정보보호 R&D 데이터 챌린지

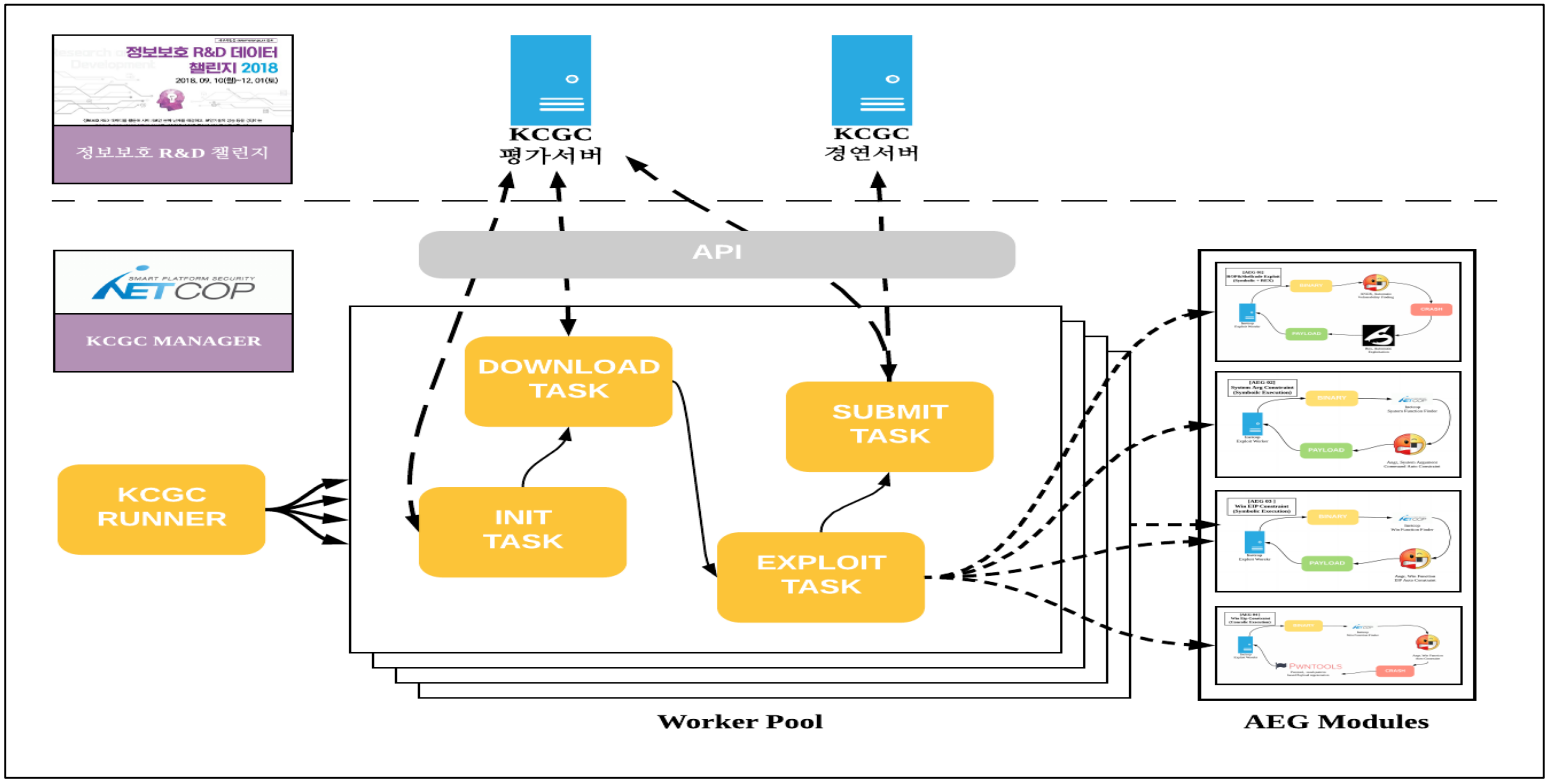
취약점 자동탐지 트랙

2018.12.04. (주)아이넷캅

목차

1. CRS 시스템 전체 구성도
    1. AEG 01 – Symbolic + Rex (ROP, Call Shellcode Technique)
    2. AEG 02 – Symbolic (System Argument Constraint)
    3. AEG 03 – Symbolic (Win EIP Constraint)
    4. AEG 04(+ 05,06,07) – Concolic (Win EIP Constraint)
2. 예선 / 본선 취약점 탐지 결과
3. Q&A

