정적 MILP와 Slack을 활용한 임베디드 시스템 동적 보호 방안 연구

박건우[1], 이하임[1], 김정연[2], 유진호[2], 서대희[3]
[1]상명대학교 컴퓨터과학과 석사과정
[2]상명대학교 경영학부 교수
[3]상명대학교 지능데이터융합학부 교수
202532012@sangmyung.kr, 202532018@sangmyung.kr, jykim@smu.ac.kr,
jhyoo@smu.ac.kr, daehseo@smu.ac.kr

A Study on the Dynamic Protection of Embedded Systems Based on Static MILP and Slack

KunWoo Park[1], HaIm Lee[1], JeongYeon Kim[2], JinHo Yoo[2], DaeHee Seo[3]
[1]Dept. of Computer Science, Sangmyung University
[2]Dept. of Business Administration, Sangmyung University
[3]Dept. of Intelligent Data Convergence, Sangmyung University

요 약

최근 엣지 클라우드 컴퓨팅 기반의 임베디드 시스템은 실시간성을 요구한다. 실시간성을 만족하지 못할 경우에는 공격자에 의해 보안 위협이 발생할 수 있으며 이로 인해 성능 저하 또는 제한시간 만료 문제가 발생할 수 있다. 이러한 문제들은 자율주행 차량, 항공기 제어같은 실시간성이 필요한 분야에서 위험을 초래할 수 있다. 기존 연구에서는 MILP(Mixed-Integer Linear Programming, 혼합 정수 선형 계획)를 기반으로 보안성과 스케줄 제약을 동시에 고려하는 정적 보호 기법을 제안하였다. 하지만 정적 보호 기법은 실행 중 발생하는 Slack(여유 시간)을 고려하지 않아 실행 이후 남은 시간이 존재하는 경우에도 새로운 공격에 대한 대응이나 추가적인 모니터링을 수행할 수 없다. 이는 곧 유연성 부족으로 인한 보안 위협으로 이어질 수도 있다.

본 연구는 이와 같은 기존 정적 보호 기법의 한계를 보완하기 위해 Slack을 활용한 동적 보호 방안을 제안한다. 제안 방식은 Task를 실행하는 과정에서 발생되는 Slack을 활용해서 성능 저하를 최소화하고 기존 설정된 위협들 뿐만 아니라 알려지지 않은 공격(Unknown Attack)까지 대응할 수 있도록 설계하여 실시간 시스템의 보안성과 안전성을 동시에 획득하는 것을 목표로 한다. 결과적으로 제안 방식을 통해 임베디드 시스템 환경에서 최적화된 방어 체계를 제공한다.

1. 서론

최근 빠르게 확산되고 있는 엣지 클라우드 컴퓨팅 기반의 임베디드 시스템(자율주행 차량, 항공기 제어)은 시간 지연이 시스템상 문제를 발생시킬 수 있는 실시간 환경에서 동작한다. 실시간성을 보장하지 못할 경우, 공격자는 연산 지연 등을 악용하여 시스템을 무력화할 수 있고 나아가 직·간접적인 재산·생명 피해로 이어질 수 있다. 따라서 임베디드 시스템 내에서 보안성과 실시간성을 동시에 확보하는 것은 필수적이다.

이러한 조건을 만족하기 위해서 기존 연구는 MILP(Mixed-Integer Linear Programming, 혼합 정수 선형 계획)을 기반으로 보안성과 실시간성을 동시에 고려하는 정적 보호 기법을 제안하였다. 하지만 정적 보호 기법은 실행 환경의 변화에 즉각적으로 대응하지 못하고 실제 실행 중 발생하는 Slack(여유 시간)을 활용하지 못하는 한계가 있다. 이로 인해 임베디드 시스템은 새로운 공격이나 예상치 못한 연산 증가에 의해 보안 문제가 발생할 수 있다.

본 연구는 이러한 기존 연구의 한계를 보완하기 위해 Slack을 활용한 동적 보호 방안을 제안한다. 제안 방식은 Slack을 통해 사전에 미리 알려진 공격 뿐만 아니라 알려지지 않은 공격에도 대응할 수 있도록 설계하여 보안성과 안전성을 동시에 제공한다.

