71, 2010.

[3] 한국원자력통제기술원, "원자력시설의 컴퓨터 및 정보시스템 보안," KINAC/RS-015, 2016.

[4] Kookheui Kwon, “Research on Vital Assets for Nuclear Cyber Security,” ANS 2017 International Topical Meeting on Probabilistic Safety Assessment and Analysis, 2017.

[5] NIST, "Security and Privacy Controls for Federal Information Systems and Organizations," NIST 800-53 Revision 4, 2013.

[6] C. Cifuentes, “An environment for the reverse engineering of executable programs,” 2nd Asia-Pacific Software Engineering Conference(APSEC), 1995.

[7] IDA, https://www.hex-rays.com/products/ida

[8] OllyDbg, http://www.ollydbg.de/

[9] C. Collberg, C. Thomborson, and D. Low, “A Taxonomy of Obfuscating Transformations,” Technical Report #148, The University of Auckland, 1997.

[10] J. Viega, “Building Secure Software: How to Avoid Security Problems the Right Way,” Addison-Wesley, 2011.

[11] S. Banescu, C. Collberg, V. Ganesh, Z. Newsham, and A. Pretschner, “Code Obfuscation Against Symbolic Execution Attacks,” 32nd Annual Conference on Computer Security Applications, ACM, 2016.