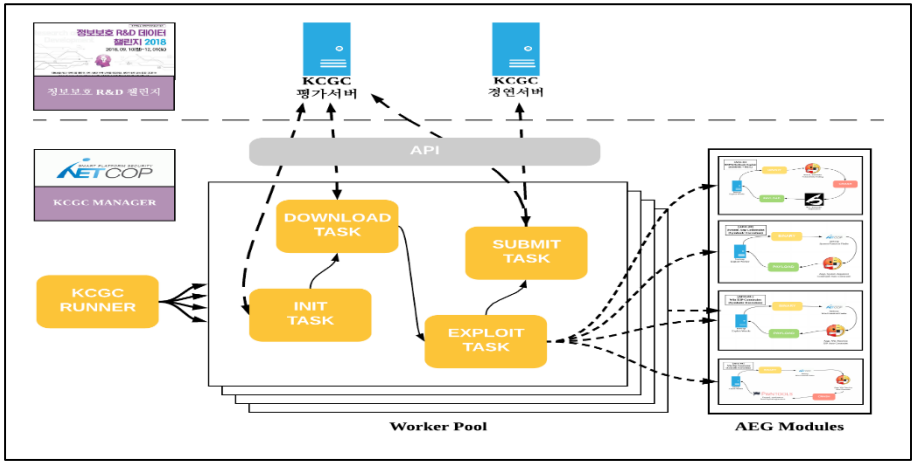
CRS 시스템 전체 구성도



[그림1] 아이넷캅 CRS 시스템 전체 구성도

CRS 시스템 전체 구성도

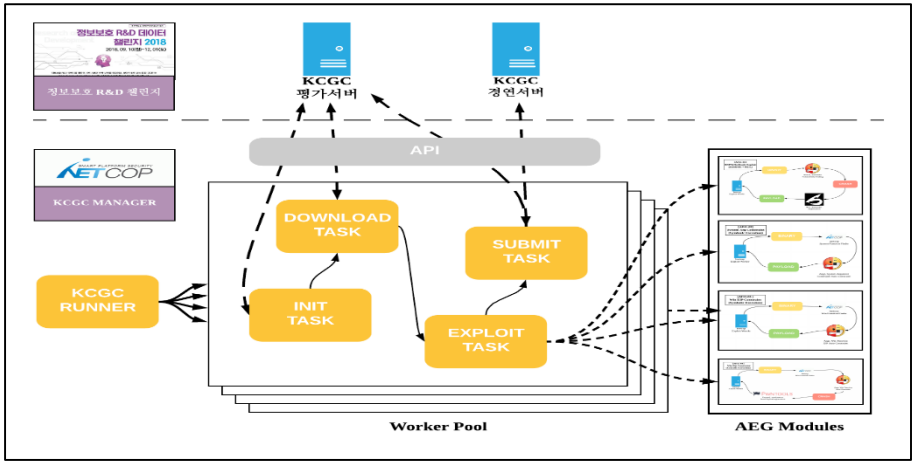
1. KCGC MANAGER (CRS 통합 시스템 프로젝트)
   1. Runner Module
      - Worker Module 프로세스 실행 역할
      - N개의 각 worker의 생명주기를 관리
      - Worker의 개수에 대해 Scalable한 구조
   2. Worker Module
   3. AEG Modules
   4. API Module