본 논문의 2장에서는 기술연구에 대해 분석하고 배경 이론에 대해 설명한다. 3장은 정적 MILP와 Slack을 활용한 동적 보호 연구 방안을 제안한다. 마지막으로 4장은 연구 요약 및 향후 연구 방향을 제시한다.

2. 정적 보호 기법

본 장에서는 기존 정적 보호 기법 연구 중 MILP 기반 정적 보호 방식 및 현재 정적 기법의 한계와 연구 동향에 대해 분석한다[1].

2.1 MILP 기반 정적 보호 방식 연구

임베디드 시스템의 실시간성을 위해 MILP 기반 정적 보호 방식이 연구되고 있다[2]. 정적 보호 기법은 실시간 소프트웨어 보안성을 강화하기 위해 Task의 CFG(Control Flow Graph, 제어 흐름 그래프)를

기반으로 CFG 내의 각 BB(Basic Block, 기본 블록)에 취약점에 대한 보안 기법을 배치하는 방식이다. 즉, 시스템 흐름을 그래프 형식으로 표현해서 어떤 Task에서 공격이 발생할 수 있는지 파악하고, 해당 Task에 방어 기법을 적용하는 식이다. 이때 취약점에 대한 적용 여부는 MILP를 활용하여 수식으로 모델링하고 동시에 보안 적용 효율성과 실행 시간의 오버헤드를 고려한다. 이를 통해 전체 Task 집합이 데드라인(특정 작업을 완료해야 하는 최종 시점)을 만족함과 동시에 보안 수준을 최대화하는 보안 기법 배치안을 도출한다.

정적 보호 방식에서 보안 취약점 점수의 심각도를 SSS(Security Susceptibility Score, 보안 민감도 점수)로 정의하고 동시에 정적 보호 기법의 보안 적용 효율성의 척도로 사용한다. 정적 보호를 적용할 경우 발생하는 오버헤드는 각 Task별로 WCET(Worst-Case Execution Time, 최악의 실행시간)에 반영하고, 이를 통해 Task의 응답시간이 데드라인을 만족하는지 분석한다. 결과적으로 MILP 기반 정적 최적화는 보안성과 스케줄링 간의 균형을 설계 단계에서 보장할 수 있도록 한다.

2.2 정적 기법의 한계와 연구 동향

MILP 기반의 정적 보호 기법은 실행 시점이 아닌 설계 시점에서 고정된 보호 구성을 설계하기 때문에 실행중 발생하는 동적 변화나 Slack을 활용하기 어렵다. 즉 보호 기법이 미리 결정되어 있어 실행 상황에 따른 유연성이 부족하고 새로운 위협이나 알려지지 않은 공격에 대응하는데 한계를 가지고 있다. 또한 모든 보호 기법을 적용하게 된다면 Task의 실행 시간이 매우 크게 증가하여 데드라인을 넘길 가능성이 있다.

최근 연구들은 이러한 기존 정적 보호 기법의 한계를 보완하기 위해 MILP 최적화와 RTA(Response Time Analysis, 응답시간 분석)를 결합하여 Task의 실행 시점에 보안 적용 범위를 조정하거나 CWE(Common Weakness Enumeration, SW 취약점 목록) 와 같은 취약점 분류 표준화를 활용해서 보안 적용의 우선순위를 정한다. 이러한 연구들은 정적 보호 기법을 기반으로 하되 보안성과 실시간성을 확보하는 발전 방향성을 보인다[3].

따라서 본 연구에서는 정적 보호와 실행 시점의 유연하게 보호 기법을 적용하기 위한 Slack 기반 동적 보호 방식(Slack-Based Dynamic Protection)을 제안한다.

3. 제안 방식

본 연구는 이러한 정적 보호 기법을 유지하면서, Task의 응답 시간 검사점 집합에서 도출되는 여유 시간을 Slack으로 정의하고, 런타임에서는 Slack이 허용하는 범위 내에서만 추가 보호를 활성화하는 Slack 기반 동적 보호 기법을 제안한다. 또한 본 연구는 알려지지 않은 공격에 대한 보안 기법의 기여도와 공격 전술 집합에서 최소한의 방어 비율을 함께 고려한다 [4],[5].

Slack 기반 동적 보호 수식에서 활용되는 변수는 다음 표 1과 같이 정리한다.