[그림1] 아이넷캅 CRS 시스템 전체 구성도

CRS 시스템 전체 구성도

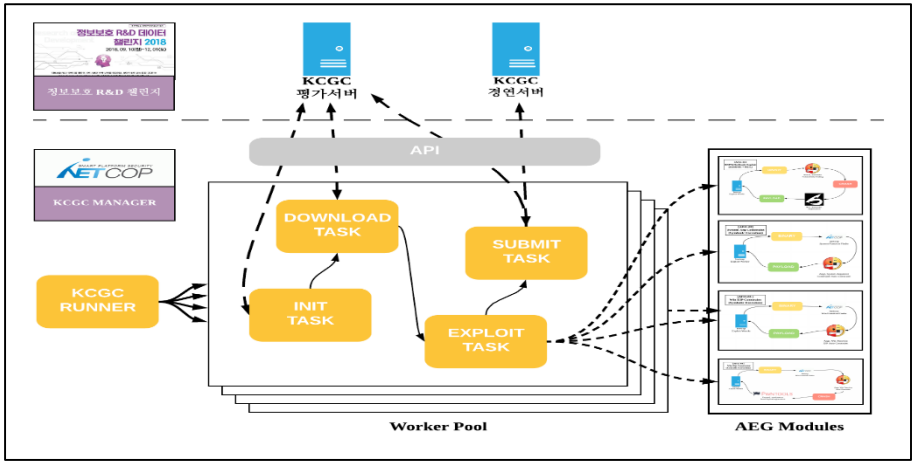
1. KCGC MANAGER (CRS 통합 시스템 프로젝트)
   1. Runner Module
   2. Worker Module
      - 4가지 독립적인 Task로 구성
        - Init Task (대회 진행 여부 파악)
        - Download Task (바이너리 다운로드)
        - Exploit Task (AEG 모듈 실행 및 payload 작성)
        - Submit Task (remote exploit 및 flag 전송)
      - Worker 프로세스의 개수에 따라 성능 조절
   3. AEG Modules
   4. API Module



[그림1] 아이넷캅 CRS 시스템 전체 구성도

CRS 시스템 전체 구성도

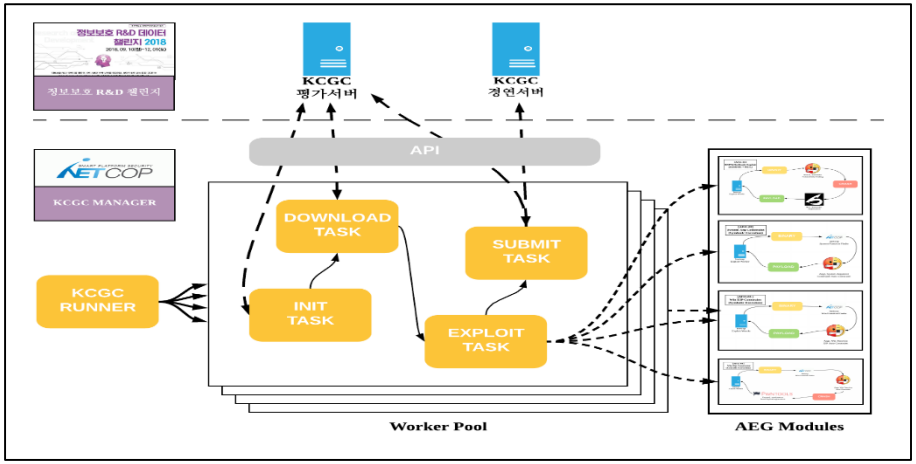
1. KCGC MANAGER (CRS 통합 시스템 프로젝트)
   1. Runner Module
   2. Worker Module
   3. AEG Modules
      - AEG 01 – Symbolic + Rex
        (ROP, Call Shellcode Technique)
      - AEG 02 – Symbolic
        (System Argument Constraint)
      - AEG 03 – Symbolic
        (Win EIP Constraint)
      - AEG 04 – Concolic (+05,06,07)
        (Win EIP Constraint)
   4. API Module