3.1 Slack 기반 동적 보호 조건

<표 1> Slack 기반 동적 보호 활용변수 모음

Symbol	Description
τ_i	i번째 Task
T_i	Task τ_i의 주기 (period)
D_i	Task τ_i의 데드라인
$hp(\tau_i)$	τ_i보다 우선순위가 높은 Task 집합
$b_{i,j}$	Task τ_i의 j번째 BB
$C_{i,j}$	보호 미적용 시 $b_{i,j}$ 실행시간
V	보호 유형 집합 (set of protection/vulnerability types)
$X_{i,j}^{v}$	기본블록 $b_{i,j}$에 보호 유형 v 적용 여부 (1/0)
$n_{i,j}^{v}$	기본블록 $b_{i,j}$에서 보호 v 호출/접근 횟수
σ_v	보호 유형 v의 단위 오버헤드
$\lambda \in paths(\tau_i)$	τ_i의 실행 경로
W_i	보호가 반영된 τ_i의 WCET
Y_i	τ_i의 응답시간 검사점 집합
$y_{i,g}$	검사점 g의 시각 (checkpoint)
$E_{i,g}$	검사점 $y_{i,g}$ 선택 여부 (1/0)
$RT_{i,g}$	검사점 $y_{i,g}$에서 계산된 응답시간
RT_i	Task τ_i의 전체 응답시간
M	Big-M 상수
$slack_i$	선택된 검사점에서의 최소 여유시간
y_i	보호 단위 i의 설치 여부 (1/0)
z_i	보호 단위 i의 기여도 반영 여부 ($z_i \le y_i$)
w_i	알려진 공격 방어 기여도 점수
q_i	알려지지 않은 공격 탐지 기여도
o_i	보호 비용 (overhead, resource, operational cost)
γ	미지 공격 기여 가중치
β	Slack 보상 가중치
Γ	전체 미지 공격 최소 기여 임계값
$a_{i,u}$	보호 단위 i가 전술 u를 방어하면 1
ρ_u	전술 u의 최소 커버 비율
u	전술 인덱스 (예: MITRE ATT&CK)

Task τ_i는 주기 T_i, 데드라인 $D_i \le T_i$, 기준선 WCET W_i을 가진다. $hp(\tau_i)$는 τ_i보다 우선순위가 높은 Task 집합이다. 그리고 Task 내의 BB의 요소인 $b_{i,j}$에서 보호 유형 u를 적용할 때 발생할 수 있

는 최악 오버헤드는 $n_{i,j}^{v} \cdot \sigma_v$이고, 해당 보호가 기여하는 보안 기여도는 w_i (알려진 공격), q_i (알려지지 않은 공격)으로 모델링한다. 마지막으로 응답시간 해석은 정적 MILP 논문에서 제시된 검사점 집합 Y_i 기반 충분조건을 사용한다[1].

3.2 Slack 기반 동적 보호 방법

Slack 기반 동적 보호는 정적 MILP의 제약을 유지하면서, 무보호 또는 최소 보호 구성을 우선 고려하여 Task별 기준선 WCET(Worst-Case Execution Time, 최악의 실행시간)를 산출한다. 본 연구는 이를 기반으로 다음과 같은 과정을 수행한다.

① 처음에는 Task 별로 무보호 또는 최소 보호 적용을 고려하여 WCET를 산출한다.

② WCET 산출 이후 각 Task의 실행 경로를 기준으로 WCET 상한을 결정한다.

③ 보호 선택 변수와 관련 변수들의 범위를 정의한다.

④ Task 우선순위 기반 스케줄링에서 상위 우선순위 작업의 간섭을 반영하여 응답시간 하한을 산출한다.

⑤ 응답시간 하한과 Big-M 기법을 통해 검사점을 선택하여 선택 여부에 따라 응답시간 제약을 활성화하거나 완화한다.

⑥ 이후 각 작업의 응답시간이 데드라인을 넘지 않도록 범위를 정의한다.

⑦ 검사점에서 데드라인까지 남은 시간에서 실행시간과 간섭을 제외하고 Slack을 구한다.

⑧ Slack을 구한 후 알려지지 않은 공격에 대한 최소 방어 기여도를 구한다.

⑨ 이때 특정 보호 적용이 방치되지 않도록 MITRE ATT&CK 위협 모델의 전술 별 최소 방어 기여도를 만족할 수 있도록 한다.