[그림1] 아이넷캅 CRS 시스템 전체 구성도

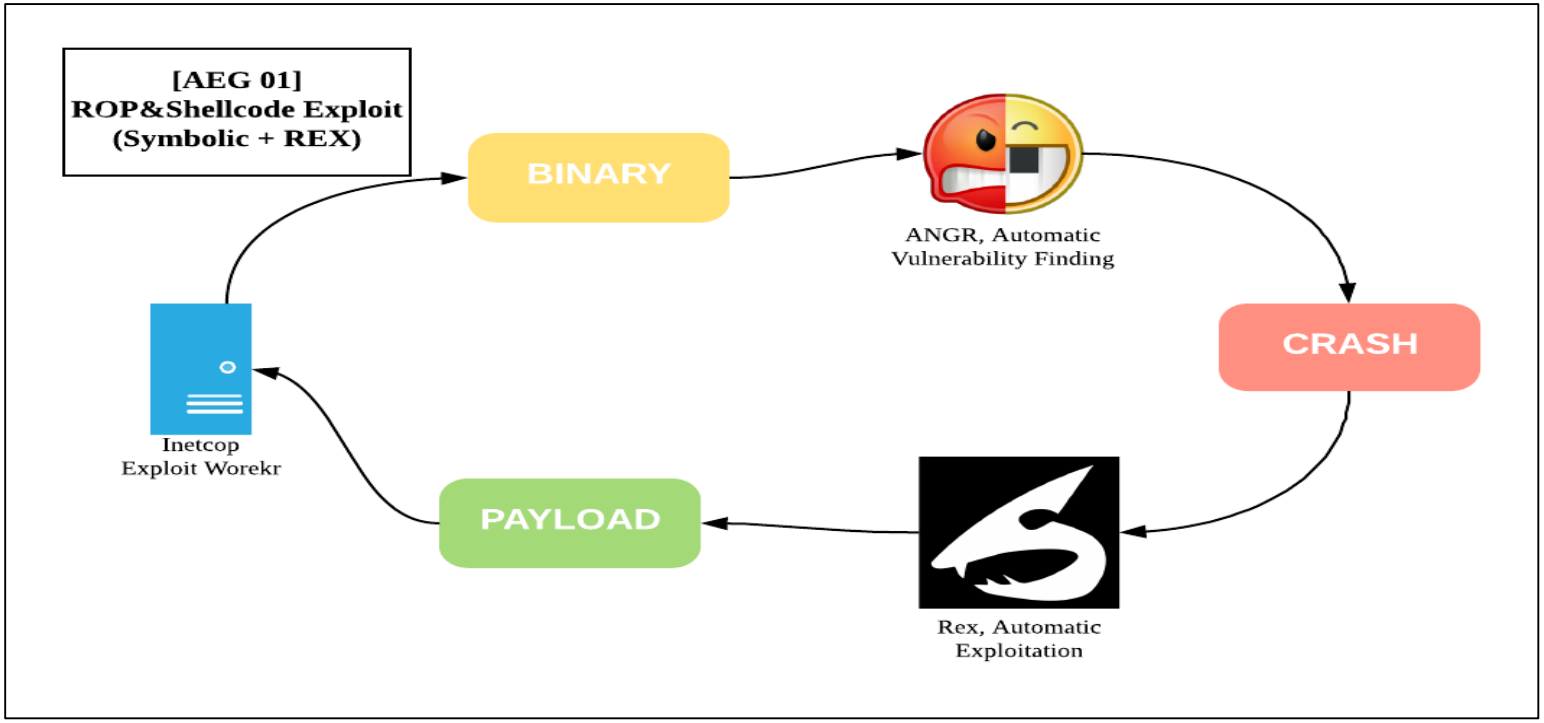
CRS 시스템 전체 구성도

1. KCGC MANAGER (CRS 통합 시스템 프로젝트)
    1. Runner Module
    2. Worker Module
    3. AEG Modules
    4. API Module
        - 평가서버와 통신 API 추상화
        - 4가지 메소드로 구성
            - get_list_server
            - get_list_cb
            - get_cb_file
            - submit_key
        - 내부 테스트시 LOCAL_API로 변경하여 전체 과정 테스트



[그림1] 아이넷캅 CRS 시스템 전체 구성도

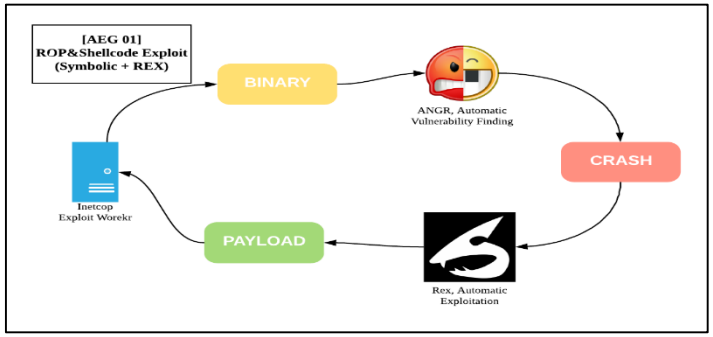
AEG 01 –
Symbolic + Rex (ROP, Call Shellcode Technique)



[그림2] ROP & Shellcode Exploit (Symbolic Execution + Rex) 구성도

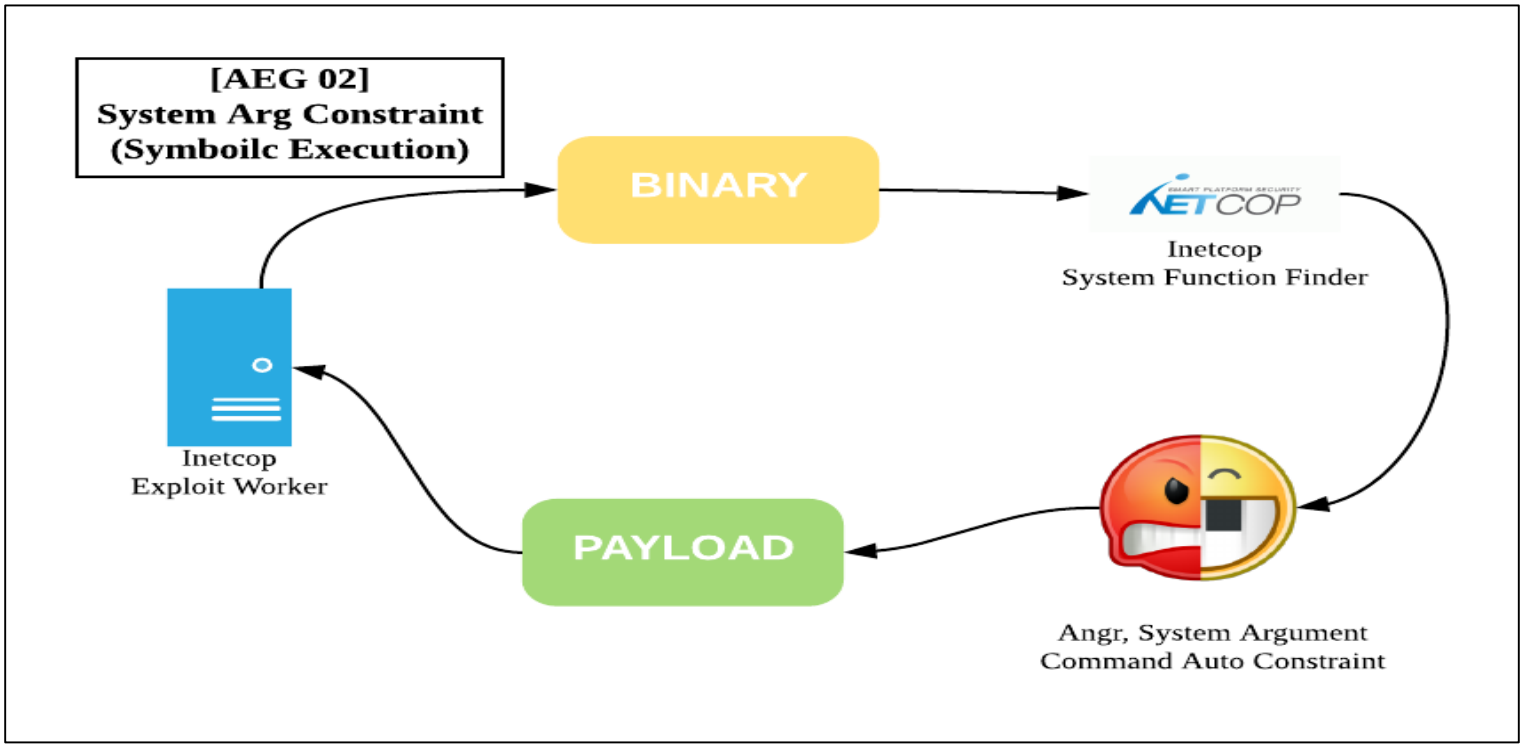
AEG 01 – Symbolic + Rex (ROP, Call Shellcode Technique)