⑩ 알려진 공격 방어 효율성, 알려지지 않은 공격 방어 기여도, 보호 비용, Slack을 종합적으로 고려하여 실시간성을 유지하면서 보안성을 최대화한다.

다음은 Slack 기반 동적 보호 방법에서 정의한 수행과정의 세부 내용을 설명한다.

① 먼저 각 작업의 WCET는 모든 실행 경로에 대해 보호 기법의 오버헤드를 포함하여 식 (1)과 같이 계산한다.

$$W_i \geq \sum_{b_{i,j} \in \lambda} \left(C_{i,j} + \sum_{v \in \mathrm{V}} X_{i,j}^{v} \cdot n_{i,j}^{v} \cdot \sigma_v \right) \tag{1}$$

② 수식 (1)의 경로 기반 WCET 상계는 보호 선택 변수 $X_{i,j}^{v}$, 호출 횟수 $n_{i,j}^{v}$, 단가 σ_v 를 누적하여 경로별 총 실행시간을 계산하고, 그 최대값이 Task의 WCET가 되도록 상한을 정한다.

$$\forall \lambda \in paths(\tau_i) \tag{2}$$

③ 우선순위 기반 스케줄링에서 응답시간은 상위 우선순위를 가지고 있는 Task의 간섭을 포함하여 하한이 주어진다.

$$RT_{i,g} \geq W_i + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{y_{i,g}}{T_j} \right\rceil W_j, \ \forall g \tag{3}$$

④, ⑤ Task의 응답시간이 실행에 필요한 최소 시간 이상으로 계산하도록 강제하는 조건은 검사점 $y_{i,g}$ 까지 보호가 적용된 Task 자체 실행시간 W_i와 상위 우선순위 Task들의 간섭을 합산하여, 해당 검사점에서의 응답시간 $RT_{i,g}$가 이 값 이상이 되도록 보장한다. 이후 MILP에서 유효한 검사점 $y_{i,g}$를 선택하기 위해 Big-M 제약을 수식에 적용한다.

$$RT_{i,g} \leq y_{i,g} + (1 - E_{i,g})M \tag{4}$$

$$\sum_{g} E_{i,g} = 1, \ RT_i \geq RT_{i,g} - (1 - E_{i,g})M \tag{5}$$

⑥ Big-M 상계 제약은 검사점 선택 변수 $E_{i,g}$가 0인 경우에는 M을 이용해 특정 상황에서는 제약을 적용하지 않아도 되도록 설계해서 응답시간의 일관성을 보장하고, 선택된 경우에는 응답시간이 검사점 값에 의해 긴밀하게 제한되도록 한다. 단일 검사점 선택 제약은 검사점 선택 변수들의 합을 1로 제한하여, 각 Task에 대해 정확히 하나의 검사점만 선택되도록 강제한다. 이때, 응답시간은 데드라인을 넘길 수 없다.

$$RT_i \leq D_i \tag{6}$$

⑦ Slack 기반 동적 보호에서 Slack은 각 검사점에서 데드라인까지 남은 시간에서 실행시간과 상위 우선순위를 가지고 있는 Task의 간섭을 차감한 값으로 정의되며, 선택된 점에서 최소 Slack을 갖는다.

$$slack_i \leq y_{i,g} - \left(W_i + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{y_{i,g}}{T_j} \right\rceil W_j \right) + (1 - E_{i,g})M \tag{7}$$

⑧ 본 연구는 알려지지 않은 공격에 대한 방어 기여도와 전술별 최소 커버리지를 함께 고려한다. 알려지지 않은 공격 하한 제약은 알려지지 않은 공격에 대한 탐지 기여도의 총합이 임계값 Γ 이상이 되도록 설정하여 알려지지 않은 공격에 대한 최소 대응 성능을 확보한다.

$$\sum_i q_i z_i \geq \Gamma \tag{8}$$

$$\frac{\sum_i a_{i,u} \cdot z_i}{\sum_i a_{i,u}} \geq \rho_u, \forall u \tag{9}$$

⑨ 커버리지 하한 제약은 전술 u별로 방어 기여도 $a_{i,u}$가 일정 비율 ρ_u 이상 반영되도록 하여, 특정 전술이 방치되지 않고 최소한의 커버리지를 유지하도록 한다. 마지막으로 보안 기여도, 알려지지 않은 공격 탐지 기여, 비용, Slack 안정성을 종합하여 최대화를 진행한다.

$$\max \sum_i w_i \cdot z_i + \gamma \sum_i q_i \cdot z_i - \sum_i o_i \cdot y_i + \beta \sum_i slack_i \tag{10}$$

⑩ 목적함수는 알려진 공격 방어 기여도 $w_i \cdot z_i$, 알려지지 않은 공격 탐지 기여 $\gamma \cdot q_i \cdot z_i$, 보호 적용 비용 $-o_i \cdot y_i$, 그리고 Slack 확보에 따른 안정성 보상 $\beta \cdot slack_i$를 동시에 고려하여, 보안성과 효율성 및 실시간성의 균형을 이루도록 설계한다.