- ANGR을 사용해 Binary에 존재하는 Crash를 탐색
- Rex로 Payload를 생성
- ROP, Call Shellcode 두 가지 technique을 사용



[그림2] ROP & Shellcode Exploit (Symbolic Execution + Rex) 구성도

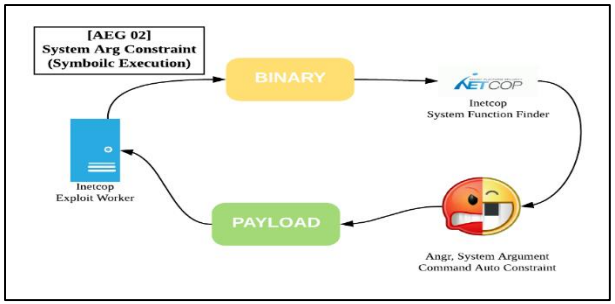
AEG 02 – Symbolic (System Argument Constraint)



[그림3] System Argument Constraint (Symbolic Execution) 구성도

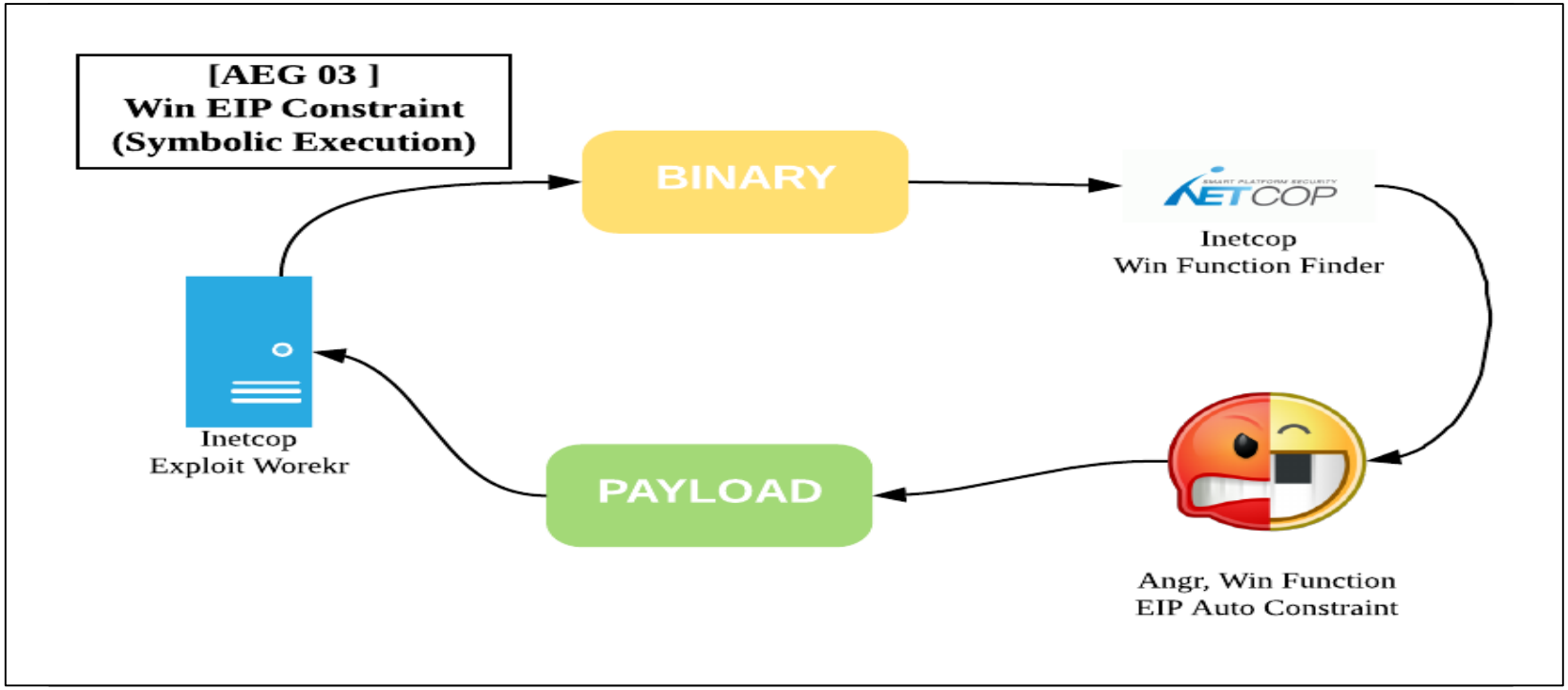
AEG 02 – Symbolic (System Argument Constraint)