3.4 Slack에 따른 대응방안

Slack은 Task와 WCET에 따라 Slack infeasible, Slack feasible, Slack feasible with margin 총 3가지 경우로 나눌 수 있다. 각 경우에 대한 설명은 다음과 같다.

- **Slack infeasible (Slack 불가능)** : 보안 수준 적용을 제외하더라도 특정 검사점 y를 초과할 경우를 나타낸다. 이 경우, 동적 보호 뿐만 아니라 스케줄링 자체가 불가능하므로 W_i조정이 필요하다.
- **Slack feasible (Slack 0 상태)** : Slack이 없기 때문에 스케줄링은 가능하나, 동적 보호를 할 수 없는 상황을 의미한다. 이렇게 될 경우에는 동적 보호를 제외한 기본 Task만 실행한다.
- **Slack feasible with margin (Slack 여유 상태)** : 모든 검사점에 시간이 남는 것을 의미한다. 매 Task 마다 여유 시간 내에서 동적 보호를 다룰 수 있다.

4. 결론

본 연구는 엣지 클라우드 기반 임베디드 시스템의 기존 정적 보호 기법이 가지는 한계를 보완하기 위해 Slack 기반 동적 보호 방식을 제안한다. 제안 방식은 WCET와 응답시간 제약을 MILP로 정적 보장 후 실행 시간 중 Slack을 활용해서 추가적으로 보호 기법을 적용할 수 있도록 제안한다. 제안 방식은 오버헤드를 제어하면서 데드라인도 준수할 수 있도록 한다. 그리고 알려진 보안 위협 뿐만 아니라 알려지지 않은 공격에 대한 최소 대응 범위와 전술 단위 커버리지를 함께 고려해서 실시간 시스템의 안전성과 보안성을 동시에 확보할 수 있도록 하였다.

향후 연구에서는 제안 방식을 실제 임베디드 시스템에 적용하여 성능과 보안 효율성을 검증할 계획이다. 제안 방식을 통해 항공기 제어, 산업 제어 시스템 등의 임베디드 시스템 분야에서 유용한 보안 강화 방식으로 기여할 수 있을 것으로 기대된다.

사사표기

이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원-대학ICT연구센터(ITRC)의 지원을 받아 수행된 연구임(IITP-2025-RS-2024-00438056)

참고문헌

[1] S. Di Leonardi, F. Aromolo, P. Fara, G. Serra, D. Casini, A. Biondi, and G. Buttazzo, Maximizing the Security Level of Real-Time Software While Preserving Temporal Constraints, IEEE Access, vol. 11, pp. 35591 - 35607, 2023.

[2] Y. Wang, A. Li, J. Wang, S. Baruah, and N. Zhang, Opportunistic Data Flow Integrity for Real-Time Cyber-Physical Systems Using Worst Case Execution Time Reservation, in Proceedings of the USENIX Security Symposium, Philadelphia, USA, 2024, pp. 1 - 16.

[3] R. Pellizzoni, et al., DeepTrust RT: Time-aware Scheduling for Secure DNN Inference in Real-Time Systems, in Proceedings of the 36th Euromicro Conference on Real-Time Systems (ECRTS), Lille, France, 2024, pp. 1 - 14.

[4] Geetishree Mishra, An Efficient Hybrid Scheduler Using Dynamic Slack for Real-Time Critical Task Scheduling in Multicore Automotive ECUs, International Journal of Embedded Systems and Applications (IJESA), vol. 5, no. 2, pp. 01-08, 2024.

[5] Daniel Casini, To MILP or not to MILP? On AI techniques for the design and optimization of real-time systems, Real-Time Systems, vol. 61, no. 2, pp. 294 - 299, 2025.