- System 함수를 탐색, 쉘을 획득하기 위한 인자 값을 생성
- Function Finder를 개발해 System 함수의 위치를 추출
- Angr Symbolic, System 인자로 입력되는 인풋을 생성



[그림3] System Argument Constraint (Symbolic Execution) 구성도

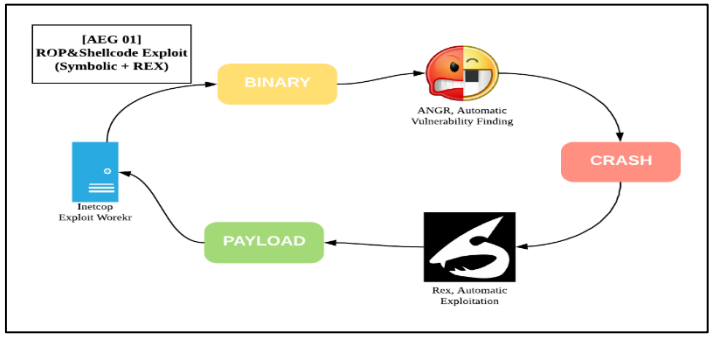
AEG 03 – Symbolic (Win EIP Constraint)



[그림4] Win EIP Constraint (Symbolic Execution) 구성도

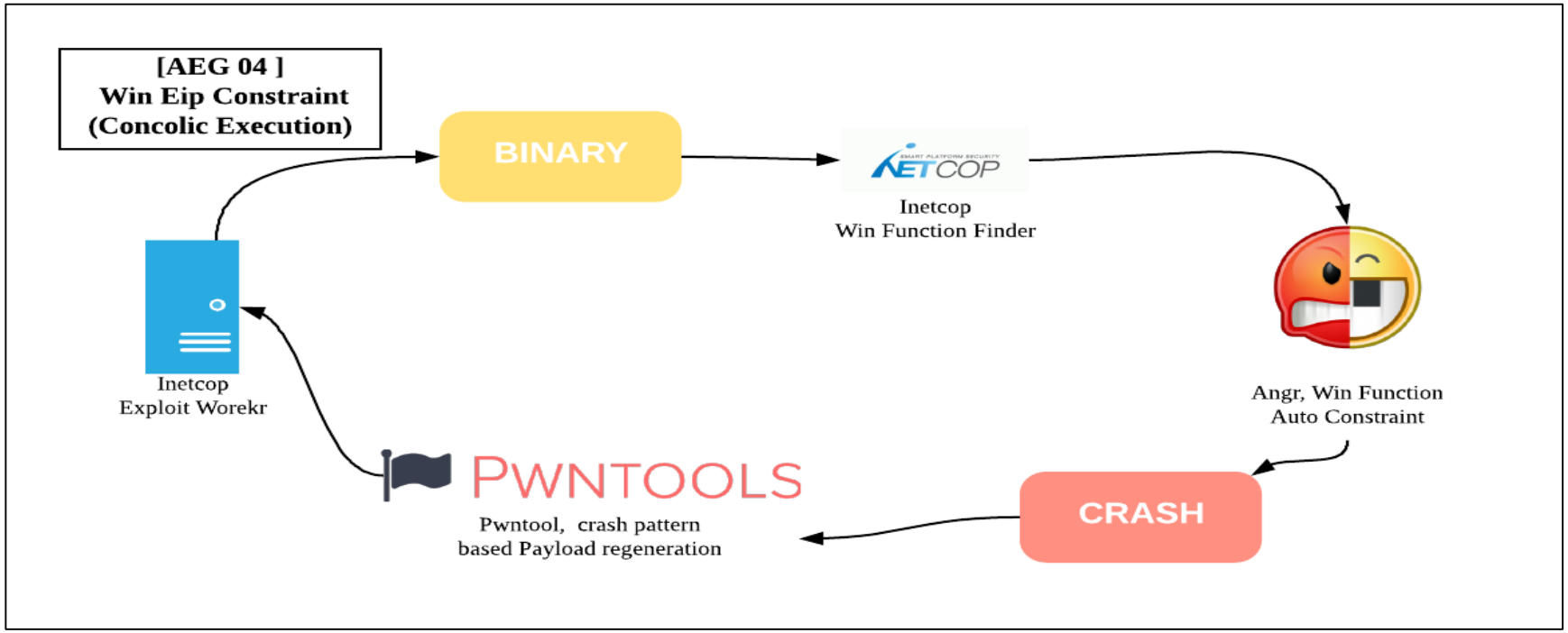
AEG 03 – Symbolic (Win EIP Constraint)

- Win Function 탐색, Win Function으로 EIP를 조작
- Function Finder 개발, Win Function의 위치를 추출
- Angr Symbolic, EIP를 Win Function 변경 인풋을 생성



[그림2] ROP & Shellcode Exploit (Symbolic Execution + Rex) 구성도

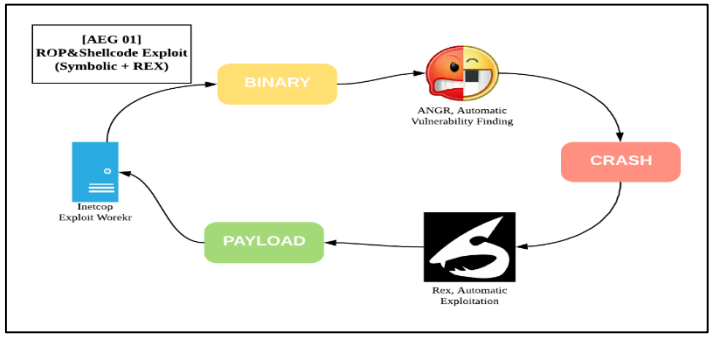
AEG 04(+ 05,06,07) – Concolic (Win EIP Constraint)



[그림5] Win EIP Constraint (Concolic Execution)

AEG 04(+ 05,06,07) – Concolic (Win EIP Constraint)

- Win Function 탐색, Win Function으로 EIP를 조작
- Function Finder를 개발해 Win Function의 위치를 추출
- Angr Symbolic Execution, input dictionary 추출
- AFL Interesting 및 symbolic dictionary 조합하여 Fuzzing
- Pwntool의 pattern 기법 활용, input EIP 영역 확인



[그림2] ROP & Shellcode Exploit (Symbolic Execution + Rex) 구성도

예선 / 본선 취약점 결과

예선					
1차		2차		3차	
번호	결과	번호	결과	번호	결과
1	O	1	X	1	O
2	O	2	O	2	O
3	O	3	X	3	X
4	O	4	O	4	X
5	O	5	X	5	O
6	X	6	O	6	O
7	O	7	X	7	O
8	O	8	O	8	X
9	O	9	X	9	X
10	O	10	X	10	O
11	X	11	O	11	O
12	O	12	O	12	O
13	O	13	O	13	X
14	O	14	X	14	X
15	X	15	X	15	X
합계	**12/15**	**합계**	**7/15**	**합계**	**8/15**

본선					
1라운드		2라운드		3라운드	
번호	결과	번호	결과	번호	결과
1	O	11	X	21	O
2	O	12	O	22	O
3	O	13	O	23	O
4	X	14	O	24	X
5	X	15	O	25	O
6	X	16	X	26	X
7	X	17	X	27	O
8	X	18	X	28	O
9	X	19	X	29	O
10	X	20	X	30	X
합계	**3/10**	**합계**	**4/10**	31	X
				32	X
				33	O
				34	X
				35	X
				36	X
				37	X
				38	X
				39	X
				40	X
				합계	**8/20**

- 예선 총 45개 바이너리 중 27개 풀이.
- 본선 총 40개 바이너리 중 15개 풀이.

5

Q&A

감사합니다